ISSN 1995-0802, Lobachevskii Journal of Mathematics, 2019, Vol. 40, No. 10, pp. 1725–1738. © Pleiades Publishing, Ltd., 2019.

The Faster Methods for Computing Bessel Functions of the First Kind of an Integer Order with Application to Graphic Processors

D. N. Tumakov*

(Submitted by E. K. Lipachev)

Institute of Computational Mathematics and Information Technologies, Kazan Federal University, Kazan, Tatarstan, 420008 Russia

Received April 20, 2019; revised May 29, 2019; accepted May 31, 2019

Abstract—Algorithms for fast computations of the Bessel functions of an integer order with required accuracy are considered. The domain of functions is split into two intervals: $0 \le x \le 8$ and $x > 8$. For the finite interval, expansion in the Chebyshev polynomials is applied. An optimal algorithm for computing functions $J_0(x)$ and $J_1(x)$ is presented. It is shown that the sufficient number of mathematical operations equals 15 for computing the function $J_0(x)$ and 16 for computing the function $J_1(x)$ in the interval $x \le 8$ with the approximation error $O(10^{-6})$. Several algorithms for approximation of the functions $J_n(x)$ at $n > 1$ are presented. The increase in speed of computations of the Bessel functions obtained through using our in-house methods in place of the Toolkit library is evaluated. Graphs showing the improvement of performance are presented.

DOI: 10.1134/S1995080219100287

Keywords and phrases: *Bessel functions, fast methods for computing, expansion in the Chebyshev polynomials, graphic processor.*

1. INTRODUCTION

The Bessel functions are solutions of Bessel's equation. The functions appear during solution of the Laplace equation or the Helmholtz equation in cylindrical or spherical coordinates. In addition, the Bessel functions can be also encountered in other applications [1, 2]. For numerical solutions of a number of problems with required accuracy, algorithms utilizing the Bessel functions are already available. In the literature, various approaches were presented including those using expansions in series [3], recurrence relations [4], quadrature rules [5, 6] and other approximation methods [7, 8].

Power series expansions [9] and Chebyshev polynomials expansions [10] are the most frequently used methods for numerical calculations of the special functions. For determining the expansion coefficients for both the case of an explicitly defined approximation function and the case of a discrete set, there exist various approaches.

Algorithms for computing the Bessel functions in the programming languages C++ [11] and Fortran [12] are available. Besides, the library of algorithms called ALGLIB contains source codes written in several programming languages. Efforts to compute special functions with any arbitrary accuracy using the environment called .NET were described in [13].

Descriptions of the Bessel functions which are included into the interfaces in the standard Linux Standard Base were presented in [9]. Also, the CUDA library Toolkit contains the Bessel functions of the first and second kinds of an integer order. The library functions of the type "float" include j0f(x), j1f(x), jnf(n,x), y0f(x), y1f(x) and ynf(n,x) which are calculated with error 2.2×10^{-6} in the maximum norm for the floating point type. Then, the library functions of the type "double" include j0(x), j1(x),

*E-mail: dtumakov@kpfu.ru

jn(n,x), y0(x), y1(x) and yn(n,x) which are calculated with error 5×10^{-12} in the maximum norm for the double-precision floating point type.

For approximation of the functions, the domain needs to be split into intervals. Usually, the domains are split into two intervals with one interval being finite and the other one being infinite. Then, the asymptotic formula is applied to the infinite interval. For example, in [14] and [15], a power-polynomial approximation was considered for intervals $0 \le x \le 3$ and $x > 3$. To match intervals to intervals used by the Toolkit, the intervals $0 \le x \le 8$ and $x > 8$ are here adopted.

In the present work, issues related to optimization of the speed of computations of the Bessel functions with given accuracy are considered. Two types of approximations are studied. The first type is used for calculations with error $\approx 10^{-6}$ using a 4-byte floating point number and the second type is used for calculations with error $\approx 10^{-12}$ using an 8-byte double-precision floating point number.

Proofs are given for the statements concerning estimates of errors of approximation of the Bessel functions by the Chebyshev polynomials $T_n(x)$ in the interval $[-q, q]$. For the floating point type, algorithms used for computing $J_n(x)$ are presented. For computing $J_0(x)$ and $J_1(x)$, optimal algorithms allowing two-fold acceleration of computational speed versus algorithms used in the Toolkit are included. Conclusions with regards to improvement of performance in filling a matrix with values of the Bessel functions through using the video card Tesla C1060 in place of the completely loaded processor i7-920 are made.

2. APPROXIMATION OF FUNCTIONS. CHEBYSHEV POLYNOMIALS OF THE FIRST KIND

A few moments regarding approximation of the functions will be considered next. Let $\{\varphi_r(x), r = 0, 1, \ldots\}$ be the basis over interval $[\alpha, \beta]$ in a certain space $\mathbf{L}$. The error of approximation of any function $f(x)$ by a polynomial

$$L_n(x) = \sum_{r=0}^{n-1} a_r \varphi_r(x)$$

is estimated in the following manner:

$$||f(x) - L_n(x)||_{\mathbf{L}} = \left|\left| \sum_{r=n}^{\infty} a_r \varphi_r(x) \right|\right|_{\mathbf{L}}.$$

Most often, the space of continuous functions $C[\alpha, \beta]$ is chosen for the space $\mathbf{L}$. But for some applications, it is more convenient to use other spaces.

The best estimate of the approximation corresponds to polynomials constructed for an explicitly defined function. For the case of a finite interpolation polynomial being constructed using discrete points, the estimate is deteriorated (P. 68, [16]). For instance, two methods of such interpolation along with estimates for them were considered in [17].

Any continuous function $f(x)$ in the interval $[-q, +q]$ can be represented in the form of the series in the Chebyshev polynomials:

$$f(x) = \sum_{n=0}^{\infty} a_n T_n\left(\frac{x}{q}\right),$$

where

$$a_0 = \frac{1}{\pi} \int_{-1}^{+1} f(qx) T_0(x) dx, \quad a_n = \frac{2}{\pi} \int_{-1}^{+1} f(qx) T_n(x) dx, \quad n = 1, 2, \ldots.$$

The main moments concerning calculation of the Chebyshev polynomials will be considered. The Chebyshev polynomials of the first kind can be determined by using the following recurrence relation:

$$T_0(x) = 1, \quad T_1(x) = x,$$
$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x), \quad n = 1, 2, \ldots$$

The even Chebyshev polynomials can be represented as

$$T_0(x) = 1, \quad T_2(x) = 2x^2 - 1, \tag{1}$$

$$T_{n+2}(x) = (4x^2 - 2)T_n(x) - T_{n-2}(x), \quad n = 2, 4, \ldots \tag{2}$$

The recurrence relations for the odd polynomials remain the same as the following initial functions:

$$T_1(x) = x, \quad T_3(x) = 4x^3 - 3x. \tag{3}$$

3. COMPUTING THE BESSEL FUNCTION J_0

Theorem 1. *For the Bessel function of the first kind of order zero, the following representation is valid*

$$J_0(x) = 2\sum_{k=0}^{n-1}(-1)^k J_k^2\left(\frac{q}{2}\right) T_{2k}\left(\frac{x}{q}\right) + R_n(x), \quad -q \le x \le q,$$

with estimate

$$|\widetilde{R}_n| = \max_{-q\le x\le q}|R_n(x)| \le \frac{4}{\pi q e}\sum_{k=n}^{\infty}\left(\frac{qe}{4k}\right)^{2k+1}.$$

In particular, for the case $q = 8$ the following inequalities will be true: $|\widetilde{R}_9| \le 4.23 \times 10^{-6}$ and $|\widetilde{R}_{13}| \le 3.58 \times 10^{-12}$.

Proof. The function $J_0(x)$ can be represented in the form (P. 338, [6]):

$$J_0(x) = \sum_{k=0}^{\infty} a_k T_{2k}\left(\frac{x}{q}\right), \quad -q \le x \le q,$$

where

$$a_n = \frac{2 - \delta_{n,0}}{\pi}\int_{-1}^{+1}\frac{J_0(qx)T_{2n}(x)}{\sqrt{1-x^2}}dx = 2(-1)^n J_n^2\left(\frac{q}{2}\right).$$

In turn, the Bessel function can be represented in the form (P. 34, [18]):

$$J_n(z) = \frac{z^n}{2^n\Gamma(n+1/2)\Gamma(1/2)}\int_0^{\pi} e^{iz\cos\theta}\sin^{2n}\theta d\theta.$$

The function can be estimated by its absolute value as

$$|J_n(z)| \le \frac{z^n}{2^n\Gamma(n+1/2)\Gamma(1/2)}\int_0^{\pi}\left|\sin^{2n}\theta\right| d\theta = \frac{z^n}{2^n n!}.$$

By using the integrals (1.2.5.2) from [15], the following equalities can be obtained:

$$\int_0^{\pi}\sin^{2n}\theta d\theta = \int_0^{\pi/2}\sin^{2n}\theta d\theta + \int_0^{\pi/2}\cos^{2n}\theta d\theta = \sqrt{\pi}\frac{\Gamma(n+1/2)}{\Gamma(n+1)}.$$

Using the following representation for the Gamma functions: $\Gamma(n+1) = n!$, $\Gamma(1/2) = \sqrt{\pi}$ and the Stirling formula

$$n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n\left(1 + \frac{1}{12n} + \frac{1}{288n^2} + O(n^{-3})\right)$$

the sought estimate can be obtained:

$$|\widetilde{R}_n| \le 2\sum_{k=n}^{\infty} J_k^2\left(\frac{q}{2}\right) \le 2\sum_{k=n}^{\infty} \frac{q^{2k}}{4^{2k}(k!)^2} \le \frac{4}{\pi qe}\sum_{k=n}^{\infty}\left(\frac{qe}{4k}\right)^{2k+1}.$$

□

For domain $|x| \le 8$, the following truncated Chebyshev series for $J_0(x)$ with error $O(10^{-6})$ can be written as (Table 9.1, [6])

$$\begin{aligned} J_0(x) \approx{}& 0.15772797147489011956 * T_0\left(\frac{x}{8}\right) \\ &- 0.00872344235285222129 * T_2\left(\frac{x}{8}\right) + 0.26517861320333680987 * T_4\left(\frac{x}{8}\right) \\ &- 0.37009499387264977903 * T_6\left(\frac{x}{8}\right) + 0.15806710233209726128 * T_8\left(\frac{x}{8}\right) \\ &- 0.03489376941140888516 * T_{10}\left(\frac{x}{8}\right) + 0.00481918006946760450 * T_{12}\left(\frac{x}{8}\right) \\ &- 0.00046062616620627505 * T_{14}\left(\frac{x}{8}\right) + 0.00003246032882100508 * T_{16}\left(\frac{x}{8}\right). \end{aligned} \tag{4}$$

The number of operations needed for calculation of the function $J_0(x)$ via formula (4) can be easily determined. Nine multiplications and nine additions are needed for calculation via formula (4) (the value $T_0(x) = 1$ was accounted for here). Eleven multiplications and nine additions are needed for calculation of the Chebyshev polynomials via recurrence relations (1), (2) whereas one more multiplication is needed for transition to the argument $0.125 * x = x/8$. In total, calculation via formula (4) requires 21 multiplications and 18 additions.

The Chebyshev polynomial can be expanded in powers of $x/8$. In this case, nine multiplications and nine additions are needed for calculation of the expression for the function $J_0(x)$ via the power polynomial. Additionally, ten multiplications are needed for calculation of $(x/8)^{2k}$, $k = \overline{1,9}$. In total, 19 multiplications and nine additions are needed.

The algorithm can be refined by using Horner's method:

$$\begin{aligned} J_0(x) \approx{}& 0.99999816 + z^2(-15.999699 + z^2(63.991859 + z^2(-113.69278 \\ &+ z^2(113.33067 + z^2(-71.484509 + z^2(29.990545 + z^2(-8.0280898 + 1.0636601z^2)))))))), \end{aligned}$$

where $y = x/8$. Complexity of the algorithm based on Horner's method constitutes ten multiplications and eight additions which include two multiplications: $(0.125 * x) * (0.125 * x)$. Aside from the algorithm, special approaches can be used with pre-treatment of the coefficients. Some of the methods were defined in [21] and the section 4.6.4 of [22]. Here, the method described in [23] will be used. For polynomials of the order $n = 2k$, the algorithm has the form

$$\begin{aligned} p_1 &= x\,(x + b_1)\,, \\ p_2 &= (p_1 + b_2)\,(p_1 + x + b_3) + b_4, \\ p_3 &= p_2\,(p_1 + b_5) + b_6, \\ &\cdots\cdots\cdots\cdots\cdots \\ p_k &= p_{k-1}\,(p_1 + b_{2k-1}) + b_{2k}, \quad k \ge 2. \end{aligned} \tag{5}$$

Such schemes will be called the **adaptation of coefficients** schemes. The algorithm includes k multiplications and $2k + 1$ additions. Application of the scheme can be expanded toward the case of polynomials of order $n = 2k + 1$. In this case, $k + 1$ multiplications and $2k + 2$ additions are needed for the calculation. Note that the coefficients of scheme b_k sometimes take complex values. In the case, the algorithm will be complicated through taking the coefficients as real. It leads to increase of the number of operations.

If algorithm (5) is applied directly to the power representation (4), then some of the complex coefficients b_k are obtained. Thus, the scheme is changed in such a way that the complex coefficients will be excluded. For this purpose, the term "$+y$" is added to p_3 and the following scheme will be obtained:

$$z = 0.125 * x, \quad y = z * z,$$

Table 1. The number of the operations necessary for calculation of $J_0(x)$ at $|x| \le 8$.

Type of operations	Subroutine in Toolkit	Chebyshev series	Power series	Horner's method	Adaptation of coefficients	Normalized adaptation
$*$	19	18	17	10	7	6
$+$	16	19	8	8	9	9
Total	35	37	25	18	16	15

$$p_1 = y(y - 2.3869021),$$
$$y = y + p_1,$$
$$p_2 = (p_1 + 1.0987437)\,(y - 0.2551663) + 0.67600329,$$
$$p_3 = (p_1 + 1.8584643)\,(y + 4.6311704) + 1.9407388,$$
$$J_0(x) \approx 1.0636601 p_2 p_3 - 3.4387229.$$

The number of operations for the algorithm is equal to seven multiplications and nine additions. One multiplication is necessary for calculation of the coefficient corresponding to the highest degree. However, the multiplication can be avoided. Thus, the following algorithm can be obtained:

$$z = 0.12548309 * x, \quad y = z * z,$$
$$p_1 = y(y - 2.4015149),$$
$$y = y + p_1,$$
$$p_2 = (p_1 + 1.1118167)\,(y - 0.25900994) + 0.69601147,$$
$$p_3 = (p_1 + 1.8671225)\,(y + 4.7195298) + 2.0662144,$$
$$J_0(x) \approx p_2 p_3 - 3.4387229. \tag{6}$$

Such schemes will be called the **normalized adaptation** schemes.

The total costs of mathematical operations for calculation of $J_0(x)$ carried out by various methods are shown in Table 1. Note that in [15], when approximating by polynomials on the interval $-3 \le x \le 3$, 13 multiplication operations and 6 additions were used (19 operations in total). At the same time, the resulting error was $O(10^{-8})$. The advantage of the interval used in this work is that the second interval for $x > 3$ in paper [15] will require significantly more calculations.

At $x \ge 8$, the following approximation (P. 350, [19]) can be used:

$$J_0(x) \approx \sqrt{\frac{2}{\pi x}} \left(P_0\left(\frac{64}{x^2}\right) \cos\left(x - \frac{\pi}{4}\right) + Q_0\left(\frac{64}{x^2}\right) \frac{8}{x} \sin\left(x - \frac{\pi}{4}\right) \right), \tag{7}$$

where the error of approximation of $J_0(x)$ depends on accuracy of calculation of the functions $P_0(x)$ and $Q_0(x)$. The following polynomials approach $P_0(x)$ and $Q_0(x)$ with the absolute errors 3×10^{-8} and 1.8×10^{-8} [19]:

$$P_0(t) = 0.9999969248 - 0.0010731477\, t,$$

$$Q_0(t) = 0.0156249820 - 0.000142707859\, t + 0.0000059374342\, t^2.$$

For calculation via formula (6), the following algorithms can be used:

$$p = 8/x, \quad q = p * p, \quad t = x - 0.78539816,$$
$$J_0(x) = \sqrt{0.079577472 * p} * ((0.99999692 - 0.0010731477 * q) * \cos t$$
$$+ (0.015624982 + (-0.00014270786 + 0.0000059374342 * q) * q) * p * \sin t).$$

The total number of operations can be calculated. It includes one division, nine multiplications, five additions, calculations of one square root and two trigonometrical functions. The sine and cosine

functions are taken here as two separated functions, but in practice, these functions are most often calculated as one function.

Instead of using formula (7), the other expression can be considered for calculation of $J_0(x)$ [20]:

$$J_0(x) = \sqrt{\frac{2}{\pi x}}\left(1 - \frac{1}{16x^2} + \frac{53}{512x^4}\right)\cos\left(x - \frac{\pi}{4} - \frac{1}{8x} + \frac{25}{384x^3}\right).$$

The following set of operations can be used for numerical realization of the method:

$$\begin{gathered} p = 0.63661977/x, \quad q = p * p, \\ J_0(x) = \sqrt{p} * (1 + (0.63021018 * q - 0.15421257) * q) * \cos(x - 0.78539816 \\ + (0.25232973 * q - 0.19634954) * p). \end{gathered} \tag{8}$$

The total of operations for the algorithm can be calculated. It includes one division, seven multiplications, five additions, calculations of one square root and one cosine function.

Calculation of the function in the Toolkit requires one division, twelve multiplications and nine additions. In addition, calculations one square root and one cosine function are required.

For improvement of approximation of the functions through reducing the error to 8×10^{-11} and 4.3×10^{-11}, it is necessary to calculate $P_0(t)$ and $Q_0(t)$ in formula (7) in the form

$$\begin{gathered} P_0(t) = 0.999999999921 - 0.001098628627856\ t + 0.0000273451040704\ t^2 \\ - 0.00000207337064\ t^3 + 0.000000209388721\ t^4, \end{gathered}$$

$$\begin{gathered} Q_0(t) = 0.01562499995755 - 0.0001430488765351\ t + 0.0000069111476516\ t^2 \\ - 0.0000007621095161\ t^3 + 0.00000009349451522\ t^4. \end{gathered}$$

Note that instead of representation (7), the decomposition can be used as described in [20].

4. COMPUTING THE BESSEL FUNCTION J_1

Theorem 2. *For the Bessel function of the first kind of order one, the following representation is true*

$$J_1(x) = 2\sum_{k=0}^{n-1}(-1)^k J_k\left(\frac{q}{2}\right) J_{k+1}\left(\frac{q}{2}\right) T_{2k+1}\left(\frac{x}{q}\right) + R_n(x), \quad -q \le x \le q,$$

with estimate

$$|\widetilde{R}_n| = |\max_{-q\le x\le q} R_n(x)| \le \frac{1}{\pi e(n+1)}\sum_{k=n}^{\infty}\left(\frac{qe}{4k}\right)^{2k+1}.$$

In particular, for the case $q = 8$, the following inequalities will be true $|\widetilde{R}_9| \le 8.45 \times 10^{-7}$ and $|\widetilde{R}_{13}| \le 5.12 \times 10^{-13}$.

Proof. The function $J_1(x)$ can be represented in the following form (P. 339, [6]):

$$J_1(x) = \sum_{k=0}^{\infty} a_k T_{2k+1}\left(\frac{x}{q}\right), \quad -q \le x \le q,$$

where

$$a_n = \frac{2 - \delta_{n,0}}{\pi}\int_{-1}^{+1}\frac{J_1(qx)T_{2n+1}(x)}{\sqrt{1-x^2}}dx = 2(-1)^n J_n\left(\frac{q}{2}\right) J_{n+1}\left(\frac{q}{2}\right).$$

The estimate for the Bessel function from the proof of the previous theorem and the Stirling formula can be used. Then, the following estimate is obtained as

$$|\widetilde{R}_n| \le 2\sum_{k=n}^{\infty}\left|J_k\left(\frac{q}{2}\right)\right| \cdot \left|J_{k+1}\left(\frac{q}{2}\right)\right|$$

$$\leq 2\sum_{k=n}^{\infty}\frac{q^{2k+1}}{4^{2k+1}(k!)^2(k+1)} \leq \frac{1}{\pi e(n+1)}\sum_{k=n}^{\infty}\left(\frac{qe}{4k}\right)^{2k+1}.$$

□

For domain $|x| \leq 8$, the following truncated Chebyshev series for $J_1(x)$ with error $O(10^{-7})$ can be written as (Table 9.2, [6])

$$\begin{aligned} J_1(x) \approx\ & 0.05245819033465648458 * T_1\left(\frac{x}{8}\right) \\ & + 0.04809646915823037394 * T_3\left(\frac{x}{8}\right) + 0.31327508236156718380 * T_5\left(\frac{x}{8}\right) \\ & - 0.24186740844740748475 * T_7\left(\frac{x}{8}\right) + 0.07426679621678703781 * T_9\left(\frac{x}{8}\right) \\ & - 0.01296762731173517510 * T_{11}\left(\frac{x}{8}\right) + 0.00148991289666763839 * T_{13}\left(\frac{x}{8}\right) \\ & - 0.00012227868505432427 * T_{15}\left(\frac{x}{8}\right) + 0.00000756263022969605 * T_{17}\left(\frac{x}{8}\right). \end{aligned} \tag{9}$$

The last formula that uses Horner's method can be written down in the form

$$\begin{aligned} J_1(x) \approx\ & z(3.9999927 + z^2(-31.99956 + z^2(85.325487 + z^2(-113.714 \\ & + z^2(90.7399 + z^2(-47.804681 + z^2(17.301693 + z^2(-4.1098182 + 0.49562453z^2)))))))). \end{aligned}$$

The adaptation of coefficients schemes are applied. The following numerical algorithm for calculation of $J_1(x)$ can be obtained:

$$\begin{gathered} z = 0.125 * x, \quad y = z * z, \\ p_1 = y(y - 2.5730502), \\ y = y + p_1, \\ p_2 = (p_1 + 0.2737503)\,(y + 7.5411595) + 15.764876, \\ p_3 = p_2\,(y + 0.31017717) - 0.35522006, \\ J_1(x) \approx z(0.49562453p_3\,(p_1 + 1.4985605) + 0.15639002). \end{gathered}$$

As well as in the case of $J_0(x)$, an initial algorithm contains complex coefficients. Thus, the coefficients can be gotten rid of and the algorithm with real coefficients can be obtained.

The normalized adaptation scheme for the function $J_1(x)$ takes the form

$$\begin{gathered} z = 0.11994381 * x, \quad y = z * z, \\ p_1 = y(y - 2.4087342), \\ y = y + p_1, \\ p_2 = (p_1 + 0.57043493)\,(y + 6.0949586) + 9.3528179, \\ p_3 = p_2\,(y + 0.24958757) - 0.18457525, \\ J_1(x) \approx z(p_3\,(p_1 + 1.3196524) + 0.18651755). \end{gathered} \tag{10}$$

The total number of operations for calculation via formula (9) with recurrence relations (2) and (3) can be calculated. The total costs of mathematical operations for calculation of $J_1(x)$ carried out by various methods are shown in Table 2. Note that in [15], when approximating by polynomials on the interval $-3 \leq x \leq 3$, 14 multiplication operations and 6 additions were used (20 operations in total). At the same time, the resulting error was $O(10^{-8})$. As in the case of calculating the function J_0, the advantage of the interval used in this work is that the second interval will require significantly less calculations.

At $x \geq 8$, the following approximation (P. 351, [19]) can be used:

$$J_1(x) \approx \sqrt{\frac{2}{\pi x}}\left(P_1\left(\frac{64}{x^2}\right)\cos\left(x - \frac{3\pi}{4}\right) - Q_1\left(\frac{64}{x^2}\right)\frac{8}{x}\sin\left(x - \frac{3\pi}{4}\right)\right), \tag{11}$$

Table 2. The number of the operations necessary for calculation of $J_1(x)$ at $|x| \le 8$.

Type of operations	Subroutine in Toolkit	Chebyshev series	Power series	Horner's method	Adaptation of coefficients	Normalized adaptation
$*$	16	19	18	11	8	7
$+$	15	20	8	8	9	9
Total	31	39	26	19	17	16

where the error of approximation of $J_1(x)$ depends on accuracy of calculation of the functions $P_1(x)$ and $Q_1(x)$. The following polynomials approach $P_1(x)$ and $Q_1(x)$ with the absolute errors 4×10^{-6} and 9.1×10^{-7} [19]:

$$P_1(t) = 1.000003987 + 0.001798103\, t,$$
$$Q_1(t) = 0.0468740856 - 0.0001925964\, t.$$

For calculation via formula (11), the following algorithm can be used:

$$p = 8/x, \quad q = p * p, \quad t = x - 2.3561945,$$
$$J_1(x) = \sqrt{0.079577472 * p} * ((1.000004 + 0.001798103 * q) * \cos t$$
$$- (0.046874086 - 0.0001925964 * q) * p * \sin t).$$

The total number of operations used in the algorithm can be calculated. They include one division, eight multiplications, four additions, one square rooting and two calculations of trigonometrical functions.

For calculation of the function, a different method is considered in [20] in the form:

$$J_1(x) = \sqrt{\frac{2}{\pi x}} \left(1 + \frac{3}{16x^2} - \frac{99}{512x^4}\right) \cos\left(x - \frac{3\pi}{4} + \frac{3}{8x} - \frac{21}{128x^3}\right).$$

For numerical realization of the method, the following set of operations can be used:

$$p = 1/x, \quad q = p * p,$$
$$J_1(x) = \sqrt{0.63661977 * p} \; * \; (1 + (0.1875 - 0.19335938 * q) * q)$$
$$* \cos\left(x - 2.3561945 + (0.375 - 0.1640625 * q) * p\right).$$

One division, eight multiplications, five additions, calculations of one square root and one sine function are needed for the realization.

The number of operations in the last algorithm can be reduced by one multiplication:

$$p = 0.63661977/x, \quad q = p * p,$$
$$J_1(x) = \sqrt{p} * (1 + (0.46263771 - 1.1771851 * q) * q)$$
$$* \cos(x - 2.3561945 + (0.58904862 - 0.63587091 * q) * p). \tag{12}$$

The needed operations include one division, seven multiplications, five additions, calculations of one square root and one cosine function.

As to the Toolkit library, one division, twelve multiplications, nine additions, calculations of one square root and one cosine function are needed for the calculation.

For improvement of approximation of the functions through reducing the error to 9×10^{-11} and 4.8×10^{-11}, it is necessary to calculate $P_1(t)$ and $Q_1(t)$ in the following form:

$$P_1(t) = 1.0000000000885 + 0.001831050000764\, t - 0.000035163964961\, t^2$$
$$+ 0.000002457520174\, t^3 - 0.000000240337019\, t^4,$$
$$Q_1(t) = 0.04687499995287 - 0.000200269087313\, t + 0.000008449199096\, t^2$$
$$- 0.00000088228987\, t^3 + 0.000000105787412\, t^4.$$

As well as for the case of calculation $J_0(x)$, it is also possible to use the decomposition described in [20] instead of the algorithm (12).

5. COMPUTING THE BESSEL FUNCTION J_m

Theorem 3. *For the Bessel function of the first kind at $m = 0, 1, \ldots$, the following representation is true*

$$J_{2m}(x) = \sum_{k=0}^{n-1} \epsilon_k J_{m+k}\left(\frac{q}{2}\right) J_{m-k}\left(\frac{q}{2}\right) T_{2k}\left(\frac{x}{q}\right) + R_n^0(x), \quad -q \le x \le q,$$

$$J_{2m+1}(x) = 2\sum_{k=0}^{n-1} J_{m+k+1}\left(\frac{q}{2}\right) J_{m-k}\left(\frac{q}{2}\right) T_{2k+1}\left(\frac{x}{q}\right) + R_n^1(x), \quad -q \le x \le q, \tag{13}$$

where $\epsilon_k = \{1, k = 0;\ 2, k = 1, 2, \ldots\}$ and

$$|\widetilde{R}_n^0| = \left|\max_{-q\le x\le q} R_n^0(x)\right| \le \frac{4}{\pi q e}\sum_{k=n}^{\infty}\left(\frac{qe}{4k}\right)^{2k+1},$$

$$|\widetilde{R}_n^1| = \left|\max_{-q\le x\le q} R_n^1(x)\right| \le \frac{1}{\pi e(n+1)}\sum_{k=n}^{\infty}\left(\frac{qe}{4k}\right)^{2k+1}.$$

Proof. The following formula holds for the even Bessel functions (P. 338, [6]):

$$J_{2m}(qx) = \sum_{n=0}^{\infty} \epsilon_n J_{m+n}\left(\frac{q}{2}\right) J_{m-n}\left(\frac{q}{2}\right) T_{2n}(x), \quad -1 \le x \le 1.$$

The remainder of the infinite series is estimated in the following manner:

$$|\widetilde{R}_n^0| \le 2\sum_{k=n}^{\infty} |J_{m+k}\left(\frac{q}{2}\right)| \cdot |J_{m-k}\left(\frac{q}{2}\right)| \le 2\sum_{k=n}^{\infty} \frac{q^{k+m+|k-m|}}{4^{k+m+|k-m|}(k+m)!|k-m|!}$$
$$\le 2\sum_{k=n}^{\infty} \frac{q^{2k}}{4^{2k}(k!)^2} \le \frac{4}{\pi q e}\sum_{k=n}^{\infty}\left(\frac{qe}{4k}\right)^{2k+1}.$$

The odd Bessel functions can be represented in the form (P. 338, [6])

$$J_{2m+1}(qx) = 2\sum_{n=0}^{\infty} J_{m+n+1}\left(\frac{q}{2}\right) J_{m-n}\left(\frac{q}{2}\right) T_{2n+1}(x), \quad -1 \le x \le 1$$

with the remainder

$$|\widetilde{R}_n^1| \le 2\sum_{k=n}^{\infty} |J_{m+k+1}\left(\frac{q}{2}\right)| \cdot |J_{m-k}\left(\frac{q}{2}\right)| \le 2\sum_{k=n}^{\infty} \frac{q^{k+m+1+|k-m|}}{4^{k+m+1+|k-m|}(k+m+1)!|k-m|!}$$
$$\le 2\sum_{k=n}^{\infty} \frac{q^{2k+1}}{4^{2k+1}(k!)^2(m+k+1)} \le \frac{1}{\pi e(n+1)}\sum_{k=n}^{\infty}\left(\frac{qe}{4k}\right)^{2k+1}.$$

□

Values of $J_m(4)$ necessary for calculation of the first twenty Bessel functions are given in Table 3.

It is cost-efficient to carry out calculations via formulas (13) for the known number m. The formulas from Lemma 1 and Lemma 2 are special cases of (13). If necessary to calculate the functions for any $m = 1 \ldots 20$, then 26 coefficients $J_k(q/2)$ are necessary to store in the memory.

Below, some other algorithms will also be considered. For example, the recurrence relation can be used for calculation of values of the functions $J_m(x)$:

$$J_{m+1}(x) = \frac{2m}{x} J_m(x) - J_{m-1}(x).$$

The increase in the error of calculation of $J_0(x)$ and $J_1(x)$ will not take place, if $x > 2m$. The increase in the order of m leads to the increase in the value of multiplier $2m/x$ and, hence, to deterioration of the estimate of approximation of $J_{m+1}(x)$ with respect to $J_m(x)$ by the factor of $2m/x$.

Table 3. Values of the function of $J_m(4)$.

m	$J_m(4)$	m	$J_m(4)$
0	$-3.97149809863847510000e-01$	10	$1.95040554660034210000e-04$
1	$-6.60433280235490920000e-02$	11	$3.66009120826084370000e-05$
2	$3.64128145852072980000e-01$	12	$6.26446179431221040000e-06$
3	$4.30171473875621820000e-01$	13	$9.85858683264748730000e-07$
4	$2.81129064961360030000e-01$	14	$1.43619646908673070000e-07$
5	$1.32086656047098290000e-01$	15	$1.94788450959593060000e-08$
6	$4.90875751563855380000e-02$	16	$2.47169131102181330000e-09$
7	$1.51760694220584420000e-02$	17	$2.94685392215174880000e-10$
8	$4.02866782081900100000e-03$	18	$3.31345228071837140000e-11$
9	$9.38601861217564450000e-04$	19	$3.52531304947822489586e-12$

For calculation of $J_n(x)$ in the case $x < 2m$, formula (9.1.46) from [15] can be used:

$$1 = J_0(x) + 2\sum_{m=1}^{\infty} J_{2m}(x).$$

The formula can be rewritten as

$$1 = J_0(x) + 2\sum_{m=1}^{M} J_{2m}(x) + 2\sum_{m=M+1}^{\infty} J_{2m}(x). \tag{14}$$

At the numerical realization of (14), the value of $M = M(x, \varepsilon)$ is chosen such that the remainder of the series in (14) does not exceed a certain given value of ε. Thus, the assumption that $J_m(x) = 0$ for $m > M$ can be adopted. Then, using the inverse recurrence relation

$$J_{m-1}(x) = \frac{2m}{x} J_m(x) - J_{m+1}(x), \tag{15}$$

expressions for calculation of all the functions J_m at $m < M$ via J_M will be obtained.

At calculation of all the values of $J_m(x)$, $m < M$ via formula (15), the values of functions with an even index will be summed. If the number of m coincides with the given value of n, then the value of the function will be stored. The stored value is then divided by the sum calculated in the right-hand side of (14).

Below, ways of choosing the value of M will be considered in detail. Note that in the Toolkit, the value is chosen as $M = n + \sqrt{40n}$. The choice is often justified but in certain cases, it can lead to worsening the error of calculation of $J_n(x)$ to $O(10^{-4})$. The purpose is to choose the value of M such that $|J_m(x)| < \varepsilon$ at $m > M$. Also, the choice of the value of M must be made such that remainder of the series in (14) can be neglected. Apparently, satisfying the $|J_m(x)| < \varepsilon \sim 10^{-6}$ is sufficient for the values of x less the first extremum of $J_m(x)$. Level lines of $J_n(x) = 8 * 10^{-6}$ will be considered. They are denoted by dashed lines in Fig. 1. The values of n to the right of the solid line $n = 1.282 * x + 8.692$ correspond to the values less than $8 * 10^{-6}$.

6. NUMERICAL RESULTS

Numerical experiments using a video card Tesla C1060 and a quad-core processor i7–920 having clock frequency 2.67 GHz are carried out. Note that the used video card has 240 streaming processors. Besides, each multiprocessor is capable of simultaneously processing 32 threads of one warp. Thus, using the video card, in the ideal case one can run simultaneously 240 by 32 computational processes.

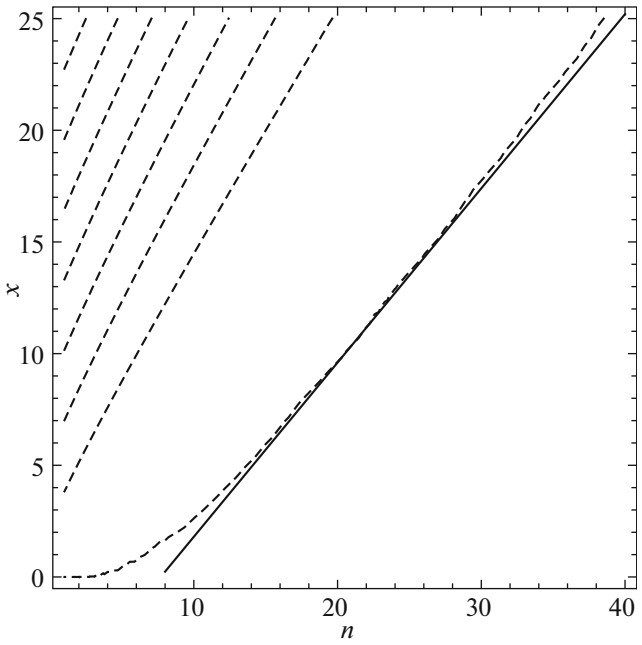

Fig. 1. Dashed lines are for level lines of $J_n(x) = 8 * 10^{-6}$. Solid line is for the line $n = 1.282 * x + 8.692$.

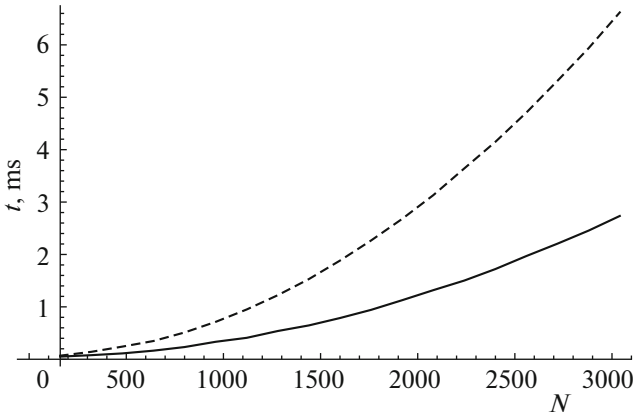

Fig. 2. Dependence of time consumed for filling the matrix with values of the function $J_0(x)$, $0 \le x \le 8$ on the matrix size N. Solid line is for normalized adaptation. Dashed line is for the Toolkit function. Calculations performed on the GPU.

Apparently, the maximum performance improvement is expected to take place at the number of threads exceeding 7680.

Dependence of time consumed for filling the matrix t with values of the Bessel function on the matrix size N is investigated. The test is also essential from the practical point of view. The matrices completely filled with the Bessel functions can be needed, for example, in solving integral equations by using numerical methods. If the GPU is used for computations, the number of threads in the blocks can be chosen in a more efficient manner such that the warps are filled with threads completely. For example, one can choose $N = 16$ and obtain the total number of threads equal to $N \times N = 256$ which is multiples of the number of threads in the warp, i.e. 32.

Results of the tests of time consumed for filling the matrix with values of the function $J_0(x)$ at $0 \le x \le 8$ are presented in Fig. 2. The solid line is for algorithm (6), whereas the dashed line is for functions of the Toolkit. In both cases, the filling time has a parabolic dependence on the matrix size N. For example, for filling the matrix 3000×3000, the filling times 2.7 ms and 6.6 ms are needed for algorithm (6) and for functions of the Toolkit, respectively. Note that the parabolic dependence of the

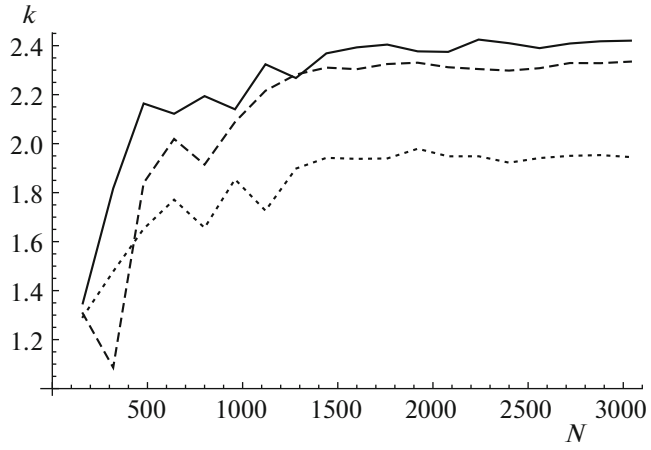

Fig. 3. Dependence of the computational speedup factor k on the matrix size N for new algorithms and the functions of the Toolkit. The case is $0 \le x \le 8$. Solid line is for $J_0(x)$, dashed line is for $J_1(x)$, and dotted line is $J_5(x)$. The calculations are performed on the GPU by the formula (15).

matrix filling time on the matrix size is observed for all the used methods. However, it must be noted that for smaller matrix sizes N, the filling time grows slower that N^2. It is due not only to the incomplete multiprocessor utilization (the number of streams is less than 7680), but also to off times present at the small number of streams.

Graphs depicting dependence of the computational speedup factor k on the matrix size N for new algorithms and for the functions of the Toolkit are presented in Fig. 3. The function domain is from 0 to 8. The solid line is for the speedup factor k of calculation of $J_0(x)$ using the algorithm (6). The dashed line is for calculation of $J_1(x)$ using the algorithm (10). At greater matrix sizes, both lines go to values near $k \approx 2.3 - 2.4$. The dotted line is for the speedup factor k for calculation of $J_5(x)$. In this case, the speedup factor is slightly smaller ($k \approx 1.9$). Note that for $J_n(x)$ it is slightly more complicated to compare speeds of calculation of the functions since the algorithms presented in this work and the functions in the Toolkit depend differently on relationships for index n and on values of the argument x of the function $J_n(x)$.

Improvement of performance in calculations of elements of the matrix $N \times N$ for the algorithms using the card Tesla C1060 versus calculations carried out with the help of the processor i7 combined with the technology OpenMP is presented in Fig. 4. Calculations using the CPU were carried out using functions from the library math.h. It is worth noting that functions at CPU allow calculating the Bessel functions with double precision. For both $J_0(x)$ (solid line) and $J_1(x)$ (dashed line), the speedup factor $k \approx 50$. The graphs are plotted for $0 \le x \le 8$. Similar results are obtained for the case $x > 8$.

Calculations of values of $J_n(x)$ via formula (13) will be discussed next. For calculations at CPU, the computational speed is more than doubled compared to the other algorithms. In contrast, for calculations at GPU, the computational speed decreases by several times. It can be best explained with differences in addressing to memories by CPU and GPU.

If performance of algorithms for calculations of $J_0(x)$ and $J_1(x)$ at CPU described in the present work is compared versus functions of the library math.h, then the speedup achieved by application of our algorithms makes up approximately 1.8. Thus, it can be concluded that improvement of performance achieved at calculations at GPU versus CPU is by a factor of over 27. It must be also taken into account that during testing, in addition to carrying out calculations of certain functions, performed also is writing the calculated values into both graphical memory and RAM.

A few words about efficiency of using several threads can be said. For the computational tests, eight threads at four cores are used. Note that at using two threads instead of one, the speedup was equal to approximately two. The use of four threads improves performance by a factor of 3.5–3.8. The advantage of calculations at eight threads versus four threads makes up 1.2–1.4. Apparently, it must be kept in mind that the use of the technology "Hyper-Threading" for the other type of the CPU architecture can provide different numbers.

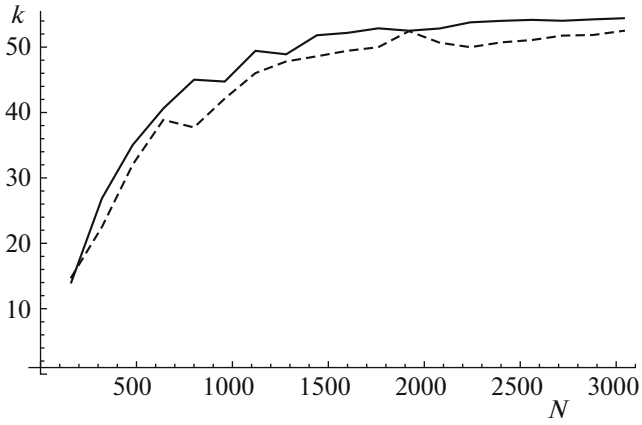

Fig. 4. Dependence of the computational speedup factor k on the matrix size N for new algorithms and for calculation at CPU. The case is $0 \le x \le 8$. Solid line is for $J_0(x)$, dashed line is for $J_1(x)$.

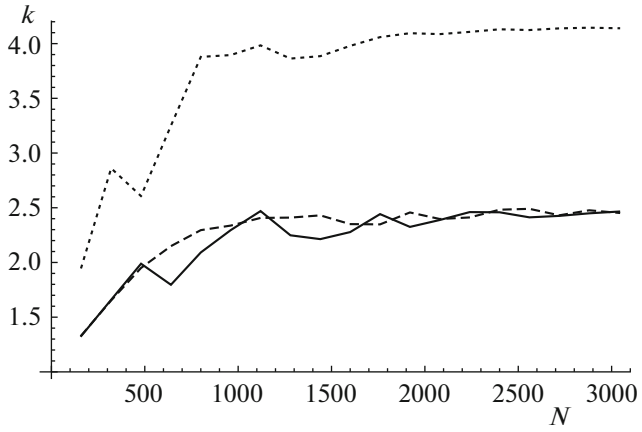

Fig. 5. Dependence of the computational speedup factor k on the matrix size N for new algorithms and the functions of the Toolkit. The case is $8 \le x \le 16$. Solid line is for $J_0(x)$, dashed line is for $J_1(x)$, and dotted line is $J_5(x)$.

Results for computational speedups for functions $J_0(x)$ and $J_1(x)$ at $x > 8$ for algorithms (8) and (12) versus functions of the Toolkit are presented in Fig. 5. As in the case $0 \le x \le 8$, the speedup makes up the value of 2.3.

7. CONCLUSIONS

In the presented work fast algorithms for calculations of the Bessel functions of float-type arguments of integer order $J_n(x)$ with the error $O(10^{-6})$ were developed. It was shown that for $J_0(x)$ and $J_1(x)$ the increase in the speed of calculations versus calculations carried out in the Toolkit is by a factor of 2.3–2.4. This result was achieved through the use of normalized adaptation for Horner's method. Efficiency of calculations carried out with the use of the video card Tesla C1060 versus calculations carried out with the use of the processor i7–920 was analyzed. It was shown that the improvement in performance when the presented algorithms are used makes up the value over 27, and improvement in performance versus functions of the library math.h used for filling the large matrices with the values of $J_0(x)$ and $J_1(x)$ makes up the value of approximately 50.

FUNDING

The work is performed according to the Russian Government Program of Competitive Growth of Kazan Federal University.

REFERENCES

1. B. G. Korenev, *Bessel Functions and their Applications* (Taylor and Francis, London, 2002).
2. A. V. Anufrieva, K. B. Igudesman, and D. N. Tumakov, "Peculiarities of elastic wave refraction from the layer with fractal distribution of density, " Appl. Math. Sci. **8**, 5875–5886 (2014).
3. N. S. Bakhvalov, N. P. Zhidkov, and G. M. Kobelkov, *Numerical Methods* (Binom, Laboratoriya znanii, Moscow, 2003) [in Russian].
4. S. Pashkovskii, *Computing Applications of Polynomials and Chebyshev's Rows* (Nauka, Moscow, 1983) [in Russian].
5. O. B. Arushanyan, N. I. Volchenskova, and S. F. Zaletkin, "About some properties of the partial sums of Chebyshev's rows," Differ. Uravn. Prots. Upravl. **3**, 26–34 (2009).
6. Y. L. Luke, *Mathematical Functions and their Approximations* (Academic, New York, 1975).
7. E. A. Karatsuba, "Fast calculations of transcendental functions," Probl. Peredachi Inform. **27**, 76–99 (1991).
8. A. V. Ponomarenko, "Development of model realization for the Bessel functions from the LSB standard," Tr. Inst. Sist. Programm. RAN **10**, 115–142 (2006).
9. N. Brisebarre, J.-M. Muller, and A. Tisserand, "Computing machine-efficient polynomial approximations," ACM Trans. Math. Software **32**, 236–256 (2006).
10. J. C. Mason and D. Handscomb, *Chebyshev Polynomials* (Chapman and Hall/CRC, Boca Raton, FL, 2003).
11. W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing* (Cambridge Univ. Press, Cambridge, 1992).
12. W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in Fortran 77: The Art of Scientific Computing* (Cambridge Univ. Press, Cambridge, 1992).
13. V. Dzhambov and S. Drangajov, "Computing of special functions with arbitrary precision in the environment of .NET framework," Cybern. Inform. Technol. **11** (2), 32–45 (2011).
14. J. N. Newman, "Approximations for the Bessel and Struve functions," Math. Comput. **43** (168), 551–556 (1984).
15. M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables* (Dover, New York, 1972).
16. L. Fox and I. B. Parker, *Chebyshev Polynomials in Numerical Analysis* (Oxford Univ. Press, London, 1968).
17. Yu. A. Klokov and A. Ya. Shkerstena, "Assessment of an error of interpolation formulas of Lagrange-Chebysheva," Zh. Vychisl. Mat. Mat. Fiz. **27**, 1418–1420 (1987).
18. G. N. Watson, *A Treatise on the Theory of Bessel Functions* (Macmillan, Univ. Press, New York, 1945).
19. B. A. Popov and G. S. Tesler, *Functions Evaluation on the PC* (Naukova Dumka, Kiev, 1984) [in Russian].
20. J. Harrison, "Fast and accurate Bessel function computation," in *Proceedings of the 19th IEEE Symposium on Computer Arithmetic, ARITH 2009, Portland, Oregon, June 8–10, 2009* (2009), pp. 104–113.
21. V. Ya. Pan, "About ways of calculation of values of polynomials," Usp. Mat. Nauk **21**, 103–134 (1966).
22. D. Knuth, *The Art of Computer Programming* (Addison-Wesley Professional, Reading, MA, 1997), Vol. 2.
23. E. G. Belaga, "Calculation of polynomials—from Newton up to now," Kvant **7**, 29–35 (1974).