БЪЛГАРСКА АКАДЕМИЯ НА НАУКИТЕ • BULGARIAN ACADEMY OF SCIENCES

ПРОБЛЕМИ НА ТЕХНИЧЕСКАТА КИБЕРНЕТИКА И РОБОТИКАТА, **66**
PROBLEMS OF ENGINEERING CYBERNETICS AND ROBOTICS, **66**

София • 2015 • Sofia

On the Fast Computing of Some Constants. High Precision Computation with .NET Framework C# and X-MPIR

Velichko Dzhambov

Institute of Information and Communication Technologies, 1113 Sofia
Emails: jambov@abv.bg

***Abstract**: This paper concerns implementing some methods designed for computation of various mathematical constants with high precision in specific environment, namely .NET Framework. The work is part of a series tracing the progress of creating tools for high precision computations in this environment and may be considered as continuation, in this direction, of (Dzhambov, 2011), (Dzhambov, 2014a), (Dzhambov, 2014b), that includes special function calculations with arbitrary precision and some additional numerical methods. The results are illustrated with the help of application, using the current state-of-art library being created for implementation of numerical methods of arbitrary precision in a given environment.*

***Keywords:** High precision computation, parallel computations, mathematical constants, computational mathematics, .NET Framework, X-MPIR*

1 Introduction

Arbitrary precision computations are not a self-purpose. They are related to receiving precise values where often different mathematical constants are needed.. In (Dzhambov, 2011) an announcement is made for creating of high precision computation library in the environment of .NET Framework. The application SFCALC was represented there, illustrating a part of the potentialities of the library being created. Some extensions of this library are added further concerning

computing of definite integrals (Dzhambov, 2014a), solving of systems of nonlinear ordinary differential equations, and root finding of nonlinear algebraic .equation (real roots), described (Dzhambov, 2014b).

They are at least two reasons that make mathematical constants subject of particular interest in high precision computing. Some of these, like $\pi, e, \gamma, \ln 2$, are frequently used in process of computing transcendental mathematical functions (special and elementary). Additionally high precision computing of mathematical constants is preliminary condition if we want to implement some algorithm for integer relation detection (for example PSQL). The special application MPConsts presented here envelops several methods devoted to effective high precision computing of some mathematical constants.

One of the purposes is to demonstrate that useful and mutually complementing computational tools for solving non-trivial problems with high precision computations may be implemented in a concrete environment which is wide spread for personal computers but it seems to be underestimated by the software developers for scientific applications. There are reasons of course for this, but in our opinion, there are advantages also: comparatively easy integration in various functionalities such as visualization and interactivity in one and the same application if, of course, the necessary methods for solving the concrete problems are available which is the purpose of the library being created. We of course keep in mind that the creation of a complete system covering a great range of tools for solving of various mathematical models is a task too ambitious due at least to the huge amount of the work required. But the implementation of some basic tools makes it possible for the interested investigator to create own applications in the given environment, exactly matching to his concrete interests. To viz. create and not be a user of a ready system. Another purpose which hope to achieve is to provide such a possibility. The general structure and some extensions of the library being created are already described in previous papers and we do not repeat them. In all cases there is also conclusion, that further improvements are possible, including using the multi-core architecture of the contemporary processors, allowing parallel computations. Here is the first step of this program.

2 Speedup resources

Maximal effectiveness is reached if we have at our disposal parallel algorithm using almost everywhere integer operations. That is not always possible of course, but where it is, the results impress. The base idea is to use binary splitting for particular type of power series (hypergeometric, where all coefficients are rational numbers). The idea is quite old (Brent, 1976). (Haible, 1998) is one of the best explanations. Chapter 4 of (Brent, 2011) is also very useful. (Yee, 2011) is a wide panorama of using multicore processor's architecture. The time complexity of binary splitting algorithms is $O(M(d)(\log d)^{\alpha})$ for some $\alpha > 1$, but the effectiveness is due to the fact, that as much as possible multiplications are pushed to the region where multiplication becomes efficient. So in the complexity evaluation the factor $M(d)$ is

important here. Perhaps the main advantage is the possibility of parallel implementation.

Because the speed is an important goal, we were obliged to make changes in original files being part of the procedure generating dynamic link libraries (XMPIR, site), so they correspond to the numerous signature changes in the last version of MPIR (2.7.0, 29.06.2015, (MPIR, site)). Of course we also profited by the new possibility to adjust the dll-file according to the type of the processor.

MPConsts use parallel computations (binary splitting for suitable series) for some constants, namely $\pi, e, \gamma, \ln 2, G, \zeta(3)$, where the last two are respectively the Catalan's constant $G = \beta(2) = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)^2}$ and the Apéry's constant $\zeta(3) = \sum_{n=1}^{\infty} \frac{1}{n^3}$. However for some constants they own particular method are used. These methods are briefly described in place below.

3 Comparative results for some constants

The data presented in tables below summarize time elapsed to compute some constants with different precision (in decimal digits). For the comparison the program γ-crunher (Yee, 2015) is chosen, because this program is undisputed leader at the moment. Some data are presented for PIFast too ((PiFast, site), quite old, but the best 10 years ago). It must be noted that the program γ-crunher uses many optimizations (even determination of suitable sizes of chunks to be passed to parallel processing, avoiding this way time wastage for synchronization). This program is designed to be extremely effective for vey high precision (including if necessary extending tools like external data storage). An excellent overview of methods used is previously mentioned paper (Yee, 2011). There is an explanation too why for a (relatively) modest precisions (several hundreds or thousands digits) the program is not so spectacular. Unfortunately the code is not open. Because some of our goals (in many cases several thousands digits suffice - some constant identification with PSLQ, for example) are reached without extremely high precision, we find that results obtained with MPConsts are good enough. All tests are performed on the author's laptop whose characteristics are: processor Intel(R) Core(TM) i7-3610QM CPU @ 2.30GHz (4 physical and 8 logical processors).16 GB RAM, 64-bit operating system Windows 7 Enterprise (Microsoft Windows NT 6.1.7601 Service Pack 1).

10^6 digits	γ-crunher	MPConsts	PiFast
e	0.112	0.245	0.39
π	0.230	0.522	1.04
ln(2)	0.506	1.453	
Apéry	0.667	2.859	
Lemniscate	1.353	2.845	
Catalan	2.891	9.799	
γ	4.081	48.642	

Table 1. Timings for computing several constants with 1,000,000 decimal digits.

10^5 digits	γ-crunher	MPConsts	PiFast
e	0.030	0.039	0.08
π	0.061	0.050	0.10
ln(2)	0.088	0.134	
Apéry	0.078	0.207	
Lemniscate	0.126	0.186	
Catalan	0.228	0.612	
γ	0.302	2.510	

Table 2. Timings for computing several constants with 100,000 decimal digits.

10^4 digits	γ-crunher	MPConsts	PiFast
e	0.021	0.020	0.03
π	0.041	0.024	0.03
ln(2)	0.059	0.026	
Apéry	0.024	0.028	
Lemniscate	0.041	0.034	
Catalan	0.040	0.054	
γ	0.092	0.182	

Table 3. Timings for computing several constants with 10,000 decimal digits.

10^3 digits	γ-crunher	MPConsts
e	0.021	0.020
π	0.033	0.023
ln(2)	0.050	0.019
Apéry	0.033	0.018
Lemniscate	0.039	0.029
Catalan	0.025	0.021
γ	0.086	0.044

Table 4. Timings for computing several constants with 1,000,000 decimal digits.

All data are in seconds. Lemniscate in the tables above is the name of the constant $2\int_0^1 \frac{dx}{\sqrt{1-x^4}} = \left[\Gamma\left(1/4\right)\right]^2 \Big/ \left(2\sqrt{2\pi}\right)$. This is the only constant here, for which another method is used (arithmetic geometric mean, more specifically $\frac{\pi}{AGM(1,\sqrt{2})}$). In PIFast less than 10,000 digits are not allowed.

4 Khinchin's constant

For a given real number x with regular continued fraction representation:

$$x = q_0 + \frac{1|}{|q_1} + \frac{1|}{|q_2} + \cdots,\ q_0 \in \mathbb{Z},\ q_n \in \mathbb{Z}^+,\ n > 0 \qquad (1)$$

let $K(x)$ denote the limit of the geometric mean $\{q_k\}_{k=1}^{n}$, i.e.

$$K(x) = \lim_{n\to\infty}\left(\prod_{k=1}^{n} q_k\right)^{\frac{1}{n}} \qquad (2)$$

Khinchin proved that this limit is equal for almost all real numbers, i.e. K(x)=K for a set of real numbers with Lebesgue measure 1. More specifically:

$$K = \prod_{n=1}^{\infty}\left(1+\frac{1}{n(n+2)}\right)^{\frac{\ln n}{\ln 2}} \qquad (3)$$

The method of computation we use is based on [13]. This method is almost straightforward application of theorem 3 in the same paper:

$$\begin{aligned}\ln(K)\ln(2) &= \sum_{s=1}^{\infty}\zeta(2s,N)\frac{A_s}{s} \\ &- \sum_{k=2}^{\infty}\ln\left(1-\frac{1}{k}\right)\ln\left(1+\frac{1}{k}\right)\end{aligned} \qquad (4)$$

where $\zeta(s,N) = \sum_{n=1}^{\infty}\frac{1}{(n+N)^s}$ is the Hurwitz function and for non negative integer N its value is $\zeta(s) - \sum_{n=1}^{N}\frac{1}{n^s}$, $A_s = \sum_{m=1}^{2s-1}\frac{(-1)^{m-1}}{m}$.

To compute values of the ζ-function at even integer the formula:

$$\begin{aligned}\zeta(2n) &= (-1)^{n-1}\frac{2^{2n}B_{2n}}{2(2n)!}\pi^{2n} \\ &= \frac{T_n}{2^{2n+1}(1-2^{-2n})(2n-1)!}\pi^{2n}\end{aligned} \qquad (5)$$

is used, where B2n are the Bernoulli nimbers and T_n are the tangent numbers. We use TangentNumbers algorithm in (Brent, 2011).

In (10000 digits of Khinchin's constant, site) a result of 10025 decimal digits is reported, obtained with Python (mpmath+gmpy) for 11 minutes (machine used and version of the software are not mentioned). MPConsts takes on the author's laptop 33.5 seconds. In Gourdon, X., 2013 a result can be found (1998 ,Xavier Gourdon) with 110,000 digits. This result is still a record ((Numbers, constants and computation, site), Last update: August 12 2010). The computer used is sgi R10000. Time of computing: 22 hours and 23 minutes. With MPConsts on the author's laptop the time is 5 hours and 8 minutes.

This old record can be surpassed, of course, but this is not an objective. Moreover, our implementation can be optimized obviously. The ratio of the times is about 1/4.4 and the ratio of the processors speed is 9.2.

5 Landau-Ramanujan Constant

Let $B(x)$ denote the number of positive integers les than x, which can be presented as a sum of two perfect squares. In number theory a result exists that reads $B(x) = O\left(\frac{x}{\sqrt{\ln(x)}}\right)$. The limit $\lambda = \lim_{n\to\infty} \frac{B(x)\sqrt{\ln(x)}}{x}$ is called Landau-Ramanujan constant. Effective computing of this constant is possible if some analytical representation exists. Fortunately one is found by Shanks [17]:

$$\lambda = \frac{1}{\sqrt{2}} \prod_{n=1}^{\infty} \left[\left(1 - \frac{1}{2^{2^n}}\right) \frac{\zeta\left(2^n\right)}{\beta\left(2^n\right)} \right]^{\frac{1}{2^{n+1}}} \qquad (6)$$

Here besides the Riemann ζ-function, Dirichlet β-функцията $\beta(s) = \sum_{k=0}^{\infty} \frac{(-1)^k}{(2k+1)^s}$ is present. One elegant deduction can be found in (Flajolet, 1996). The proposed in MPConsts implementation is based of the idea to use as far as possible values of the ζ-function at even integers. For that purpose the following equations can be used (Lima, 2012), (Srivastava, 2012)

$$\left.\begin{aligned} \beta(2n) &= (-1)^{n+1} \frac{(\pi/2)^{2n-1}}{(2n-1)!} \ln 2 \\ &- (-1)^{n+1} \sum_{k=1}^{n-1} (-1)^k \frac{(\pi/2)^{2n-2k-1}}{(2n-2k-1)!} \left(1 - \frac{1}{2^{2k}}\right) \zeta(2k+1) \\ &+ (-1)^n \frac{\pi^{2n-1}}{2^{2n}} \sum_{k=0}^{\infty} \frac{\zeta(2k+2)}{2k(2k+1)\ldots(2k+2n-1)} \left(\frac{1}{2^{2k}} - \frac{1}{2^{4k}}\right) \end{aligned}\right\} (7) \quad (1)$$

$$\zeta(2n+1) = (-1)^n \frac{2(2\pi)^{2n}}{(2n-1)2^{2n}+1} \left[\sum_{k=1}^{n-1} \frac{(-1)^{k-1} k}{(2n-2k+1)!} \frac{\zeta(2k+1)}{\pi^{2k}} + \sum_{k=0}^{\infty} \frac{(2k)!}{(2n+2k+1)!} \frac{\zeta(2k)}{2^{2k}} \right] \qquad (8)$$

We also can take advantage of the representation (Flajolet, 1996)

$$\left(1 - 2^{-s}\right) \frac{\zeta(s)}{\beta(s)} = \prod_r \frac{1 + r^{-s}}{1 - r^{-s}} \qquad (9)$$

where r range over primes $\equiv 3 \bmod 4$,i.e.

$$\lambda\sqrt{2} = \prod_{n=1}^{\infty} \left[\prod_r \frac{1 + r^{-2^n}}{1 - r^{-2^n}} \right]^{\frac{1}{2^{n+1}}} \qquad (10)$$

$$\begin{aligned}\ln(\lambda\sqrt{2}) &= \sum_{n=1}^{\infty}\frac{1}{2^{n+1}}\ln\prod_{r}\frac{1+r^{-2^n}}{1-r^{-2^n}}\\ &= \sum_{n=1}^{\infty}\frac{1}{2^{n+1}}\sum_{r}\ln\left(\frac{1+r^{-2^n}}{1-r^{-2^n}}\right)=\\ &= \sum_{n=1}^{\infty}\sum_{r}\frac{1}{2^{n+1}}\ln\left(\frac{1+r^{-2^n}}{1-r^{-2^n}}\right)\\ &= \sum_{r}\sum_{n=1}^{\infty}\frac{1}{2^{n+1}}\ln\left(\frac{1+r^{-2^n}}{1-r^{-2^n}}\right)\end{aligned} \qquad (11)$$

For r^{2^n} large enough we can compute directly an inner sum.
In our implementation (can be optimized, surely) MPConsts compute 4000 digits for 44 minutes, 2000 digits for 5 minutes and 38 seconds, 1000 digits for 37 seconds, 200 digits for 0.6 seconds.

6 Example of constants, computed as zeros of function: Laplace limit, Ramanujan–Soldner constant

The Laplace limit is connected to a problem of celestial mechanics. For a body moving in an ellipse with eccentricity ε Kepler's equation $M = E - \varepsilon\sin E$ relates the mean anomaly M with the eccentric anomaly E. The power series in ε of the solution (the equation is not solvable in elementary functions) has a radius of convergence, called Laplace limit. It can be proved that this limit is a solution of the equation $\frac{xe^{\sqrt{1+x^2}}}{1+\sqrt{1+x^2}} - 1 = 0$. MPConsts uses iterative adjustment of a hard coded good initial approximation. The solution takes respectively 15.2 seconds for 100,000 decimal digits, 12 minutes and 32 seconds for 1,000,000 decimal digits.
The Ramanujan–Soldner constant constant is defined as the unique positive zero of the integral logarithm $\mathrm{li}(x) = \int_0^x \frac{\mathrm{dt}}{\ln(t)}$. The same approach give us 100,000 decimal digits for about 4 minutes and 28 seconds.

7 Conclusions

The paper presents a part of the currently done implementation of a library and tools for computing with arbitrary accuracy in environment, not typical for this purpose, namely .NET Framework. The combination of well selected methods and the excellent possibilities for integration of different functionalities in this environment worth the efforts, and enable the achievement of non-trivial results even on a home computer. The application, herein presented, though useful, is mainly illustration of the library possibilities including the use of the multi-core

architecture of the modern processors, executing parallel calculations at some stages of the algorithms used and could be further improved in different aspects.

Acknowledgements

The research work reported in the paper is partially supported by the project AComIn “Advanced Computing for Innovation”, Grant 316087, funded by FP7 Capacity Programme (Research Potential of Convergence Regions).

References

1. Dzhambov, V., S. Drangajov, 2011, *Computing of Special Functions with Arbitrary Precision in the Environment of .NET Framework*, Cybernetics and Information Technologies, Volume 11, No 2, 32-45.
2. Dzhambov, V., 2014a, *High Precision Computing of Definite Integrals with .NET Framework C# and X-MPIR*, Cybernetics and Information Technologies, Volume 14, No 1, 172-182.

Методов еффективного вычисления некоторых констант на PC"

Величко Джамбов

Институт информационных и коммуникационных технологий, 1113 София

Р е з ю м е

Описна методика вычислений произвольной точности некоторых растпространеных и более специальных математических констант на персональном компютере. Приводятся сравнительные результаты. Использована библиотека MPIR и язык програнирования C#."