Rigorous enclosures of minimal detectable differences for general ANOVA models

Ali Baharev[*,1], Hermann Schichl[1], Endre Rév[2]

(1) Faculty of Mathematics, University of Vienna
Oskar-Morgenstern-Platz 1, 1090 Vienna, Austria

(2) Department of Chemical and Environmental Process Engineering
Budapest University of Technology and Economics
Pf. 91., 1521 Budapest, Hungary

email: ali.baharev@gmail.com
(*) Author to whom all correspondence should be addressed.

December 14, 2015
`ncbeta.tex`

Contents

Abstract

The computation of minimal detectable differences in general ANOVA designs is the poster child of numerical difficulties. Floating-point underflow and overflow issues, round-off errors introducing large relative errors in the results, drastically increasing computation time and / or the algorithm hanging up, and instability due to numerical cancellation have all been reported. A self-verifying approach is proposed in this paper to systematically test the accuracy of existing floating-point algorithms if the degree of freedom of the denominator of the F test statistic is even. This novel algorithm is available as a standalone command line application, and can be used to verify the accuracy of any other statistical software for computing minimal detectable differences. An interesting property of the proposed method is that it only depends on the four arithmetic operations, the power function, and the exponential function. Numerical examples are presented; the algorithm in the R package `fpow` is proved to be accurate for 6 significant digits in all the checked cases.

Keywords: minimal detectable differences, ANOVA, noncentrality parameter, noncentral F-distribution, self-validating numerical method, interval arithmetic, power of F test

1 Introduction

Motivation The computation of minimal detectable differences for general ANOVA models belonging to a specified type II error probability requires calculations based on the noncentral F-distribution, see for example Johnson and Leone (1977, p. 170). These calculations may involve several numerical difficulties, such as underflow and overflow problems (Benton and Krishnamoorthy, 2003, Ding, 1997, Helstrom and Ritcey, 1985), round-off errors that may introduce large relative errors into the result (Frick, 1990), and drastically increasing computation time and/or hanging up (Benton and Krishnamoorthy, 2003, Chattamvelli, 1995). The algorithms of Norton (1983) and Lenth (1987) are exposed to over/underflow problems; the Appendix 12 of Lorenzen and Anderson (1993, p. 374) is most likely bogus due to overflow problems, see Baharev and Kemény (2008). As a consequence, correct values with guaranteed accuracy are vital to test the accuracy of existing numerical algorithms.

Automatic error analysis Self-validating numerical methods for computing various distributions are described by Wang and Kennedy (1992, 1990, 1994); according to the authors' best knowledge there is just one method, Wang and Kennedy (1995), for computing noncentral F probabilities and percentiles involving automatic error analysis. A self-verifying algorithm is presented in this paper for computing values of minimal detectable differences with guaranteed accuracy if the degree of freedom of the denominator of the F test statistic is even. The proposed method is based on interval arithmetic which we review in the next section.

2 Interval arithmetic

Interval arithmetic is an extension of real arithmetic defined on the set of real intervals, rather than the set of real numbers. According to a survey paper by Kearfott (1996), a form of interval arithmetic perhaps first appeared in Burkill (1924). Modern interval arithmetic was originally invented independently in the late 1950s by several researchers; including Warmus (1956), Sunaga (1958) and finally Moore (1959), who set firm foundations for the field in his many publications, including the foundational book Moore (1966). Since then, interval arithmetic is being used to rigorously solve numerical problems.

Let $\mathbb{IR}$ be the set of all real intervals, and take $\mathbf{x}, \mathbf{y} \in \mathbb{IR}$. We set $\underline{x} := \inf \mathbf{x}$ and $\overline{x} := \sup \mathbf{x}$, such that $\mathbf{x} = [\underline{x}, \overline{x}]$. Furthermore, the width of $\mathbf{x}$ is defined as $\mathrm{wid}(\mathbf{x}) := \overline{x} - \underline{x}$, the radius of $\mathbf{x}$ as $\mathrm{rad}(\mathbf{x}) := \frac{1}{2}\mathrm{wid}(\mathbf{x})$, the magnitude of $\mathbf{x}$ as $|\mathbf{x}| := \max(\underline{x}, \overline{x})$, and the mignitude of $\mathbf{x}$ as $\langle \mathbf{x} \rangle := \min\{|x| \mid x \in \mathbf{y}\}$. For $\mathbf{x}$ bounded we set the midpoint of $\mathbf{x}$ as $\check{x} := \frac{1}{2}(\underline{x} + \overline{x})$. We define the elementary operations for *interval arithmetic* by the rule $\mathbf{x} \diamond \mathbf{y} = \square\{x \diamond y \mid x \in \mathbf{x}, y \in \mathbf{y}\}, \forall \diamond \in \{+, -, \times, \div, \hat{}\}$, where $\square S$ denotes the smallest interval containing the set S. Thus, the ranges of the five elementary interval arithmetic operations are exactly the ranges of the their real-valued counterparts. Although this rule characterizes these operations mathematically, the usefulness of interval arithmetic is due to the *operational definitions* based on interval bounds Hickey et al. (2001). For example, let $\mathbf{x} = [\underline{x}, \overline{x}]$ and $\mathbf{y} = [\underline{y}, \overline{y}]$, it can be easily proved that

$$\begin{aligned}
\mathbf{x} + \mathbf{y} &= [\underline{x} + \underline{y}, \overline{x} + \overline{y}], \\
\mathbf{x} - \mathbf{y} &= [\underline{x} - \overline{y}, \overline{x} - \underline{y}], \\
\mathbf{x} \times \mathbf{y} &= [\min\{\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y}\}, \max\{\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y}\}], \\
\mathbf{x} \div \mathbf{y} &= \mathbf{x} \times 1/\mathbf{y} \text{ if } 0 \notin \mathbf{y}, \text{ where } 1/\mathbf{y} = [1/\overline{y}, 1/\underline{y}], \\
\mathbf{x}^{\mathbf{y}} &= [\min\{\underline{x}^{\underline{y}}, \underline{x}^{\overline{y}}, \overline{x}^{\underline{y}}, \overline{x}^{\overline{y}}\}, \max\{\underline{x}^{\underline{y}}, \underline{x}^{\overline{y}}, \overline{x}^{\underline{y}}, \overline{x}^{\overline{y}}\}], \quad \underline{y} > 0.
\end{aligned}$$

In addition, for a function $\varphi : \mathbb{R} \to \mathbb{R}$ and an interval $\mathbf{x}$ we define

$$\varphi(\mathbf{x}) := \square\{\varphi(x) \mid x \in \mathbf{x}\}.$$

Moreover, if a factorable function f composed of these elementary arithmetic operations and elementary functions $\varphi \in \{\sin, \cos, \exp, \log, \ldots\}$ is given, *bounds on the range* of f can be obtained by replacing the real arithmetic operations and the real functions by their corresponding interval arithmetic counterparts.

The finite nature of computers precludes an exact representation of the *reals*. In practice, the real set, $\mathbb{R}$, is approximated by a finite set $\bar{\mathbb{F}} = \mathbb{F} \cup \{-\infty, +\infty\}$, where $\mathbb{F}$ is the set of floating-point numbers. The set of real intervals is then approximated by the set $\mathbb{I}$ of intervals with bounds in $\bar{\mathbb{F}}$. The power of interval arithmetic lies in its implementation on computers. In particular, *outwardly rounded* interval arithmetic allows *rigorous enclosures* for the ranges of operations and functions. This makes a qualitative difference in scientific computations, since the results are now intervals in which the exact result is guaranteed to lie. Interval arithmetic can be carried out for virtually every expression that can be evaluated with floating-point arithmetic. However, two important points have to be considered: Interval arithmetic is only *subdistributive*, so expressions that are equivalent in real arithmetic differ in interval arithmetic, giving different amounts of overestimation (the amount by which the real range of the function over an interval and the result computed by interval arithmetic differ). Therefore, computations should be arranged so that overestimation of ranges is minimized. Readers are referred to Alefeld and Herzberger (1983), Hickey et al. (2001), Jaulin et al. (2001), Neumaier (1990) for more details on basic interval methods.

Interval Newton's method The univariate interval Newton's method is a method for solving equations and it is able to bound all zeros of a continuously differentiable univariate function in a given interval (initial interval) with both mathematical and computational certainty. Let $f:\mathbf{x}\to\mathbb{R}$ be continuously differentiable, and assume the existence of $\hat{x}\in\mathbf{x}$ with $f(\hat{x})=0$, and let $\tilde{x}\in\mathbf{x}$. Then by the mean value theorem we get

$$f(\hat{x})=0=f(\tilde{x})+f'(\xi)(\hat{x}-\tilde{x}),$$

for some $\xi\in\mathbf{x}$. Therefore,

$$\hat{x}=\tilde{x}-\frac{f(\tilde{x})}{f'(\xi)}.$$

Now let $\mathfrak{f}'$ be an interval extension of f' and $\mathfrak{f}$ be an interval extension of f. Then by the properties of interval arithmetic we get

$$\hat{x}\in\tilde{x}-\frac{\mathfrak{f}(\tilde{x}}{\mathfrak{f}'(\mathbf{x})}=:N(\mathfrak{f},\mathfrak{f}';\mathbf{x},\tilde{x}).$$

The operator N is called univariate interval Newton operator. Using this operator we can define the interval Newton iteration as

$$\mathbf{x}^{(k+1)}=\mathbf{x}^{(k)}\cap N(\mathfrak{f},\mathfrak{f}';\mathbf{x}^{(k)},\check{x}^{(k)}),$$

starting with $\mathbf{x}^{(0)}=\mathbf{x}$. This iteration has the properties that whenever $\mathbf{x}^{(k)}=\emptyset$ for some k, then $\mathbf{x}$ does not contain a zero of f. Otherwise $\hat{x}\in\mathbf{x}^{(k)}$ for all k, and $\mathrm{wid}(\mathbf{x}^{(k+1)})=O(\mathrm{wid}(\mathbf{x}^{(k)})^2)$ locally under mild assumptions on f and $\mathbf{x}^{(0)}$. Furthermore, if for any k we find that $\mathbf{x}^{(k+1)}\subseteq\mathrm{int}(\mathbf{x}^{(k)})$, i.e., that the interval Newton operator maps the box $\mathbf{x}^{(k)}$ into its interior, then $\mathbf{x}^{(k)}$ contains a unique zero of f.

Algorithmic Differentiation The interval Newton operator requires an interval extension $\mathfrak{f}'$ of the derivative of f. For every factorable function f such an extension can be constructed using algorithmic differentiation techniques, e.g., see Berz et al. (1996), Griewank and Corliss (1991), Griewank and Walther (2008). For univariate functions the most efficient approach is via the algebra of differential numbers $\mathscr{D}_1 := \mathbb{R} \times \mathbb{R}$, equipped with the following basic operations: Let $df := (f, f'), dg := (g, g') \in \mathscr{D}_1$, and $\varphi : \mathbb{R} \to \mathbb{R}$. Define

$$\begin{aligned} df \pm dg &:= (f \pm g, f' \pm g'), \\ df \cdot dg &:= (f \cdot g, f' \cdot g + f \cdot g'), \\ df / dg &:= (f/g, (f' \cdot g - f \cdot g')/g^2), \quad g \neq 0, \\ df^{dg} &:= (f^g, f^g \cdot (g' \cdot \log(f) + g \cdot f'/f)), \quad f > 0, \\ \varphi(df) &:= (\varphi(f), \varphi'(f) \cdot f'). \end{aligned} \tag{1}$$

The set of real numbers is embedded in $\mathscr{D}_1$ by $r \mapsto (r, 0)$.

If $\tilde{f}(x)$ is an expression representing f using arithmetic operations and elementary functions, we can use $\tilde{f}$ to calculate $(y, y') = \tilde{f}((x, 1))$ on $\mathscr{D}_1$ by replacing the operations and elementary functions in $\tilde{f}$ by their counterparts on $\mathscr{D}_1$, and then $y' = f'(x)$.

This approach can be generalized to compute an interval extension $\mathfrak{f}'$ of f' by defining the algebra of interval differential numbers $\mathbb{I}\mathscr{D}_1 := \mathbb{IR} \times \mathbb{IR}$ and introducing again the operations (1) on $\mathbb{I}\mathscr{D}_1$ now using interval arithmetic operations in the components of the interval differential numbers. Using this algebra and an expression $\tilde{f}$ for f, we get by computing $(\mathbf{y}, \mathbf{y}') = \tilde{f}((\mathbf{x}, 1))$ an enclosure $\mathbf{y}' \supseteq f'(\mathbf{x})$ and thereby an interval extension $\mathfrak{f}'$ of f'.

3 The proposed method

Formulas The F test statistic used in the analysis of variance problems follows the non-central F distribution when the null hypothesis is false, i.e., $F_0 = F_{nc}(\nu_1, \nu_2, \lambda)$, where ν_1 and ν_2 are the degrees of freedom of the nominator and the denominator, respectively, of the F test statistic; λ is the non-centrality parameter. This non-centrality parameter is related to the size of effects. The probability of the Type II error β (not detecting an effect) is

$$\beta = P\left[F_0 \leq F_\alpha(\nu_1, \nu_2)\right] = P\left[F_{nc}(\nu_1, \nu_2, \lambda) \leq F_\alpha(\nu_1, \nu_2)\right], \tag{2}$$

where α is the significance level of the test. The power calculation consists of calculating the power ($Power = 1 - \beta$) for certain effect sizes. When it is used in the reversed way, the power is fixed (e.g. $Power = 0.9$), and the size of the effect is calculated; this is considered as effect of detectable size, see Lorenzen and Anderson (1993).

The cdf $F(w, \nu_1, \nu_2, \lambda)$ of the noncentral F-distribution with ν_1, ν_2 degrees of freedom and noncentrality parameter λ, and the cdf $I_x(a, b; \lambda)$ of the noncentral beta distribution with shape parameters a and b and noncentrality parameter λ are related as follows:

$$F(w, \nu_1, \nu_2, \lambda) = I_x(a, b; \lambda), \tag{3}$$

where $a = \frac{\nu_1}{2}$, $b = \frac{\nu_2}{2}$, and $x = \frac{\nu_1 w}{\nu_1 w + \nu_2}$. We will work with the noncentral beta distribution from now on.

The incomplete noncentral beta function ratio $I_x(a,b;\lambda)$ for $0 \le x \le 1, a > 0, b > 0, \lambda \ge 0$ is defined as

$$I_x(a,b;\lambda) = \sum_{i=0}^{\infty} \frac{e^{-(\lambda/2)}(\lambda/2)^i}{i!} I_x(a+i,b), \tag{4}$$

where $I_x(a,b)$ is the usual incomplete beta function ratio,

$$I_x(a,b) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \int_0^x t^{a-1}(1-t)^{b-1}\mathrm{d}t, \tag{5}$$

and $\Gamma(a)$ is the (complete) gamma function

$$\Gamma(a) = \int_0^{\infty} t^{a-1}e^{-t}\mathrm{d}t, \quad a > 0, \tag{6}$$

see for example Johnson et al. (1995).

Equation (4) cannot be used directly for actual numerical computations. The proposed method uses the the closed formula

$$I_x(a,b) = x^a \left(1 + \sum_{n=1}^{b-1} \left(\prod_{m=1}^{n} \frac{a+m-1}{m}\right)(1-x)^n\right) \tag{7}$$

by Singh and Relyea (1992) (misprint corrected by Chattamvelli (1995)) for the computation of the cdf of the central beta distribution, and the closed formula

$$I_x(a,b;\lambda) = e^{-(\lambda/2)(1-x)} \sum_{i=0}^{b-1} \frac{((\lambda/2)(1-x))^i}{i!} I_x(a+i,b-i) \tag{8}$$

by Sibuya (1967) (and later published by Johnson et al. (1995)) for the cdf of the noncentral beta distribution. The shape parameter b must be integer. It is very interesting that the evaluation of (8), using (7), is possible in finitely many steps, requiring only the four arithmetic operations, the power function, and the exponential function; the formulas do not depend on any function for computing statistical distributions.

Detectable differences for a specified type II error probability β are determined by the noncentrality parameter λ for which the cdf value of the noncentral beta distribution equals β

$$I_{x_{1-\alpha}}(a,b;\lambda) = \beta, \tag{9}$$

where $x_{1-\alpha}$ is the upper α quantile of the central beta distribution with shape parameters $a = \nu_1/2$ and $b = \nu_2/2$; α denotes the allowed type I error probability. The proposed method therefore solves (9) with interval Newton's method for λ, given a, b, α and β.

Possible outcomes of the interval Newton's method Rigorous enclosures of x_α and λ are computed with univariate interval Newton's method using automatic differentiation, see Hammer et al. (1995). The procedure can have three different outcomes depending on the function and the initial interval. It is always unambiguous which case is obtained.

a) All solutions are rigorously enclosed, and each enclosure contains a unique zero. The result is the list of these enclosures. (In our case, there can be at most one zero, i.e., the list has at most one element.)
b) It is proved with mathematical rigor that the function cannot have any zeros in the initial interval.
c) There is at least one enclosure among the resulting enclosures of zeros which may contain a zero but verification of existence and/or uniqueness of a zero in that particular enclosure failed.

Pseudo-code of the proposed algorithm

Step 0. The input of the proposed algorithm is the output of another algorithm, $x_{1-\alpha}$ and λ, that we want to test. The initial intervals x_0 and λ_0 are obtained from the inputs $x_{1-\alpha}$ and λ by inflating them as follows:

$$x_0 = [(1-\varepsilon_x)x_{1-\alpha},\ (1+\varepsilon_x)x_{1-\alpha}]\,, \text{ and } \lambda_0 = [(1-\varepsilon_\lambda)\lambda,\ (1+\varepsilon_\lambda)\lambda]\,, \tag{10}$$

where the inflation parameters ε_x and ε_λ are sufficiently small user-defined real numbers, for example 10^{-6}.

Step 1. A narrow interval containing the theoretically correct value of $x_{1-\alpha}$ is computed with interval Newton's method, using (7). If interval Newton's method results in case b (guaranteed *not* to have a solution) or case c (verification failed), then exit with the corresponding error message.

Step 2. Equation (9) is solved for λ with interval Newton's method, using (8) and (7). The possible outcomes are: A rigorous enclosure of the true value of λ is obtained (case a), or λ_0 is proved not to contain the correct value (case b), or the verification fails (case c).

If the initial intervals contain the theoretically correct value and the interval Newton's method succeeds in proving it (case a in both Step 1 and Step 2), then the algorithm under test is at least ε_x or ε_λ accurate, respectively, in that studied case. Analogously, if the initial intervals do not contain the theoretically correct value and the interval Newton's method reliably proves that (case b), then the accuracy of the algorithm under test is less than ε_x or ε_λ accurate, respectively. These facts make it possible to systematically test the accuracy of existing algorithms in an automated manner. The very rare but unfortunate case c, when reliable conclusion cannot be drawn and further investigation may be needed, can usually be remedied by simply changing (increasing) the inflation parameters.

Implementation details The above algorithm is implemented in C-XSC using the module `nlfzero`. C-XSC is available from `http://www.xsc.de`, and it is documented in the book of Hammer et al. (1995). C-XSC implements interval Newton's method in one variable using automatic differentiation. No higher precision internal data format is used. All computations are done using the IEEE double format (64 bit). Evaluation of (7) and (8) could be significantly speeded up since it involves several redundant operations. For example, it is possible to reduce the number of switches between rounding modes considering that $a > 0$, $b > 0$, $0 \le x \le 1$ always hold. However, (7) and (8) are used directly, and no efforts were made to decrease the computation time because it was found to be satisfactory by the authors.

4 Usage

The standalone command line application reads from the standard input, and writes to the standard output line-by-line. If the data to be verified is given in a text file called `input.txt`, then the program can be executed like this:

```
mindiffver <input.txt >output.txt
```

The verified results will be written to the `output.txt` file. The format of the input and output are detailed in the next paragraphs.

Input format Each line of the input is supposed to have the following format:

$$a \quad b \quad x_{1-\alpha} \quad \lambda \quad \alpha \quad \beta \quad \varepsilon_x \quad \varepsilon_\lambda$$

where the items are separated by arbitrary whitespace. The search intervals x_0 and λ_0 are computed according to (10).

Parameter ranges for the input The following ranges are checked by the program:

1. All input values must be strictly positive.
2. b must be integer.
3. $x, \alpha, \beta, \varepsilon_x, \varepsilon_\lambda < 1$ must hold.
4. $\varepsilon_x \ge \delta_x$, and $\varepsilon_\lambda \ge \delta_\lambda$ must hold, where $\delta_x = 10^{-12}$ and $\delta_\lambda = 10^{-10}$ are the currently set tolerances for x and λ in the interval Newton iteration.

Assumptions (not enforced) The parameters are assumed to lie in the domain that is relevant for practical applications, roughly: $a \le 25, b \le 500, 0.01 \le \alpha, \beta \le 0.99$; the inflation parameters are also assumed to be sane, say $< 10^{-4}$. Violating these assumptions may cause performance degradation and the algorithm may start reporting failures but incorrect results will never appear in the output.

Output format There are three possible outcomes, corresponding to the outcomes of the interval Newton method.

a) If the input line contains a solution and the interval Newton method is successful in verifying it, then the output is a line matching the format of the input line (items are guaranteed to be tab separated) where x and λ are guaranteed to have the precision given in the last two columns (currently set to 10^{-12} and 10^{-10} relative error, respectively).
b) If the input line does *not* contain a solution and the interval Newton method is successful in verifying it, then the output is a single line saying: `The search interval ... is verified NOT to contain a zero.`
c) In all other cases a single line error message starting with `Failed ...` is printed. See the `failures.txt` input file that systematically triggers all known failure modes, except for the first and the last line of that file (those two lines must succeed).

5 Numerical results

The corrected form of Appendix 12 of Lorenzen and Anderson (1993, p. 374), i.e., Table 3 of Baharev and Kemény (2008), fully covers the parameter range that is relevant for practical applications. This table (except rows for which b is not integer, i.e., $b = 0.5, 1.5, 2.5, 3.5$) is recomputed with the self-verifying algorithm proposed in this paper. The input (tested) values of $x_{0.95}$ and λ are computed by the algorithm of Baharev and Kemény (2008), implemented in R (R Development Core Team, 2015) and available as the package `fpow`; the inflation parameter values ε_x and ε_λ are both set to 10^{-6}. Table 1 contains the solution of $I_x(a,b) = 1-\alpha$ for x, given a, b and $\alpha = 0.05$. Table 2 contains the solution of (9), where the verified $x_{0.95}$ is obtained from the previous step.

The computations have been carried out with the following hardware and software configuration. Processor: Intel(R) Core(TM) i5-3320M CPU at 2.60GHz; operating system: Ubuntu 14.04.3 LTS with 3.13.0-67-generic kernel; compiler: gcc 4.8.4, compiler optimization flag: -O3; C-XSC 2.5.4 configuration left on the default values given by the install script.

The overall computation required less than 9 seconds. The output of the proposed algorithm is $x_{0.95}$ verified up to 12 significant digits, and λ verified up to 10 significant digits. (Only 6 digits are shown in the Tables 1 and 2.) The algorithm of Baharev and Kemény (2008), implemented on the top of the built-in functions of R, is proved to be accurate for 6 significant digits in the investigated domain with mathematical certainty.

6 Acknowledgement

The research was funded by the Austrian Science Fund (FWF): P27891-N32. The authors are grateful to Professor Sándor Kemény for reading an earlier draft of this paper and for making valuable comments and suggestions.

Table 1: The upper $\alpha = 0.05$ quantiles, the solution of $I_x(a,b) = 1-\alpha$ for x. All given digits are verified with the last digit rounded to the nearest.

b	a								
	0.5	1	1.5	2	2.5	3	5	10	25
1	9.02500e-01	9.50000e-01	9.66383e-01	9.74679e-01	9.79692e-01	9.83048e-01	9.89794e-01	9.94884e-01	9.97950e-01
2	6.58372e-01	7.76393e-01	8.31750e-01	8.64650e-01	8.86622e-01	9.02389e-01	9.37150e-01	9.66681e-01	9.86158e-01
3	4.99474e-01	6.31597e-01	7.04013e-01	7.51395e-01	7.85230e-01	8.10745e-01	8.71244e-01	9.28130e-01	9.69022e-01
4	3.99294e-01	5.27129e-01	6.03932e-01	6.57408e-01	6.97399e-01	7.28662e-01	8.07097e-01	8.87334e-01	9.49692e-01
5	3.31756e-01	4.50720e-01	5.26623e-01	5.81803e-01	6.24472e-01	6.58739e-01	7.48632e-01	8.47282e-01	9.29506e-01
6	2.83463e-01	3.93038e-01	4.65976e-01	5.20703e-01	5.64102e-01	5.99689e-01	6.96463e-01	8.09135e-01	9.09126e-01
7	2.47316e-01	3.48164e-01	4.17435e-01	4.70679e-01	5.13741e-01	5.49642e-01	6.50188e-01	7.73308e-01	8.88911e-01
8	2.19284e-01	3.12344e-01	3.77834e-01	4.29136e-01	4.71285e-01	5.06901e-01	6.09138e-01	7.39886e-01	8.69067e-01
9	1.96926e-01	2.83129e-01	3.44972e-01	3.94163e-01	4.35104e-01	4.70087e-01	5.72619e-01	7.08799e-01	8.49712e-01
10	1.78687e-01	2.58866e-01	3.17294e-01	3.64359e-01	4.03954e-01	4.38105e-01	5.40005e-01	6.79913e-01	8.30912e-01
11	1.63528e-01	2.38404e-01	2.93680e-01	3.38681e-01	3.76883e-01	4.10099e-01	5.10752e-01	6.53069e-01	8.12701e-01
12	1.50733e-01	2.20922e-01	2.73308e-01	3.16340e-01	3.53157e-01	3.85390e-01	4.84396e-01	6.28099e-01	7.95094e-01
13	1.39791e-01	2.05817e-01	2.55557e-01	2.96734e-01	3.32202e-01	3.63442e-01	4.60549e-01	6.04844e-01	7.78091e-01
14	1.30326e-01	1.92636e-01	2.39958e-01	2.79396e-01	3.13568e-01	3.43825e-01	4.38883e-01	5.83155e-01	7.61683e-01
15	1.22059e-01	1.81036e-01	2.26143e-01	2.63957e-01	2.96893e-01	3.26193e-01	4.19120e-01	5.62893e-01	7.45857e-01
20	9.26567e-02	1.39108e-01	1.75534e-01	2.06725e-01	2.34411e-01	2.59467e-01	3.41807e-01	4.79012e-01	6.74797e-01
30	6.25175e-02	9.50339e-02	1.21191e-01	1.44090e-01	1.64826e-01	1.83943e-01	2.49305e-01	3.68153e-01	5.65062e-01
40	4.71693e-02	7.21575e-02	9.25215e-02	1.10553e-01	1.27053e-01	1.42414e-01	1.96078e-01	2.98634e-01	4.85211e-01
50	3.78708e-02	5.81551e-02	7.48160e-02	8.96715e-02	1.03353e-01	1.16167e-01	1.61545e-01	2.51097e-01	4.24830e-01
100	1.90711e-02	2.95130e-02	3.82269e-02	4.61073e-02	5.34614e-02	6.04365e-02	8.58514e-02	1.39660e-01	2.61259e-01
250	7.66110e-03	1.19114e-02	1.54926e-02	1.87595e-02	2.18331e-02	2.47712e-02	3.56731e-02	5.98536e-02	1.20972e-01
500	3.83600e-03	5.97355e-03	7.78040e-03	9.43349e-03	1.09931e-02	1.24879e-02	1.80690e-02	3.06519e-02	6.38108e-02

Table 2: The noncentrality parameter λ, the solution to Equation (9); probability of the type I and type II errors are 0.05 and 0.10 respectively. All given digits are verified with the last digit rounded to the nearest.

b	*a*								
	0.5	1	1.5	2	2.5	3	5	10	25
1	4.61803e+01	9.00517e+01	1.33936e+02	1.77823e+02	2.21712e+02	2.65601e+02	4.41161e+02	8.80065e+02	2.19678e+03
2	1.93236e+01	3.04220e+01	4.08997e+01	5.11554e+01	6.13048e+01	7.13948e+01	1.11490e+02	2.11206e+02	5.09746e+02
3	1.53086e+01	2.20966e+01	2.82383e+01	3.41350e+01	3.99085e+01	4.56104e+01	6.80824e+01	1.23556e+02	2.89087e+02
4	1.37822e+01	1.90179e+01	2.36054e+01	2.79378e+01	3.21380e+01	3.62590e+01	5.23587e+01	9.17573e+01	2.08805e+02
5	1.29870e+01	1.74388e+01	2.12434e+01	2.47874e+01	2.81929e+01	3.15140e+01	4.43773e+01	7.55652e+01	1.67745e+02
6	1.25009e+01	1.64830e+01	1.98196e+01	2.28917e+01	2.58210e+01	2.86621e+01	3.95760e+01	6.57902e+01	1.42832e+02
7	1.21736e+01	1.58437e+01	1.88700e+01	2.16290e+01	2.42420e+01	2.67637e+01	3.63771e+01	5.92546e+01	1.26087e+02
8	1.19383e+01	1.53865e+01	1.81924e+01	2.07289e+01	2.31168e+01	2.54110e+01	3.40954e+01	5.45774e+01	1.14042e+02
9	1.17611e+01	1.50436e+01	1.76849e+01	2.00552e+01	2.22750e+01	2.43990e+01	3.23869e+01	5.10640e+01	1.04947e+02
10	1.16228e+01	1.47769e+01	1.72907e+01	1.95324e+01	2.16217e+01	2.36137e+01	3.10600e+01	4.83276e+01	9.78285e+01
11	1.15120e+01	1.45636e+01	1.69759e+01	1.91149e+01	2.11002e+01	2.29868e+01	3.00000e+01	4.61356e+01	9.20991e+01
12	1.14212e+01	1.43892e+01	1.67186e+01	1.87739e+01	2.06743e+01	2.24748e+01	2.91336e+01	4.43398e+01	8.73842e+01
13	1.13454e+01	1.42440e+01	1.65045e+01	1.84902e+01	2.03200e+01	2.20489e+01	2.84125e+01	4.28416e+01	8.34334e+01
14	1.12812e+01	1.41211e+01	1.63236e+01	1.82505e+01	2.00207e+01	2.16891e+01	2.78028e+01	4.15723e+01	8.00727e+01
15	1.12262e+01	1.40158e+01	1.61686e+01	1.80453e+01	1.97645e+01	2.13810e+01	2.72807e+01	4.04832e+01	7.71776e+01
20	1.10375e+01	1.36562e+01	1.56397e+01	1.73453e+01	1.88905e+01	2.03304e+01	2.54976e+01	3.67467e+01	6.71490e+01
30	1.08550e+01	1.33096e+01	1.51311e+01	1.66726e+01	1.80510e+01	1.93210e+01	2.37807e+01	3.31188e+01	5.72176e+01
40	1.07660e+01	1.31411e+01	1.48842e+01	1.63462e+01	1.76437e+01	1.88312e+01	2.29461e+01	3.13414e+01	5.22527e+01
50	1.07132e+01	1.30415e+01	1.47383e+01	1.61535e+01	1.74032e+01	1.85419e+01	2.24526e+01	3.02856e+01	4.92627e+01
100	1.06093e+01	1.28456e+01	1.44516e+01	1.57748e+01	1.69308e+01	1.79737e+01	2.14819e+01	2.81960e+01	4.32297e+01
250	1.05479e+01	1.27301e+01	1.42828e+01	1.55519e+01	1.66527e+01	1.76392e+01	2.09095e+01	2.69549e+01	3.95540e+01
500	1.05276e+01	1.26919e+01	1.42270e+01	1.54783e+01	1.65608e+01	1.75288e+01	2.07203e+01	2.65432e+01	3.83154e+01

References

G. Alefeld and J. Herzberger. *Introduction to Interval Computations*. Academic Press, New York, NY, 1983.

A. Baharev and S. Kemény. On the computation of the noncentral F and noncentral beta distribution. *Statistics and Computing*, 18:333–340, 2008.

Denise Benton and K. Krishnamoorthy. Computing discrete mixtures of continuous distributions: noncentral chisquare, noncentral t and the distribution of the square of the sample multiple correlation coefficient. *Computational Statistics & Data Analysis*, 43(2):249 – 267, 2003.

Martin Berz, Christian Bischof, George Corliss, and Andreas Griewank. *Computational Differentiation: Techniques*. Applications, and Tools. SIAM, Philadelphia, Penn, 1996.

J. C. Burkill. Functions of Intervals. *Proceedings of the London Mathematical Society*, 22: 375–446, 1924.

R. Chattamvelli. On the doubly noncentral F distribution. *Computational Statistics & Data Analysis*, 20(5):481–489, 1995.

Cherng G. Ding. On using Newton's method for computing the noncentrality parameter of the noncentral F distribution. *Communications in Statistics - Simulation and Computation*, 26 (1):259–268, 1997.

H. Frick. AS R84. A remark on Algorithm AS 226, computing non-central beta probabilities. *Appl. Stat.*, 39(2):311–312, 1990.

A. Griewank and G. F. Corliss. *Automatic Differentiation of Algorithms*. SIAM Publications, Philadelphia, 1991.

Andreas Griewank and Andrea Walther. *Evaluating derivatives: principles and techniques of algorithmic differentiation*. Siam, 2008.

Rolf Hammer, Matthias Hocks, Ulrich Kulisch, and Dietmar Ratz. *C++ Toolbox for Verified Computing I, Basic Numerical Problems*. Springer-Verlag, 1995.

Carl W. Helstrom and James A. Ritcey. Evaluation of the noncentral F-distribution by numerical contour integration. *SIAM Journal on Scientific and Statistical Computing*, 6(3):505–514, 1985.

T. J. Hickey, Q. Ju, and M. H. Van Emden. Interval Arithmetic: from Principles to Implementation. *Journal of the ACM (JACM)*, 48(5):1038–1068, 2001.

L. Jaulin, M. Kieffer, O. Didrit, and E. Walter. *Applied Interval Analysis*. Springer, first edition, 2001. ISBN 1-85233-219-0.

N. L. Johnson, S. Kotz, and N. Balakrishnan. *Continuous Univariate Distributions*, volume 2. John Wiley & Sons, Inc., New York, second edition, 1995.

Norman L Johnson and Fred C Leone. *Statistics and experimental design: in engineering and the physical science*. John Wiley & Sons, New York, second edition, 1977.

B. R. Kearfott. Interval Computations: Introduction, Uses, and Resources. *Euromath Bulletin*, 2(1):95–112, 1996.

Russell V. Lenth. Algorithm AS 226: Computing noncentral beta probabilities. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 36(2):241–244, 1987.

T. J. Lorenzen and V. L. Anderson. *Design of experiments: a no-name approach*. Marcel Dekker, Inc., New York, 1993.

R. E. Moore. Automatic Error Analysis in Digital Computation. Technical Report LMSD-84821, Missiles and Space Division, Lockheed Aircraft Corporation, Sunnyvale, California, USA, 1959.

R. E. Moore. *Interval Analysis*. Prentice Hall, Englewood Cliffs, NJ, 1966.

Arnold Neumaier. *Interval Methods for Systems of Equations*. Cambridge Univ. Press, Cambridge, 1990.

Vic Norton. A simple algorithm for computing the non-central F distribution. *Appl. Stat.*, 32 (1):84–85, 1983.

R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2015. URL `http://www.R-project.org`. ISBN 3-900051-07-0.

M. Sibuya. On the noncentral beta distribution function. Unpublished manuscript, the equation can be found in the book of Johnson, Kotz, and Balakrishnan (1995), p. 485, Eq. (30.12), 1967.

Karan P. Singh and George E. Relyea. Computation of noncentral F probabilities: A computer program. *Computational Statistics & Data Analysis*, 13(1):95 – 102, 1992. The misprint on p. 97 was corrected by Chattamvelli (1995).

T. Sunaga. Theory of an Interval Algebra and its Applications to Numerical Analysis. *RAAG Memoirs*, 2:29–46, 1958.

Morgan Wang and W. J. Kennedy. A numerical method for accurately approximating multivariate normal probabilities. *Computational Statistics & Data Analysis*, 13(2):197 – 210, 1992.

Morgan Wang and William J. Kennedy. Comparison of algorithms for bivariate normal probability over a rectangle based on self-validated results from interval analysis. *Journal of Statistical Computation and Simulation*, 37(1-2):13–25, 1990.

Morgan C. Wang and William J. Kennedy. Self-validating computations of probabilities for selected central and noncentral univariate probability functions. *Journal of the American Statistical Association*, 89(427):878–887, 1994.

Morgan C. Wang and William J. Kennedy. A self-validating numerical method for computation of central and non-central F probabilities and percentiles. *Statistics and Computing*, 5(2): 155–163, 1995.

M. Warmus. Caculus of Approximations. *Bulletin de l'Académie Polonaise des Sciences*, IV(5): 253–259, 1956.