BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **11**, No 2

Sofia • 2011

Computing of Special Functions with Arbitrary Precision in the Environment of .NET Framework

Velichko Dzhambov, Stanislav Drangajov

Institute of Information and Communication Technologies, 1113 Sofia

E-mails: sdrangajov@cc.bas.bg vili_jambov@abv.bg

Abstract: *The paper describes the methods and algorithms used for realization of special functions computing with arbitrary precision in the environment of .NET Framework. .NET Framework C# is used as a tool, with the help of the MPIR library. An example is demonstrated, with a program, using the current state of the special functions library realization. Some perspectives for future development are outlined.*

Keywords: *High precision computation, special functions, computational mathematics.*

1. Introduction

Arbitrary precision computations are not a self-purpose. They are related to receiving precise values when solving mathematical models in different areas, including, for instance, non-linear dynamic systems. But due to their essence, similar calculations are not intended for direct real time control of quickly running technological processes, i. e., widely spread industrial production processes. The present paper describes the first stage of the realization of a system for arbitrary precision computing in the environment of .NET Framework, concerning special functions calculations. The used methods and algorithms for their realization are described

The library MPIR [19] is used as a tool. It is a detached version of the arbitrary precision mathematical calculations library, based on GMP (GNU MP – library for arbitrary precision mathematics of GNU). X-MPIR ensures interface in .NET

Framework C# to a library of previously compiled functions of MPIR, realized in C. To our knowledge the latest overview of the special functions calculation is given in Chapter 4 of [1]. In monograph [2] there is a detailed list of sources, which may be used for contingent check up. It is to be noted that intentionally preference is given to the methods, using quickly convergent series in the present realization of the special function library, wherever this is possible and/or practical. In most cases alterative methods exist such as continued fractions, integral presentation, using of iterative relations, and so on. But for computing with arbitrary precision the possibility for simple error evaluation is crucial. The approach adopted is not free of its specific requirements, for instance ensuring consistency of the main series and asymptotic presentation. About the specificity of the asymptotic presentation with divergent series see also [3]. Nonetheless this approach seems to be simpler in the arbitrary precision context, which is anyway apriori specified and should be easily calculated.

2. Realization and methods used

The realization of the functions, at present, is for a real argument. A basic version of a program-calculator is created implementing immediate usage of the library for the purpose of testing the functions. It is described in details in section 3 “Testing calculator”. The generally accessible sources used for the methods realized are [1- 6]. Reference [4] is most intensively used although even there the references “Methods of computation” are not always sufficient, but this does not belittle the exclusive value of the writing for any calculator. Sites [20-22] provide good initial references. Possible sources about specific special functions and constants, as well as respective computational methods when they are non-standard, e.g. power series and asymptotic presentations, are given in place in this section. Many of the functions in the library are internally used to express other functions, e.g. in a respective range or an index type, and some of them, as well as calculating the numerator and denominator in the Bernoulli numbers, are not still represented in the calculator. The special cases of the hyper geometric function are a typical example:

$$ {}_pF_q\left(a_1,\cdots,a_p;b_1,\ldots,b_q;z\right) \;=\; \sum_{n=0}^{\infty}\frac{\left(a_1\right)_n\ldots\left(a_p\right)_n}{\left(b_1\right)_n\ldots\left(b_q\right)_n}\frac{z^n}{n!}. $$

Only the special cases are used in the representations below, where $p = 0$, $q = 1$, and $p = 1$, $q = 1$, at which ($p < q + 1$) series is everywhere convergent, if $b = b_1$ is not a negative integer. However at great argument values other presentations are used.

2.1. Constants

Two variants of using the constants are applicable from a program point of view:

- storing in static fields which are initialized at first usage of the class and then they are immediately used when needed;

- calling functions which return the result required with the current precision for the calculations.

The first one is faster under the condition that the specified precision is not changed in the frame of the given calculation or the constants are computed with sufficiently high precision, which would not be overridden at computation. The second method is used for the moment

After accurate realization of elementary functions constants π, e, ln2, ln10, log2, log e, roots and powers of integers and their various combinations including arithmetic operations are available. The constant with the current precision of computing is returned at calling the respective functions. The algorithm proposed by B r e n t and M c M i l l a n in [7] is used for Euler's constant γ.

Brent and McMillan algorithm:

Suppose we want to compute Euler's constant γ with precision up to d decimal digits. If we choose n to be the greatest integer which is less than $c + (1/4)\ln 10d$ with an appropriate constant c, then

$$\left|\gamma - \frac{U(n)}{V(n)}\right| < \pi e^{4-4c} 10^{-d},$$

where $U(n)$ and $V(n)$ are computed as follows:

Define

$$A_k = \left(\frac{n^k}{k!}\right)(H_k - \ln k), \qquad U_k = \sum_{j=0}^{k} A_j,$$

$$B_k = \left(\frac{n^k}{k!}\right)^2, \qquad V_k = \sum_{j=0}^{k} B_j,$$

where H_k is the k-th harmonic number $(1 + 1/2 + \ldots + 1/k)$. Then $A_0 = -\ln n$, $B_0 = 1$, $U_0 = A_0$, $V_0 = 1$ and for $k = 1, 2, \ldots$, we receive

$$B_k = B_{k-1}\frac{n^2}{k^2}, \qquad A_k = \frac{\left(A_{k-1}\frac{n^2}{k} + B_k\right)}{k},$$

$$U_k = U_{k-1} + A_k, \quad V_k = V_{k-1} + B_k.$$

2.2. Elementary functions

Numerous sources exist for computing elementary functions. The paper of B r e n t [8] should be especially noted. The realization of the elementary transcendental functions will not be considered in details. It should be just noted that for calculating logarithm with base 2, an algorithm is used, giving sequentially the digits, in an iterative and not recursive variant, which is appropriate for arbitrary precision calculations.

2.3. Gamma function [13]

An approximation method described on [9] is used, after respective scaling of the argument.

For an argument in the interval [1, 2) the gamma function is calculated in the following way [13]:

$$x! = \Gamma(x+1) = (x+a)^{x+\frac{1}{2}} e^{-(x+a)} \sqrt{2\pi} \left[c_0 + \sum_{k=1}^{a-1} \frac{c_k}{x+k} + \varepsilon(x) \right].$$

Here a is a positive precision controlling parameter (see below) and the coefficients c_k are determined as follows:

$$c_0 = 1,$$

$$c_k = \frac{1}{\sqrt{2\pi}} \frac{(-1)^{k+1}}{(k-1)!} (-k+a)^{k-\frac{1}{2}} e^{-k+a}, \quad k = 1, 2, \ldots, a-1.$$

The relative error for $x > 0$ and $a > 2$ is $\varepsilon \le a^{-\frac{1}{2}} (2\pi)^{-\left(a+\frac{1}{2}\right)}$, so after leaving out the first multiplier, that raises the estimation, we get

$$a = -\frac{\log \varepsilon}{\log 2\pi} \approx -1.84 \log \varepsilon.$$

If we want precision of n meaning digits, then $\varepsilon = 10^{-n}$ and hence

$$a = \frac{1}{\log 2\pi} n \approx 1.26 n .$$

For negative arguments $\Gamma(x) = \pi / [\sin(\pi * x) * \Gamma(1-x)]$ is used.

For positive arguments:

If $x < 1$, then $\Gamma(x) = \Gamma(x+1)/x$ is used. If $x \ge 2$, then $x = y + n$, where $y \in [1, 2)$ and $\Gamma(x) = (y+n-1) \ldots (y+1) y \Gamma(y)$.

2.4. Incomplete gamma functions, probability integrals

Standard series and asymptotic decompositions are used.

$$\gamma(s, x) = \int_0^x t^{s-1} e^{-t} dt ,$$

$$\Gamma(s, x) = \int_x^\infty t^{s-1} e^{-t} dt ,$$

$$\gamma(s, x) + \Gamma(s, x) = \Gamma(s).$$

For small x the presentation $\gamma(s, x) = x^s s^{-1} {}_1F_1(s; s+1; -x)$ is used, and for big ones – the asymptotic presentation

$$\Gamma(s, x) = x^{s-1} e^{-x} \left(1 + \frac{s-1}{x} + \frac{(s-1)(s-2)}{x^2} + \ldots \right).$$

$\gamma(0, x)$ is not definite, but for $s \to 0$ $\Gamma(s, x)$ has a limit

$$\Gamma(0, x) = \lim_{s \to 0}\left[\Gamma(s) - \frac{1}{s} - \left(\gamma(s, x) - \frac{1}{s}\right)\right] = -\gamma - \ln x - \sum_{n=1}^{\infty}(-1)^n \frac{x^n}{n(n!)}.$$

For the time moment, the asymptotic presentation of the first argument is not included.

$$erf(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2} dt,$$

$$erfc(x) = \frac{2}{\sqrt{\pi}}\int_x^{\infty} e^{-t^2} dt,$$

$$erf(x) + erfc(x) = 1$$

The following presentation is used

$$erf(x) = \frac{2}{\sqrt{\pi}}\sum_{n=0}^{\infty}\frac{(-1)^n x^{2n+1}}{n!(2n+1)},$$

and for a big argument value

$$erfc(x) \sim \frac{e^{-x^2}}{\sqrt{\pi}\,x}\sum_{n=0}^{\infty}(-1)^n \frac{1.3\cdots(2n-1)}{\left(2x^2\right)^n}.$$

2.5. Fresnel's integrals

Standard series and asymptotic decompositions are used.
For calculating Fresnel's integrals

$$C(x) = \int_0^x \cos\left(\frac{1}{2}\pi t^2\right)dt,\ S(x) = \int_0^x \sin\left(\frac{1}{2}\pi t^2\right)dt$$

with not large argument presentations with series are used:

$$C(x) = \cos\left(\frac{1}{2}\pi x^2\right)\sum_{n=0}^{\infty}\frac{(-1)^n \pi^{2n}}{1.3\ldots(4n+1)}x^{4n+1} +$$

$$+ \sin\left(\frac{1}{2}\pi x^2\right)\sum_{n=0}^{\infty}\frac{(-1)^n \pi^{2n+1}}{1.3...(4n+3)}x^{4n+3}$$

$$S(x) = -\cos\left(\frac{1}{2}\pi x^2\right)\sum_{n=0}^{\infty}\frac{(-1)^n \pi^{2n+1}}{1.3\ldots(4n+3)}x^{4n+3} +$$

$$+ \sin\left(\frac{1}{2}\pi x^2\right)\sum_{n=0}^{\infty}\frac{(-1)^n \pi^{2n}}{1.3\ldots(4n+1)}x^{4n+1}$$

For big argument values, asymptotic series are used.

$$C(x) = \frac{1}{2} + f(x)\sin\left(\frac{1}{2}\pi x^2\right) - g(x)\cos\left(\frac{1}{2}\pi x^2\right),$$

$$S(x) = \frac{1}{2} - f(x)\cos\left(\frac{1}{2}\pi x^2\right) - g(x)\sin\left(\frac{1}{2}\pi x^2\right),$$

where

$$f(x)\sim\frac{1}{\pi x}\sum_{n=0}^{\infty}(-1)^n\frac{1.3\ldots(4n-1)}{(\pi x^2)^{2n}},$$

$$g(x)\sim\frac{1}{\pi^2 x^3}\sum_{n=0}^{\infty}(-1)^n\frac{1.3\ldots(4n+1)}{(\pi x^2)^{2n}}.$$

2.6. Bessel functions of first and second kind, modified Bessel functions, Airy functions [14]

Standard series and asymptotic decompositions are used.

The following presentation is used for not large values of the argument:

$$J_\nu(x)=\frac{\left(\frac{1}{2}x\right)^\nu}{\Gamma(\nu+1)}{}_0F_1\left(;\nu+1;-\frac{1}{4}x^2\right).$$

For big values of the argument asymptotic presentation is used

$$J_\nu(x)\sim\left(\frac{2}{\pi x}\right)^{\frac{1}{2}}\left[\cos\omega\sum_{k=0}^{\infty}(-1)^k\frac{a_{2k}(\nu)}{x^{2k}}-\sin\omega\sum(-1)^k\frac{a_{2k+1}(\nu)}{x^{2k+1}}\right],$$

where:

$$a_0(\nu)=1,$$

$$a_k(\nu)=\frac{\left(4\nu^2-1^2\right)\left(4\nu^2-3^2\right)\ldots\left(4\nu^2-(2k-1)^2\right)}{k!8^k},\quad k\geq 1,$$

$$\omega=x-\left(\frac{2\nu+1}{4}\right)\pi.$$

Bessel's functions of second kind with a non integer index are expressed through those of first kind by

$$Y_\nu(x)=\frac{J_\nu(x)\cos(\nu\pi)-J_{-\nu}(x)}{\sin(\nu\pi)}.$$

The presentation for an integer index is

$$Y_n(x)\ =-\frac{\left(\frac{1}{2}x\right)^{-n}}{\pi}\sum_{k=0}^{n-1}\frac{(n-k-1)!}{k!}\left(\frac{1}{2}x^2\right)^k+\frac{2}{\pi}\ln\left(\frac{x}{2}\right)J_n(x)-$$

$$-\frac{\left(\frac{1}{2}x\right)^n}{\pi}\sum_{k=0}^{\infty}[\psi(k+1)+\psi(n+k+1)]\frac{\left(-\frac{1}{2}x^2\right)^k}{k!(n+k)!},$$

where ψ is the logarithmic derivative of the gamma function. For positive integer values

$$\psi(1) = -\gamma,$$
$$\psi(n) = -\gamma + \sum_{k=1}^{n-1} \frac{1}{k}, \quad k \geq 2.$$

Here γ is Euler's constant.

The asymptotic presentation is

$$Y_\nu(x) \sim \left(\frac{2}{\pi x}\right)^{\frac{1}{2}} \left[\sin(\omega) \sum_{k=0}^{\infty} (-1)^k \frac{a_{2k}(\nu)}{x^{2k}} + \cos(\omega) \sum (-1)^k \frac{a_{2k+1}(\nu)}{x^{2k+1}} \right],$$

the denotations being the same as above.

For modified Bessel's *I* functions, for big and small argument respectively:

$$I_\nu(x) = \frac{\left(\frac{1}{2}x\right)^\nu}{\Gamma(\nu+1)} {}_0F_1\left(;\nu+1;\frac{1}{4}x^2\right),$$
$$I_\nu(x) \sim \frac{e^x}{\sqrt{2\pi x}} \sum_{n=0}^{\infty} (-1)^n \frac{a_{n\backslash}(\nu)}{x^n},$$

with the previous denotations for the asymptotics.

For the modified Bessel's *K* functions, for a non-integer index

$$K_\nu(x) = \frac{1}{2}\pi \frac{I_{-\nu}(x) - I_\nu(x)}{\sin(\nu\pi)}$$

and for an integer index

$$K_n(x) = \frac{1}{2}\left(\frac{x}{2}\right)^{-n} \sum_{k=0}^{n-1} \frac{(n-k-1)!}{k!} \left(\frac{1}{2}x^2\right)^k +$$
$$+ (-1)^{n+1} \ln\left(\frac{x}{2}\right) I_n(x) +$$
$$+ (-1)^n \left(\frac{x}{2}\right)^n \sum_{k=0}^{\infty} [\psi(k+1) + \psi(n+k+1)] \frac{\left(\frac{1}{2}x^2\right)^k}{k!(n+k)!};$$

and the asymptotic presentation is

$$K_\nu(x) \sim \sqrt{\frac{\pi}{2x}} e^{-x} \sum_{n=0}^{\infty} \frac{a_n(\nu)}{x^n},$$ with the previous designations.

Asymptotic presentation for big index values is not included for the moment.

For Airy's functions we use expressing through Bessel's functions:

For $x = 0$

$$A_i(0)=\frac{1}{3^{\frac{2}{3}}\Gamma\left(\frac{2}{3}\right)},\quad B_i(0)=\frac{1}{3^{\frac{1}{6}}\Gamma\left(\frac{2}{3}\right)};$$

for $x > 0$

$$A_i(x)=\frac{1}{\pi}\sqrt{\frac{x}{3}}K_{\frac{1}{3}}\left(\frac{2}{3}x^{\frac{3}{2}}\right),\quad B_i(x)=\sqrt{\frac{x}{3}}\left[I_{\frac{1}{3}}\left(\frac{2}{3}x^{\frac{3}{2}}\right)+I_{-\frac{1}{3}}\left(\frac{2}{3}x^{\frac{3}{2}}\right)\right];$$

for $x < 0$

$$A_i(-x)=\frac{\sqrt{x}}{3}\left[J_{\frac{1}{3}}\left(\frac{2}{3}x^{\frac{3}{2}}\right)+J_{-\frac{1}{3}}\left(\frac{2}{3}x^{\frac{3}{2}}\right)\right],\ B_i(-x)=\sqrt{\frac{x}{3}}\left[J_{-\frac{1}{3}}\left(\frac{2}{3}x^{\frac{3}{2}}\right)-J_{\frac{1}{3}}\left(\frac{2}{3}x^{\frac{3}{2}}\right)\right].$$

2.7. Riemann's Zeta Function [15-17]

Algorithm 3 of Borwein's publication[10] is used.

Riemann's ζ -function is an analytic continuation

$$\zeta(s)=\sum_{n=1}^{\infty}\frac{1}{n^s},\ \mathrm{Re}(s)>1=\frac{1}{\left(1-2^{1-s}\right)}\sum_{n=1}^{\infty}\frac{(-1)^{n+1}}{n^s},\ \mathrm{Re}(s)>0\,.$$

The algorithm mentioned in [10] may be outlined as follows:

Let's define

$$e_j=(-1)^j\left[\sum_{k=0}^{j-n}\frac{n!}{k!(n-k)!}-2^n\right],$$

where the empty sum is considered zero. Then

$$\zeta(s)=\frac{-1}{2^n\left(1-2^{1-s}\right)}\sum_{j=0}^{2n-1}\frac{e_j}{(j+1)^s}+\gamma_n(s),$$

where for Re(s) > 0

$$|\gamma_n(s)|\le\frac{1}{8^n}\frac{\left(1+\left|\frac{\mathrm{Im}\,s}{\mathrm{Re}\,s}\right|\right)e^{\frac{|\mathrm{Im}\,s|\pi}{2}}}{\left|1-2^{1-s}\right|},$$

and for Re(s) in [–(*n*–1), 0)

$$|\gamma_n(s)|\le\frac{4^{|\mathrm{Re}\,s|}}{8^n\left|1-2^{1-s}\right|\left|\Gamma(s)\right|}\,.$$

The residual member evaluation is easier to use in the case of a real argument.

2.8. Full elliptic integrals of first and second kind [18]

Arithmetic and geometric mean

$$K(k)=\int_0^{\frac{\pi}{2}}\frac{d\theta}{\sqrt{1-k^2\sin^2\theta}},\quad E(k)=\int_0^{\frac{\pi}{2}}\sqrt{1-k^2\sin^2\theta}\,d\theta.$$

Let a_0 and g_0 be positive numbers. We define

$$a_{n+1}=\frac{a_n+g_n}{2},\quad g_{n+1}=\sqrt{a_n g_n},\quad n=0,1,2,\ldots$$

Series $\{a_n\}$ and $\{g_n\}$ have a common limit $\mathrm{AGM}(a_0, g_0)$ (arithmetic-geometric mean). If we define $c_n=\sqrt{a_n^{\,2}-g_n^{\,2}}$,

we have

$$K(k)=\frac{\pi}{2\mathrm{AGM}\left(1,\sqrt{1-k^2}\right)},\quad E(k)=K(k)\left[a_1^{\,2}-\sum_{n=2}^{\infty}2^{n-1}c_n^{\,2}\right].$$

The asymptotic behavior near the particular point *k* = 1 is not yet realized.

2.9. Bernoulli numbers

A method proposed by M c G o w n [11]

The algorithm described in [1] may be reduced essentially to the following.

Supposing that $m\ge 2$ is even consecutively compute:

Step 1. $$K=\frac{2m!}{(2\pi)^m},$$

Step 2. $$d=\prod_{p-1\mid m}p,$$

Step 3. $$N=\left\lceil (Kd)^{1/(m-1)}\right\rceil,$$

Step 4. $$z=\prod_{p\le N}(1-p)^{-1},$$

Step 5. $$a=(-1)^{m/2+1}\lceil dKz\rceil,$$

Step 6. $$B_m=\frac{a}{d}.$$

The product in Step 2 is for all prime numbers *p*, for which *p* – 1 is divisible by *m*. In Step 4, respectively, the product is for the prime numbers less than or equal to *N*. The value of *K* should be computed precisely enough at the first step so that the calculation in Step 5 rendered the result wanted. For a value of *N* any integer greater than or equal to the one defined at Step 3 may be taken.

Let's comment in brief why this works. This algorithm uses the following results.

Firstly,

(1) $$\zeta(s)=\prod_p(1-p^{-s})^{-1}.$$

Secondly, for each integer $m \geq 1$

(2) $$2\zeta(2m) = \frac{(-1)^{m+1}(2\pi)^{2m}}{(2m)!} B_{2m},$$

hence we receive for each even integer $m \geq 4$,

$$|B_m| = \frac{2m!}{(2\pi)^m} \zeta(m).$$

Besides these two results proven by Euler the von Staudt and Clausen theorem is known (rediscovered by Ramanujan, see respective topics in [20] and [22], the prime source [12] from 1840 is hard to access), which describes the B_m devisor, presented as division of mutually simple integers through the devisors of *m*. This is the product at Step 2 of the algorithm. In Step 1 of the algorithm *K* is defined so that $|B_m| = K\zeta(m)$. Using (1) we can approximate $\zeta(m)$ from below to arbitrary precision. If a number *z* is calculated for which $0 \leq \zeta(m) - z < (Kd)^{-1}$, then $0 \leq |B_m| - zK < d^{-1}$ and therefore $0 \leq |a| - zKd < 1$. $|a|$ denotes the numerator of $|B_m|$. It follows from this that $|a| = \lceil zKd \rceil$ and hence $a = (-1)^{m/2+1} \lceil zKd \rceil$. The real calculation of *z* remains, which is reduced to: with given $s > 1$ and $\varepsilon > 0$ find a real integer *N* for which at Step 4 of the algorithm it is guaranteed that $0 \leq \zeta(s) - z < \varepsilon$. We already have $0 \leq \zeta(s) - z$, as *z* is approximation from below. In could be besides checked (prime numbers are in the product) that $\sum_{n \leq N} n^{-s} \leq \prod_{p \leq N} (1 - p^{-s})^{-1}$, consequently

$$\zeta(s) - z \leq \sum_{n=N+1}^{\infty} n^{-s} \leq \int_N^{\infty} x^{-s} dx = \frac{1}{(s-1) N^{s-1}}.$$

If we choose $N > \varepsilon^{-1/(s-1)}$, then $\frac{1}{(s-1)N^{s-1}} \leq \frac{1}{N^{s-1}} < \varepsilon$, which results in $\zeta(s) - z < \varepsilon$. For our purposes we have $s = m$ and $\varepsilon = (Kd)^{-1}$. It is therefore enough to select $N > (Kd)^{1/(m-1)}$.

2.10. Integral sine and cosine

Standard series and asymptotic decompositions are used.

$$S_i(x) = \int_0^x \frac{\sin t}{t} dt, \quad C_i(x) = -\int_x^{\infty} \frac{\cos t}{t} dt.$$

Presentation by series

$$S_i(x) = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{(2n+1)!(2n+1)}, \; C_i(x) = \gamma + \ln(x) + \sum_{n=1}^{\infty} \frac{(-1)^n x^{2n}}{(2n)!(2n)}.$$

The asymptotic presentation for big values of *x* is

$$S_i(x)=\frac{\pi}{2}-f(x)\cos x-g(x)\sin x\,,\ C_i(x)=f(x)\sin x-g(x)\cos x\,,$$

where

$$f(x)\sim\frac{1}{x}\left(1-\frac{2!}{x^2}+\frac{4!}{x^4}-\frac{6!}{x^6}+\cdots\right),\ g(x)\sim\frac{1}{x^2}\left(1-\frac{3!}{x^2}+\frac{5!}{x^4}-\frac{7!}{x^6}+\cdots\right).$$

3. Testing and calculator

The testing process, as it is well known, is the most time and labor-consuming operation for a computer program. This of course is valid to even greater degree when calculating special functions. It includes several stages and is not still completed. The main checks are for previously known values of a given function with exactly defined argument/s. An even better test is the possibility of comparing a given result when it could be expressed by different functions, thus using different methods. For checking with big arguments values where asymptotic presentations are used, this is crucial. Adding more functions assists this type of tests. There are in the net a lot of on line calculators for different types of special functions which are convenient for initial adjustment. But most of them are of limited precision. A natural opportunity is the parallel using of another program, using e.g. mpmath in Python. It is anyway convenient to have an interactive program for testing. For this purpose a prototype of a calculator is realized. It allows dynamic change of the precision used. The display field adds automatically a vertical slider for scrolling, if needed, when required by the current computing precision. The format of the numbers displayed is automatically changed depending whether the result needs exponential format. Appropriate rounding is carried out. Indication is available for a pending argument for functions with more than one argument. Digits, decimal point and arithmetic operations may be also entered from the keyboard. There is not yet overflow check. The problem is specific for the MPIR library also where the floating point numbers exponent is fixed. For 32-bit systems from 68 719 476 768 to 68 719 476 736, depending on the machine word but not equal to it. For a 64-bit system, for which the present special functions library is being developed this range is larger and this brought to underestimation of the problem initially. This overflow check will be probably executed at the level of functions. Some change will be probably necessary at that level of the special functions reaction with regard to non valid argument. In this realization when the argument is non valid they issue a message and return the input and do not cause an exception just for convenience at testing in interactive environment. The adequate approach for the function behavior in an independent environment is to be considered. There the responsibility of entering an admissible argument does not lie on the calling program. The alteration of the current model will require additional efforts for the program-user and namely processing of specific exceptions. The calculator for the moment is in a form that allows easy adding of new functions. The project presumes the 'calculator' to be transformed in a source of references for the used special functions and graphical representations.

An example follows with a calculator with 200 digits precision:

$$K\left(\frac{1}{\sqrt{2}}\right)=\frac{1}{4\sqrt{\pi}}\Gamma\left(\frac{1}{4}\right)^2.$$

Consecutive entering: 4, 1/x, $\Gamma(x)$, x^2, MS, π, $\sqrt{}$, 1/x, /, 4, =, *, MR, =, yields the right hand side of the equation (Fig. 1).

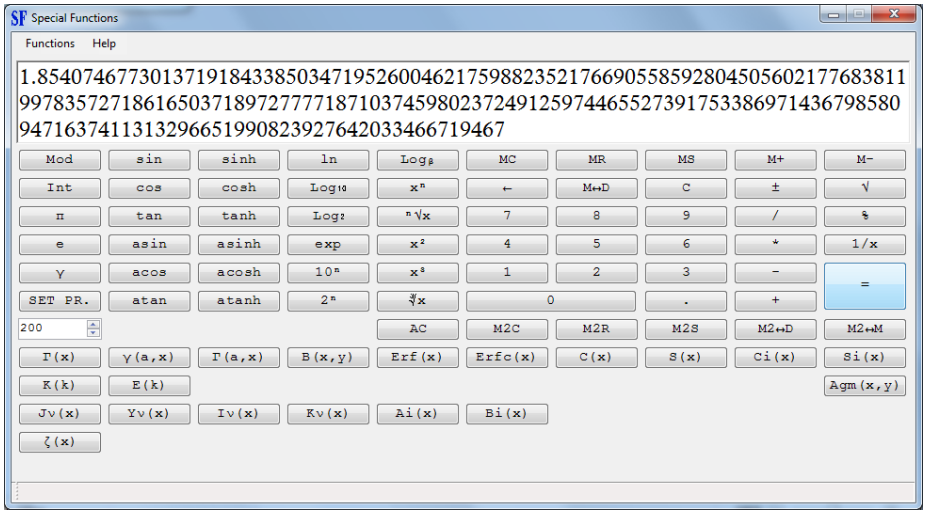

Fig. 1

It may be stored by MS and after entering 2, $\sqrt{}$, 1/*x*, *K*(*x*), the left one is received. No illustration is given since the result is the same.

Additional facilities are added for easy testing when the equivalent expression is more complicated: storing in a second register (M2S) and $\mathrm{M} \leftrightarrow D$, and $\mathrm{M2} \leftrightarrow D$, which changes the places of the last received and last stored in the respective auxiliary register results.

The project ambitions as a source of references are shown in Fig. 2.

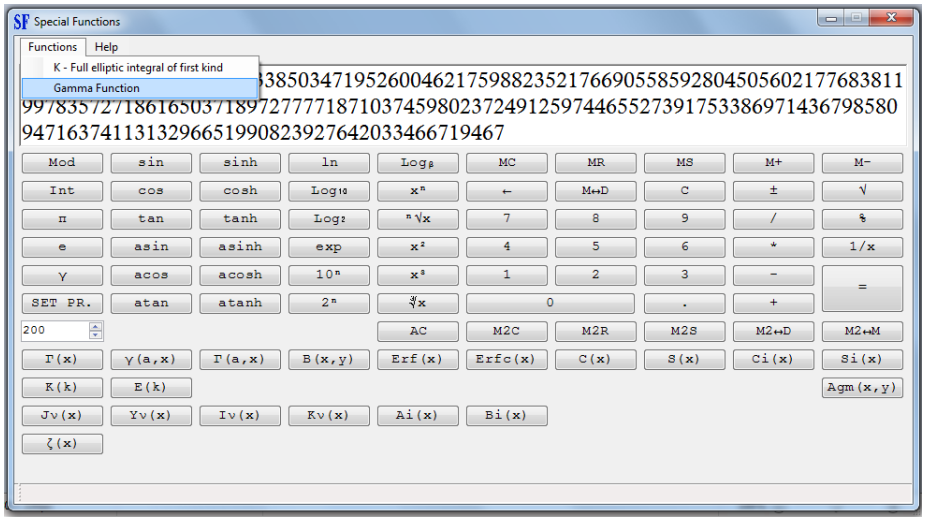

Fig. 2

After choosing the reference the results shown in Fig. 3 are obtained.

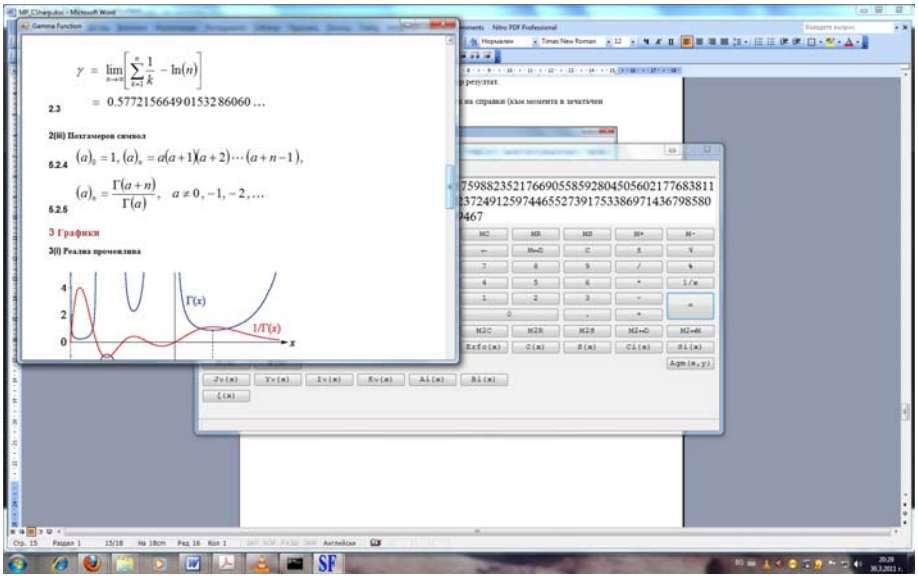

Fig. 3

4. Conclusion

The main problem in the realization of a similar project is in its range – a highly labor-consuming job. As mentioned in the introduction the purposes are in essence carried out to a great degree in other specialized products or packages in other programming languages environment. Leaving away the issue that the greater part of them are paid commercial products, and those which are not require more programming by the user, e.g. for the graphical interface, we reckon that the possibility of arbitrary precision computations combined with the other exclusively rich features of .NET Framework is worth the efforts. The completed environment for arbitrary precision computations, including also methods of the numerical calculus, provides the base for an easy graphical interface which is the next logical step of the project. Probably new types will appear in the future in .NET Framework as continuation of BigInteger but this does not seem happen soon. A calculator of arbitrary precision can by found in web with a limited set of functions, which is based on BigInteger *only*. This approach is not a good perspective for the present project. The type BigInteger is not designed for this purpose and real numbers calculations based on it is not efficient enough. A more straightforward approach is writing an own class representing floating point numbers with arbitrary precision. But this moves the aims a step aback without mentioning the technical difficulties - writing in C style with mandatory modifiers ‘unsafe’ that returns to the beginning. So the main problem remains - time and labor consumption. So for the full realization of the project besides adherence to the theme and some financial support confidence in its utility and value is necessary.

References

1. T. E. Simos (Ed.). Recent Advances in Computational and Applied Mathematics. Chapter 4. Basic Methods for Computing Special Functions by Amparo Gil, Javier Segura, and Nico M. Temme). – European Academy of Sciences, Springer, 2011.
2. G i l, A., J. S e g u r a, N. M. T e m m e. Numerical Methods for Special Functions. SIAM, 2007.
3. O l v e r, F. W. J. Asymptotics and Special Functions. Academic Press, 1974.
4. W. Frank, J. Olver (Eds.). NIST Handbook of Mathematical Functions, National Institute of Standards and Technology. Cambridge University Press, 2010.
5. M. Abramowitz, I. A. Stegun (Eds.). Handbook of Mathematical Functions with Formulas, Graphs and Mathematical Tables. National Bureau of Standards and Applied Mathematics Series. Vol. **55**, 1964.
6. B a t e m a n, H., A. E r d é l y i. Higher Transcendental Functions. Vol. **1**, **2**, **3**, 1953-1955.
7. B r e n t, R. P., . E . M. M c M i l l a n. Some New Algorithms for High-Precision Computation of Euler's Constant. – Mathematics of Computation, Vol. **34**, January 1980, No 149, 305-312.
8. B r e n t, R. P. Fast Multi-Precision Evaluation of Elementary Functions. – Journal of the Associations for Computing Machinery, Vol. **23**, April 1976, No 2, 242-251.
9. S p o u g e, J. L. Computation of the Gamma, Digamma and Trigamma Functions. – SIAM Journal on Numerical Analysis, **31** , 1994, No 3, 931-944.
10. B o r w e i n, P. An Efficient Algorithm for the Riemann Zeta Function. – In: Proc. of Canadian Mathematical Society Conference, 1991.
11. M c G o w n, K. J. Computing Bernoulli Numbers Quickly, December 8, 2005.
12. S t a u d t, K. G. C. von. Beweis eines Lehrsatzes, die Bernoullischen Zahlen betreffend. – J. Reine Angew. Math., **21**, 1840, 372-374.
13. A r t i n, E. The Gamma Function, Holt, Rinehart and Winston, 1964.
14. W a t s o n, G. N. A Treatise on the Theory of Bessel Functions. Second Edition. Cambridge University Press, 1945.
15. T i t c h m a r s h, E. C. The Theory of the Riemann Zeta-Function. Second Edition. Reprinted, Oxford, Clarendon Press, 1988.
16. E d w a r d s, H. M. Riemann's Zeta Function. Academic Press, 1974.
17. K a r a t s u b a, A. A., S. M. V o r o n i n. The Riemann Zeta-Function. Walter and Gruyter, 1992.
18. B y r d, P. F., M. D. F r i e d m a n. Handbook of Elliptic Integrals for Engineers and Scientists. Second Edition. Revised, Springer-Verlag, 1971.
19. MPIR site.

    http://www.mpir.org/
20. Wolfram Mathworld site.

    http://mathworld.wolfram.com/
21. Wolfram Functions site.

    http://functions.wolfram.com/
22. Wikipedia site,

    http://en.wikipedia.org/