How to solve the "A Hundred-dollar, Hundred-digit Challenge"

Eckard Specht[1], Martin Wohlgemuth[2], Clemens Raab[3], Lutz Gehlen[4]

[1]Department of Experimental Physics, Otto-von-Guericke-University of Magdeburg,
D–39106 Magdeburg, Germany
specht@hydra.nat.uni-magdeburg.de

[2]D–58456 Witten, Germany
wohlgemuth@matroid.com

[3]student of technical mathematics at the Johannes-Kepler-University of Linz,
A–4040 Linz, Austria
clemens.raab@liwest.at

[4]student of physics at the Ruprecht-Karls-University of Heidelberg,
D–69120 Heidelberg, Germany
lutz.gehlen@gmx.net

Introduction

On March 11, 2002, one of the authors discovered at Eric Weisstein's "World of Mathematics" website a notice that SIAM News has published recently "A Hundred-dollar, Hundred-digit Challenge" proposed by Nick Trefethen, Oxford University. Normally, most of the authors spend a fraction of their time to answer questions of pupils and students to elementary mathematical problems on an Internet bulletin board called "Mathe-Treff". These questions repeat every few months over and over again, so that this challenge was a welcome change for us. A couple of days later it turns out that we formed a team with four contestants from different backgrounds, institutions and countries.

Numerical aspects

In order to solve these very hard problems it is necessary to have some experience not only in numerics but also in other mathematical topics such as analysis, linear algebra, approximation and probability theory. Nevertheless, a broad knowledge of numerical methods is the most important fund.

Four different architectures, each with its own precision, were used for the calculations:

- "normal" `double` precision arithmetics (53 mantissa bits, approx. 16 decimal places) on an Intel Pentium (800 MHz processor, 1 GB Memory) with Linux' `gcc/g++`-Compiler,
- `long double` precision arithmetics (113 mantissa bits, approx. 34 decimal places) on a Compaq GS 160 (16×731 MHz processors, 16 GB memory) running Tru64 UNIX,
- the GNU Multiple Precision Arithmetic Library (GMP) [1], version 4.0.1, with arbitrary number of mantissa bits on both machines, and
- program Mathematica 4.1 on a Intel Pentium III 800 MHz notebook which allows arbitrary precision, too.

It should be mentioned that the GMP library is a great tool for everyone who likes to perform high precision calculations. If you have developed a program that uses `double` precision, you have simply to change all `double` variables into `mpf_class` variables (using the C++ interface of the GMP library) in the source text of your program in order to get arbitrary precision. This is a very convenient and time-saving way to improve the accuracy of calculations using non-standard arithmetics.

Program development was done on the Pentium boxes and several other computers while the mainframe was used for those problems which need a huge amount of memory.

Results

What follows are the main ideas to solve the ten proposed problems of the challenge. Numerical results given in this section are truncated to 12–15 decimal places. For more digits see Appendix.

#1 It is not a good idea to attack the integral in its proposed form

$$I = \int_0^1 \frac{\cos(\frac{\ln x}{x})}{x}\,\mathrm{d}x \tag{1}$$

with numerical integration schemes (i.e. ROMBERG integration) since the integrand oscillates with rising frequency and amplitudes as x tends to 0^+ (Fig. 1a). A simple substitution $t = 1/x$ transforms the integral into

$$I = \int_1^\infty \frac{\cos(t \ln t)}{t}\,\mathrm{d}t. \tag{2}$$

After this transformation the oscillations still remain, but their amplitudes decrease with larger values of t (Fig. 1b). Finally, it is desirable to have equidistant zeros of the integrand in (2), so we perform another substitution: $u = t \ln t$. It is clear that this transformation cannot be inverted in closed analytical form. With the help of LAMBERT's W-Function [2] (which is the inverse $W = W(u)$ of $u(W) = W\mathrm{e}^W$) we can write the integral in the following form:

$$I = \int_0^\infty \frac{\cos u}{\frac{u}{W(u)} + u}\,\mathrm{d}u = \int_0^\infty \cos(u) f(u)\,\mathrm{d}u, \qquad f(u) = \frac{W}{u(W+1)}. \tag{3}$$

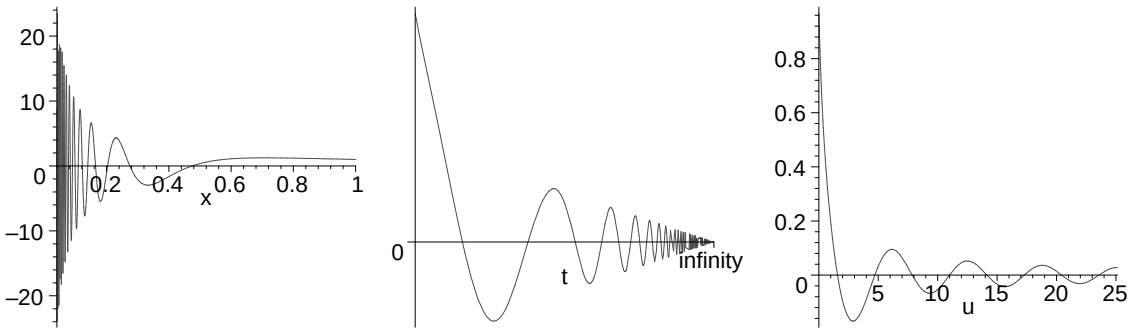

Figure 1. Sketches of the integrands in equations (1), (2) and (3).

Here $f(u)$ is bounded but decreases only with $O(1/u)$ for $u \to \infty$. A common technique to achieve a high accuracy in numerical integration is i) to split the integral into two parts:

$$I = \int_0^{\bar{u}} \cos(u) f(u)\,\mathrm{d}u + \int_{\bar{u}}^\infty \cos(u) f(u)\,\mathrm{d}u = I_1 + I_2, \tag{4}$$

where I_1 is well suited for a standard integration routine, and ii) to increase the rate of convergence in the second term by repeated integration by parts. The result simplifies if we choose $\bar{u}$ as an integer multiple of π. Analytical manipulations show that the kth derivative can be written as

$$f^{(k)}(u) = \frac{(-1)^k}{u^{k+1}(W+1)^{2k+1}} \cdot W^{k+1} \cdot \tau_k(W)$$

where $\tau_k(W)$ is a polynomial of degree k with integer coefficients which satisfies the recurrence relation

$$\tau_{k+1}(W) = [(k+1)W + 3k + 2]\tau_k(W) - (W+1)\tau_k'(W).$$

The coefficients $c_{k,j}$ of the polynomials

$$\tau_k(W) = \sum_{j=0}^{k} c_{k,j} W^j$$

may be obtained by the following formulae:

$$\left.\begin{array}{l} c_{0,0} = 1, \\ c_{k,k+1} = 0, \quad c_{k+1,0} = (3k+2)c_{k,0} - c_{k,1}, \\ c_{k+1,j} = (k+1)c_{k,j-1} + (3k-j+2)c_{k,j} - (j+1)c_{k,j+1}, \quad j = 1, \ldots, k, \\ c_{k+1,k+1} = (k+1)c_{k,k}. \end{array}\right\} \quad k = 0, 1, \ldots$$

Carrying out the integration by parts we get an asymptotic expansion of the indefinite integral I_2 which is semiconvergent:

$$I_2 = -\cos(\bar{u})\left[f'(\bar{u}) - f'''(\bar{u}) + \cdots + (-1)^{k-1} f^{(2k-1)}(\bar{u})\right] + (-1)^k \int_{\bar{u}}^{\infty} \cos(u) f^{(2k)}(u)\,\mathrm{d}u. \qquad (5)$$

If we choose a large value for $\bar{u}$ the summands in the bracketed expression decrease very rapidly. Together with a high precision numerical integration for the integral I_1 in (4) there is no need to evaluate the remaining integral in (5).

Our result: `0.32336 74316 77778 ...`

#2 We believe that this problem is the easiest from all ten problems, and it was indeed the first we solved within a few days. A good starting point was a self-written library of routines which allows to draw exact geometric figures. All what we need to calculate the path of the photon are intersections of lines and circles and how to mirror lines at normals. With `double` precision arithmetics we got a first rough sketch of the situation (Fig. 2).

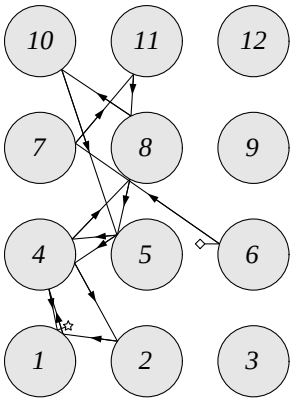

Figure 2. Path of the photon starting at $(x, y) = (0.5, 0.1)$ (marked by a diamond; the circle around the origin has number 5) and ending up at $(x, y) = (-0.736, -0.669)$ (marked by a penta-star).

No transcendental functions like $\sin(x)$, $\cos(x)$, $\exp(x)$ or $\log(x)$ are necessary to perform the calculations, only the square root $\sqrt{x}$ is needed. However, this is a built-in function in the GMP library, so that the program could be written quickly. Finally only a linear interpolation has to be done at the last line segment to get the desired position after $t = 10$.

Our result: `0.99526 29194 43354 ...`

#3 The problem here is to calculate the *spectral norm*

$$||A||_2 = \sqrt{\text{maximum eigenvalue of } C = A^{\mathrm{T}}A}$$

of an infinite matrix A. The fact that A is an infinite matrix is not really cumbersome since it is a natural approach to take only square $N \times N$ submatrices from A (or much better from C) and one may hope that the norm converges with increasing N. So the problem reduces to an effective calculation of the largest eigenvalue of C which is clearly a symmetric matrix. Several attempts were made to tackle this problem (Jacobi transformations, chapter 11.1 in [3] and other), but the simplest of all, the *power method* (chapter 8.5.3 in [4]), has been shown to be the most effective to find the dominant eigenvalue. Here is a short outline of the method:

Step 1: Choose x_0

Step 2: For $k = 1, 2, 3, \ldots$ do until convergence

$$\hat{x}_k = Cx_{k-1}$$

$$x_k = \hat{x}_k / \max(\hat{x}_k)$$

End

This algorithm is very easy to implement and shows a rapid convergence with the starting vector $x_0 = (1, 1, \ldots, 1)$. Nevertheless, this problem has turned out to be very time-comsuming since the matrix multiplication $C = A^{\mathrm{T}}A$ is a N^3 process. One of the Compaq processors needed over a week to establish C with a dimension of $N = 16\,000$ (remember, this task requires over 4 GB of memory, and C has $2.56 \cdot 10^8$ elements for which approximately $4 \cdot 10^{12}$ multiplications have to be performed!). Actually, an application of the power method for $N = 100, 200, 300, \ldots, 16\,000$ shows a sufficient convergence within the claimed accuracy of ten decimal places.

But we can do better. Because the given matrix A has a well-defined structure with elements

$$A_{ij} = \frac{2}{i^2 + 2ij + j^2 - i - 3j + 2}$$

the same is true for matrix C:

$$C_{ij} = \sum_{k=1}^{\infty} A_{ik}A_{jk} = 4\sum_{k=1}^{\infty} \frac{1}{k^2 + (2i-3)k + (i^2 - i + 2)} \cdot \frac{1}{k^2 + (2j-3)k + (j^2 - j + 2)}.$$

If one finds an analytical expression for this series which converges much better than $O(k^{-4})$, then the elements C_{ij} can be calculated in a separate routine which gets us rid of all memory problems and the time-consuming matrix multiplication. For the diagonal elements C_{ii} we find [5]

$$C_{ii} = \frac{8\pi}{a^3}\tanh\left(\frac{\pi a}{2}\right) - \frac{4\pi^2}{a^2}\operatorname{sech}^2\left(\frac{\pi a}{2}\right) - 64\sum_{l=0}^{i-2} \frac{1}{[(2l+1)^2 + 8i - 1]^2}.$$

Unfortunately, we haven't found an appropriate expression in the case $i \neq j$ yet.

Our result: `1.27422 41528 20684 ....`

#4 By determining bounds for each summand of the function one gets a rough idea of how small the minimum could be. The first summand is in the range $[\mathrm{e}^{-1}; \mathrm{e}]$ the next four summands are in $[-1; 1]$ and the last summand is always non-negative. Taking the lowest value for each summand a lower bound for the function is $\mathrm{e}^{-1} - 4 = -3.63$. The next question was if the function gets below zero at all. From the last summand we concluded that all negative values must occur within a circle around the origin with radius $r = 2\sqrt{4 - \mathrm{e}^{-1}} = 3.81$. Now we computed $f(x, y)$ at a few thousand points in this circle (with step size 1/10) and it turned out that there are even values below -2. The smallest value was $f(-0.4, -0.1) = -2.97$. This and the fact that $\sin(\sin(80y))$

actually lies in $[-\sin(1); \sin(1)] = [-0.841; 0.841]$ reduces the circle containing the global minimum to $r = 1.42$. Then we started taking the oscillations into account. We determined the periods of each summand within this circle and took an appropriate fine raster of points as the starting points of three minimizing algorithms (two Mathematica built-in functions and one self-written program). This fine raster guarantees that no local minimum is omitted and hence the smallest of these minima is the global minimum.

Our result: `-3.30686 86474 75237 ...`.

#5 This is the hardest of all ten problems and we are not sure if the given value is correct. The easiest part is to obtain a formula for the function of interest:

$$\frac{1}{\Gamma(z)} = \sum_{k=1}^{\infty} a_k z^k, \quad a_0 = 1, \quad |z| < \infty, \tag{6}$$

$$r a_r = \sum_{k=1}^{r} (-1)^{k+1} S_k a_{r-k}, \quad r > 0,$$

$$S_1 = -\Psi(1) = \gamma = 0.57721\ldots, \quad S_k = \sum_{r=1}^{\infty} r^{-k}, \quad k > 1.$$

Let us write for the cubic polynomial

$$p(z) = a + bz + cz^2 + dz^3, \quad a, b, c, d \in \mathbb{C} \tag{7}$$

with complex constants a, b, c and d. If we truncate the TAYLOR series expansion (6) after the cubic term, the coefficients $a_0 = 1$, $a_1 = 0.57722$, $a_2 = -0.65587$, $a_3 = -0.04200$ are good start values for the unknown coefficients a, b, c, d in (7). It is clear that these coefficients approximate $1/\Gamma(z)$ more in a least-square sense than in the supremum norm.

The following two observations have been made: i) the point on the unit disk at which $\|f - p\|_\infty$ reaches its maximum lies always at the boundary of the domain, i. e. on the unit circle; ii) the coefficients a, b, c, d have only real parts. The latter is evidently true since the coefficients of (6) are real, too. Known algorithms in the one-dimensional case based on alternants like REMEZ's exchange algorithm are not applicable here.

Nevertheless a few articles have been found in literature [6], [7] to overcome these difficulties in approximation on the complex unit disk. The proposer himself gave in [6, p. 348 (Figure 2)] a value of 0.217 (we believe that his Undergraduate thesis [8] contains more text, but this work was not accessible for us). He used the CARATHÉODORY-FEJÉR *(CF) method* to get near-best approximations. Applying this method to (6) we found a eigenvalue of $\lambda = 0.211958599426322$. On the other hand, one may consider this approximation problem as an *optimization problem* for the unknown coefficients in (7) which may be solved numerically by any multidimensional minimization program. But the convergence would normally be poor because $\|f - p\|$ is nondifferentiable at the optimal point. By this approach we got a (maximal) value of 0.22393637.

Our (experimental) result: `0.22393 637 ...`.

#6 This problem belongs to the general class of *random walks with drift*. The flea moves on a two-dimensional integer lattice. Any sequence of consecutive movements on the integer lattice we call a *tour*. In common manner we abbreviate the four possible directions by N, S, E and W. A tour therefore can be described as a sequence built from these letters where a letter N, S, E, W at position i stands for a movement from position i to position $i + 1$ by taking one step to the North, South, East or West, respectively. The *length* of a tour is the number of steps. A tour that leads back to the origin will be called a *round trip*. A *proper round trip* is a round trip that starts and ends at the origin and does not touch the origin at intermediate steps. We start with some basic considerations:

- The number of steps in a proper round trip is even.

- In a proper round trip the number of W's equals the number of E's and the number of N's is the same as the number of S's.
- Any two proper round trips with the same number of E's and N's have the same probability.

We decided to handle the problem as a combinatorical one and worked on the counting of proper round trips of given length $2k$ with exactly r steps to the east. If $m(k,r)$ is this number, then the probability $p(e)$ of ever returning to the origin is given by the infinite series

$$p(e) = \sum_{k=1}^{\infty} \sum_{r=0}^{k} m(k,r) \left(\frac{1}{4}\right)^{2k-2r} \left(\frac{1}{4} - e\right)^{r} \left(\frac{1}{4} + e\right)^{r},$$

where e is the drift, or equivalently

$$p(e) = \sum_{k=1}^{\infty} \sum_{r=0}^{k} m(k,r) \left(\frac{1}{16}\right)^{k-r} \left(\frac{1}{16} - e^2\right)^{r}$$

As the problem asks for a finite value of these infinite series we felt free to assume convergence and by exchanging the order of summation we yielded the following expression:

$$p(e) = \sum_{r=0}^{\infty} \left[(1 - 16\, e^2)^r \sum_{k=r}^{\infty} m(k,r) \left(\frac{1}{16}\right)^{k} \right]$$

The inner sum does not depend on e. Thus we denoted

$$T(r) := \sum_{k=r}^{\infty} m(k,r) \left(\frac{1}{16}\right)^{k}.$$

and got the final form of our formula:

$$p(e) = \sum_{r=0}^{\infty} (1 - 16\, e^2)^r T(r). \tag{8}$$

From [9], [10] we learned that (8) diverges for $e = 0$. Actually, we expected convergence in the case $e > 0$ and made efforts to compute exact values for as many $T(r)$ as possible. For computing $T(r)$ it is crucial to find a formula for the $m(k,r)$.

It is relatively easy to count the number of round trips and of round trips with fixed number of steps to the East. The number of round trips of length n is $\binom{2n}{n}^2$ and the number of round trips with exactly r steps east is $\binom{2n}{n}\binom{n}{r}^2$. The set of round trips with fixed number of E's can be disjunctively partitioned as follows: Any round trip of length $2n$ starts with a proper round trip of length $2k$ followed by a round trip of length $2n - 2k$. If we'd ask for round trips of length $2k$ with exactly r steps to the East, and denote this number by $a(k,r)$ we see that

$$a(k,r) = \sum_{i=0}^{k} \sum_{j=0}^{r} m(k-i, r-j) a(i,r).$$

For $i = j = 0$ the summand becomes $m(k,r)a(0,0)$. Let $a(0,0)$ be 1 and for convenience $a(k,r) = 0$ for $r > k$. By resolving for $m(k,r)$ we got

$$m(k,r) = a(k,r) - \sum_{i=1}^{k} \sum_{j=1}^{r} m(k-i, r-j) a(i,r)$$

Moreover, we found some useful recurrences for the $a(k,r)$ and the $m(k,r)$:

$$a(0,0)=1, \qquad a(k,0)=\frac{2(2k-1)}{k}\,a(k-1,0), \qquad a(k,r)=a(k,k-r),$$

$$a(k,r)=\frac{2k(2k-1)}{(k-r)^2}\,a(k-1,r), \qquad a(k,r)=\frac{(k+1-r)^2}{r^2}\,a(k,r-1),$$

$$m(1,0)=m(1,1)=2, \qquad m(k,r)=m(k,k-r),$$

$$m(k,0)=\frac{1}{2k-1}\,a(k,0), \qquad m(k,r)=\frac{2(2k-3)}{k-2r}\,[m(k-1,r)-m(k-1,r-1)], \;\; k>2r.$$

With these results in hand we started programming. The program was written in C++ language using the apf-package for arbitrary precision arithmetics [11]. In several attempts and with iterated improvements of the program we were able to compute $T(r)$ for values $r \le 400$. Finally, we applied Newton approximation to approximate the roots of

$$p_n(e) := \sum_{r=0}^{n} [(1-16\,e^2)^r T(r)] - 0.5$$

Whereas this polynomial in e of degree $2n$ gives a value for e smaller than the correct value, we estimated coefficients $T^*(r)$ for higher degrees in such a way that the polynomial $p_n^*(e)$ has a value greater than $p_\infty(e)$. The simplest way to do so is setting $T^*(r+i)=T(r)$ where r is the highest index of a properly computed T-value. We took $p_n^*(e)$ as an upper bound for $p_\infty(e)$. The calculations showed a slow but sufficient convergence of $\lim\limits_{n\to\infty} p_n(e)$.

Our result: `0.06191 39544 739 ....`

#7 Clearly, we only have to solve a linear system of equations $Ax = b$ where b is the unit vector $(1,0,0,\ldots,0)$. The first element from the solution vector x is the $(1,1)$ entry of A^{-1} we are looking for. It is not hard to see that A is symmetric and diagonally dominant (see Figure 3). Furthermore, first calculations with $N \times N$ submatrices of A had shown that the matrix is positive definite. Two approaches generally come into account: i) *direct* methods and ii) *iterative* methods. First we tried to apply a CHOLESKY decomposition. But matrix A has $199,990,000$ elements below its main diagonal which means that memory exhausting problems arise if we store all elements of A. Since the well-known "fill-in" (replacement of zero elements with non-zero elements while the decomposition) occurs only in the envelope of the matrix, special storage schemes for sparse matrices were developed in the past decades.

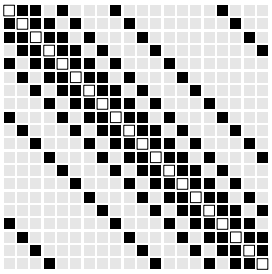

Figure 3. a) Structure of a 20×20 submatrix of A; the diagonal (hollow squares) contains all successive primes $2,3,5,\ldots$, entries marked by a full square are 1's, and gray squares are 0's.

The effective size of A is here $148,723,029$ elements which could be dropped to $100,659,904$ elements by application of the Reverse Cuthill-McKee algorithm [12]. A run on the Compaq yields a result with an accuracy of 34 decimal places.

Next we applied the iterative JACOBI method

$$x_i^{(k+1)} = \frac{1}{C_{ii}}\left(d_i - \sum_{j=1, j\ne i}^{n} C_{ij} x_j^{(k)}\right), \quad i=1,\ldots,n,$$

$$A = L + D + U,$$
$$C = -D^{-1}(L + U),$$
$$d = D^{-1}b.$$

This approach is much better than a direct method since there is no need to store all elements of matrix C (if one has all powers of two and all the primes in hand, a simple function call makes the element available). An implementation of the algorithm with use of the GMP library confirms the value obtained by the direct method.

Our result: `0.72507 83462 68401 ....`

#8 Three different approaches are applicable to solve this challenge: i) to find an analytical solution of the partial differential equation which satisfies the given initial and boundary conditions, ii) finite difference methods (FDM), or iii) finite element methods (FEM). However, the latter both have a significant drawback: Since we have a transient problem in which a time must be computed, the equidistant time-stepping scheme usually used in FDM or FEM codes have to be adapted to our purpose. Moreover, the development of well-working codes is a very time-consuming and difficult task.

Fortunately, the proposed problem is a *linear* boundary value problem. A standard method to solve this kind of problems are Fourier transforms, and indeed in [13], the solution for the heat transfer in a rectangular plate inside the region $[0, a] \times [0, b]$ can be found:

$$u(x, y, t) = \frac{4\pi}{ab^2} \sum_{m=1}^{\infty} \sum_{n=1}^{\infty} \frac{n}{\mu_{mn}} F_{\mathrm{s}}(m)(1 - \mathrm{e}^{-\mu_{mn}t}) \sin\left(\frac{m\pi x}{a}\right) \sin\left(\frac{n\pi y}{b}\right),$$

where $a = b = 2$ are the width and height of the plate, respectively, $\mu_{mn} = \pi^2 \left(\frac{m^2}{a^2} + \frac{n^2}{b^2}\right)$ and

$$F_{\mathrm{s}}(m) = \int_0^a u(\xi) \sin\left(\frac{m\pi\xi}{a}\right) \mathrm{d}\xi = 5 \int_0^2 \sin\left(\frac{m\pi\xi}{2}\right) \mathrm{d}\xi = \begin{cases} \frac{20}{m\pi} & \text{for } m \text{ odd} \\ 0 & \text{for } m \text{ even} \end{cases}$$

is the Fourier sine transform of the given boundary condition $u(\xi) = 5$ along the lower side of the square. For the centre of the plate at $(1, 1)$ we obtain with $m = 2k + 1$, $n = 2l + 1$:

$$\bar{u}(t) = u(1, 1, t) = 10 \sum_{k=0}^{\infty} \frac{(-1)^k}{2k+1} \sum_{l=0}^{\infty} \frac{(-1)^l (2l+1)}{\mu_{mn}} (1 - \mathrm{e}^{-\mu_{mn}t}).$$

It is surprising to get such a short formula for the temperature. However, it turned out that the inner series converges only very slowly. Fortunately, a closed form for the slowest convergent term is given in [5]:

$$\sum_{l=0}^{\infty} \frac{(-1)^l (2l+1)}{\mu_{mn}} = \frac{4}{\pi^2} \sum_{l=0}^{\infty} \frac{(-1)^l (2l+1)}{(2l+1)^2 + (2k+1)^2} = \frac{1}{\pi} \operatorname{sech}\left[\frac{\pi(2k+1)}{2}\right],$$

where $\operatorname{sech}(x) = (\cosh x)^{-1}$ is the hyperbolic secant. Thus we have

$$\begin{aligned} \bar{u}(t) = u(1, 1, t) &= 10 \sum_{k=0}^{\infty} \frac{(-1)^k}{2k+1} \left[\frac{1}{\pi} \operatorname{sech}\left(\pi k + \frac{\pi}{2}\right) - \sum_{l=0}^{\infty} \frac{(-1)^l (2l+1)}{\mu_{mn}} \mathrm{e}^{-\mu_{mn}t}\right] \\ &= \frac{5}{4} - \frac{40}{\pi^2} \sum_{k=0}^{\infty} \frac{(-1)^k}{2k+1} \sum_{l=0}^{\infty} \frac{(-1)^l (2l+1)}{(2k+1)^2 + (2l+1)^2} \mathrm{e}^{-\frac{\pi^2}{4}[(2k+1)^2 + (2l+1)^2]t}. \end{aligned} \tag{9}$$

Now both the remaining inner series and the outer series are sufficiently fast convergent. To be sure that the derived solution (9) is correct, several FEM input files have been generated with varying

mesh refinements using ordinary 4- and 8-node quadrilateral elements for the commercial finite element program ABAQUS [14]. A comparison between the calculated values for some selected times shows a good agreement within the fixed precision of the FE code. Finally, the application of a simple root finding algorithm like the bisection method to (9) yields the value of t for $u = 1$.

Our result: `0.42401 13870 33688 ...`

#9 As in the other problems it is necessary to apply some analytical manipulations to the given integral

$$I(\alpha) = [2 + \sin(10\alpha)] \int_0^2 x^\alpha \sin\left(\frac{\alpha}{2-x}\right) \mathrm{d}x. \tag{10}$$

First we substitute $y = \frac{\alpha}{2-x}$ into (10) and yield with $\beta = \frac{\alpha}{2}$:

$$I(\beta) = 2\beta \cdot 4^\beta \cdot [2 + \sin(20\beta)] \int_\beta^\infty \frac{\left(1 - \frac{\beta}{y}\right)^{2\beta}}{y^2} \sin y \, \mathrm{d}y.$$

If we introduce a new variable $z = y - \beta$ such that the lower boundary of the integral is 0, then the chances to find an analytical expression for the integral are good:

$$\begin{aligned} I(\beta) &= m(\beta) \int_0^\infty \frac{z^{2\beta}}{(z+\beta)^{2\beta+2}} \sin(z+\beta) \, \mathrm{d}z \\ &= m(\beta)\cos\beta \int_0^\infty \frac{z^{2\beta}}{(z+\beta)^{2\beta+2}} \sin z \, \mathrm{d}z + m(\beta)\sin\beta \int_0^\infty \frac{z^{2\beta}}{(z+\beta)^{2\beta+2}} \cos z \, \mathrm{d}z, \\ m(\beta) &= 2\beta \cdot 4^\beta \cdot [2 + \sin(20\beta)]. \end{aligned}$$

With the help of a computer algebra system we get

$$\begin{aligned} \int_0^\infty \frac{z^{2\beta}}{(z+\beta)^{2\beta+2}} \sin z \, \mathrm{d}z &= \frac{\pi}{2}\beta(\beta+1)\, {}_2F_3\left(2+\beta, \frac{3}{2}+\beta; 2, \frac{3}{2}, \frac{3}{2}; -\frac{\beta^2}{4}\right) - \frac{1}{2\,\Gamma(2+2\beta)} \times \\ &\sum_{k=0}^\infty \frac{(-\beta^2)^k \Gamma(2+2\beta+2k)}{(2k)!(2k+1)!} \left[\Psi(1+\beta+k) + \Psi\left(\frac{3}{2}+\beta+k\right)\right. \\ &\qquad \left. -\Psi\left(\frac{1}{2}+k\right) - 2\,\Psi(1+k) - \Psi\left(\frac{3}{2}+k\right) + 2\ln\beta - 2\ln 2\right], \\ \int_0^\infty \frac{z^{2\beta}}{(z+\beta)^{2\beta+2}} \cos z \, \mathrm{d}z &= -\frac{\pi}{2}\, {}_2F_3\left(1+\beta, \frac{3}{2}+\beta; 1, \frac{1}{2}, \frac{3}{2}; -\frac{\beta^2}{4}\right) + \frac{1}{\beta(1+2\beta)} - \frac{\beta}{2\,\Gamma(2+2\beta)} \times \\ &\sum_{k=0}^\infty \frac{(-\beta^2)^k \Gamma(3+2\beta+2k)}{(2k+1)!(2k+2)!)} \left[\Psi(2+\beta+k) + \Psi\left(\frac{3}{2}+\beta+k\right)\right. \\ &\qquad \left. -\Psi(1+k) - 2\,\Psi\left(\frac{3}{2}+k\right) - \Psi(2+k) + 2\ln\beta - 2\ln 2\right]. \end{aligned}$$

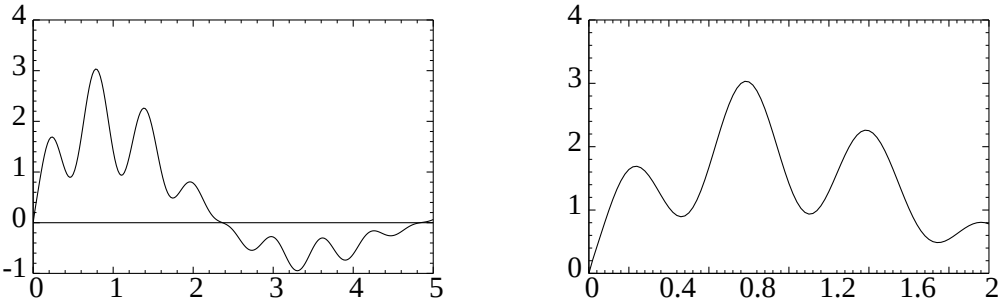

Figure 4. $I(\alpha)$ in the intervals $[0, 5]$ and $[0, 2]$.

The last formulae are well suited to calculate the integral $I(\alpha)$ with sufficient precision. Figure 4 shows that the maximum must be searched in the interval $\alpha = [0.78, 0.79]$. A simple algorithm that finds the maximum of a given function was used to get the desired value of α.

Our result: `0.78593 36743 50371 ...`.

#10 In this really hard problem all discontinuous approaches like random walk Monte-Carlo simulations or methods by Markov chains only lead to approximate solutions. So the key idea is to transform the problem into an analytical one. It is well-known that BROWNIAN motion of particles can be described via a linear diffusion equation of the form

$$c_t = c_{xx} + c_{yy} \tag{11}$$

for the concentration $c(x, y, t)$. Let the rectangle be the domain $0 \le x \le a$, $0 \le y \le b$. Then the boundary conditions are given by

$$c(0, y, t) = c(a, y, t) = c(x, 0, t) = c(x, b, t) = 0, \tag{12}$$

reflecting the fact that we have absorbing boundaries for all particles (or, in random walk terminology, we are only interested in *first passage times*). The initial condition at $t = 0$ represents a point source of particles in time and space:

$$c(x, y, t) = \delta\Big(x - \frac{a}{2}\Big) \cdot \delta\Big(y - \frac{b}{2}\Big) \cdot \delta(t). \tag{13}$$

Equations (11)–(13) have the solution

$$c(x, y, t) = \frac{4}{ab} \sum_{k=0}^{\infty} \sum_{l=0}^{\infty} (-1)^{k+l} \sin\left[\frac{(2k+1)\pi x}{a}\right] \sin\left[\frac{(2l+1)\pi y}{b}\right] \mathrm{e}^{-\mu_{kl} t},$$

$$\mu_{kl} = \pi^2 \left[\frac{(2k+1)^2}{a^2} + \frac{(2l+1)^2}{b^2}\right].$$

Integrating over t from 0 to ∞ we get

$$\bar{c}(x, y) = \frac{4}{ab} \sum_{k=0}^{\infty} \sum_{l=0}^{\infty} \frac{(-1)^{k+l}}{\mu_{kl}} \sin\left[\frac{(2k+1)\pi x}{a}\right] \sin\left[\frac{(2l+1)\pi y}{b}\right].$$

Now we have to calculate the fluxes $\frac{\partial \bar{c}}{\partial x}$ at $x = 0$ and $\frac{\partial \bar{c}}{\partial y}$ at $y = 0$ in order to determine the transition probabilities through the boundary of the rectangle by integration:

$$j_x = \left.\frac{\partial \bar{c}}{\partial x}\right|_{x=0} = \frac{4\pi}{a^2 b} \sum_{k=0}^{\infty} \sum_{l=0}^{\infty} \frac{(-1)^{k+l}(2k+1)}{\mu_{kl}} \sin\left[\frac{(2l+1)\pi y}{b}\right],$$

$$\Phi_x = \int_0^b j_x \,\mathrm{d}y = \frac{8}{a^2} \sum_{k=0}^{\infty} \sum_{l=0}^{\infty} \frac{(-1)^{k+l}(2k+1)}{\mu_{kl}(2l+1)}, \tag{14}$$

$$j_y = \left.\frac{\partial \bar{c}}{\partial y}\right|_{y=0} = \frac{4\pi}{ab^2} \sum_{k=0}^{\infty} \sum_{l=0}^{\infty} \frac{(-1)^{k+l}(2l+1)}{\mu_{kl}} \sin\left[\frac{(2k+1)\pi x}{a}\right],$$

$$\Phi_y = \int_0^a j_y \,\mathrm{d}x = \frac{8}{b^2} \sum_{k=0}^{\infty} \sum_{l=0}^{\infty} \frac{(-1)^{k+l}(2l+1)}{\mu_{kl}(2k+1)}. \tag{15}$$

The series in (14) and (15) show a very slow convergence because of a leading term of

$$\sum_{k=0}^{\infty} \frac{(-1)^k}{2k+1} = \frac{\pi}{4}.$$

Thus we write

$$\begin{aligned}
\Phi_x &= \frac{8b^2}{\pi^2 a^2} \sum_{k=0}^{\infty} (-1)^k (2k+1) \sum_{l=0}^{\infty} \frac{(-1)^l}{2l+1} \cdot \frac{1}{(2l+1)^2 + (2k+1)^2 \frac{b^2}{a^2}} \\
&= \frac{2}{\pi} \sum_{k=0}^{\infty} \frac{(-1)^k}{2k+1} \left\{ 1 - \operatorname{sech}\left[\frac{(2k+1)\pi b}{2a}\right] \right\} = \frac{1}{2} - \frac{2}{\pi} \sum_{k=0}^{\infty} \frac{(-1)^k}{2k+1} \operatorname{sech}\left[\frac{(2k+1)\pi b}{2a}\right], \\
\Phi_y &= \frac{8a^2}{\pi^2 b^2} \sum_{l=0}^{\infty} (-1)^l (2l+1) \sum_{k=0}^{\infty} \frac{(-1)^k}{2k+1} \cdot \frac{1}{(2k+1)^2 + (2l+1)^2 \frac{a^2}{b^2}} \\
&= \frac{2}{\pi} \sum_{l=0}^{\infty} \frac{(-1)^l}{2l+1} \left\{ 1 - \operatorname{sech}\left[\frac{(2l+1)\pi a}{2b}\right] \right\} = \frac{1}{2} - \frac{2}{\pi} \sum_{l=0}^{\infty} \frac{(-1)^l}{2l+1} \operatorname{sech}\left[\frac{(2l+1)\pi a}{2b}\right].
\end{aligned}$$

With $a = 1$ and $b = 10$ we have $\Phi_x \approx 0.49999980812$, $\Phi_y \approx 0.00000019188$ and $\Phi_x + \Phi_y = \frac{1}{2}$. Hence the claimed probability is $2\,\Phi_y$.

Our result: `0.00000 03837 58797 92512 ....`

Appendix

This appendix contains numerical results with more than the claimed ten decimal places as we were able to calculate them. The geometric mean of the number of all calculated digits is 375.

Problem	decimal places
#1	500
#2	50,000
#3	16
#4	5,120
#5	8
#6	12
#7	1,000
#8	500
#9	56
#10	10,000

#1: 0.32336 74316 77778 76139 93700 87952 17044 66510 46625 72546
96616 81036 44343 17903 37210 67289 44319 30370 46410 24513
80280 52733 12171 51284 33295 51003 34351 88538 14345 93384
99591 00496 20318 72302 43755 30585 03268 67826 08481 41436
21008 28454 47223 25433 19742 32895 62208 38303 60518 06138
17022 88759 27796 22388 94141 94808 89778 62776 56720 09163
63254 80930 06261 23688 30709 51561 72596 26227 10912 27961
81848 62649 25203 54425 80587 83035 12878 92271 35061 76972
50436 34273 24278 06269 81954 59935 93793 49487 03377 00399
38878 95844 49419 38532 52470 10512 63491 41166 86928 08429
(67050 54509) ...

#2: 0.99526 29194 43354 16089 03118 09426 72162 10294 66922 73415
43498 03208 85807 29861 79622 83063 20991 74981 89761 88760
30606 29732 61905 11722 08723 65706 97381 38277 98622 29860
17930 10235 84829 17122 47521 53389 37385 42120 28543 14982
29185 33578 56840 74227 09486 59177 51175 36045 44941 26248
54005 93772 07153 18743 54126 18678 93293 90653 70898 27732
15219 57086 46010 76407 55534 86493 64168 47633 88035 40535
49030 87704 95376 66182 65559 78453 72132 30110 76961 72983
75574 59902 66890 08663 19509 60381 27627 73049 01538 07015
23212 87830 24736 88796 83759 96054 00764 69555 64771 45943
66427 67584 15087 79345 61867 43650 04564 35356 43208 41981
83764 03078 60213 14625 04797 15434 91438 11458 14139 82234
60495 14807 86311 78024 51128 39236 83977 61067 47040 98230
74045 29162 80483 57668 20630 85421 99842 89850 90755 96185
77456 11949 21097 68564 53284 52690 99814 37840 16325 13898
46486 31178 39705 01991 74167 41223 21485 36149 61054 65417
17872 19163 80453 10684 05502 33617 15176 82498 85540 11350
88275 26387 25114 96817 40213 84003 62808 78771 61697 60955
12139 66061 43969 46628 67244 60371 41057 98856 78210 34723
72787 52029 21191 30561 71210 32945 34414 64029 29305 89953
26451 27108 41527 86068 44385 74138 90852 65360 88004 17274
85202 01196 30541 33721 41142 55841 44863 52865 55976 14172 ...

Up to 50,000 digits are given in a file named `SIAM2`.

#3: 1.27422 41528 20684 ...

#4: -3.30686 86474 75237 28007 61137 70898 51565 71664 82361 47628
82175 01293 08549 63091 99837 88829 50358 25488 07528 34991
86192 63089 11364 30717 13269 54791 83570 52753 54814 85323
49311 96660 55496 49177 09878 05562 73980 61774 08729 01417
02642 72392 22708 62073 34639 47755 14744 03455 59705 90007
10130 93048 45497 90041 25739 78770 06760 66240 31615 97724
61121 78932 59536 29405 39375 33901 07412 91488 67476 63311
69622 79417 36294 23962 92658 32107 91633 81013 16851 06326
78554 01990 04015 24711 71435 56882 34420 63752 16854 31767
30945 79094 25768 88235 10971 48504 85308 35209 86326 33998
35579 47849 15463 01045 47099 67946 73110 70004 30981 79105
93652 67442 94088 39748 63388 21903 64165 44051 72094 76651
68360 17535 35141 47576 01247 91643 62997 03563 53417 69389
57353 50598 63547 36421 22822 52868 88787 49695 50982 78981
43628 52139 04941 15341 24539 53852 69727 85125 72673 62553
18854 66044 43742 86187 36795 38628 38938 58689 97963 04777
94863 18537 16589 90522 50504 50767 23434 75949 68878 04395
24509 79673 71042 61919 98157 79909 79508 96729 78863 17173
39280 79948 61717 75182 54295 47302 47728 37610 15961 31982
38299 14040 69551 49745 64405 11842 54114 24061 12230 33234 ...

at $(x, y) =$

(-0.02440 30796 94375 17190 36133 08329 69288 59043 96181 85897
57489 60954 72354 29391 54937 62509 99868 76116 04664 44995 9 ...,
0.21061 24271 55355 77059 15911 00554 52553 52691 37074 21982
91755 01945 40772 19987 84780 78400 65302 59049 68645 29875 ...)

Up to 5,120 digits are given in a file named `SIAM4`.

#6: 0.06191 39544 739 ...

#7: 0.72507 83462 68401 16746 86877 19251 16096 88691 80594 47950
89578 78164 76920 77731 89994 59628 35735 92392 78647 82020
49710 76163 24699 84743 35581 69808 59303 87935 60189 41388
74208 37214 15508 59641 62181 96791 21149 36017 00800 57048
46219 85810 99579 54060 95529 48509 46641 19229 71401 89746
74330 48160 70329 55026 93570 26200 20117 88628 90337 89890
91223 65989 05590 67822 39653 40191 57226 34372 74683 27230
07731 62898 48083 69479 09805 79565 25947 69084 92494 87995
96275 51014 01300 57403 21928 64322 81619 67441 48402 61396
65434 15463 35545 62690 40316 01064 98652 55555 17247 44790
34089 21626 14255 40950 36824 53893 60417 99625 43288 33428
03134 52579 57011 29386 87781 22950 43428 83547 97817 26950
33922 43262 93136 65178 10625 10848 33477 04107 27409 87014
21149 00003 54211 03341 41030 57501 84974 60985 24612 68468
11559 96295 03342 81995 78332 86981 44926 59660 67316 08390
65151 29775 66488 82938 05732 89676 17441 94082 05711 06457
36367 12144 98379 45834 21880 20990 72709 29953 97904 53238
17281 04427 27016 09466 27634 96330 51069 05541 39223 03643
79341 41593 10904 31282 47494 86262 53154 03185 88558 72876
25911 70771 13391 63180 97110 92922 78220 27307 63631 64056
07253 24217 12766 58466 47536 88470 17625 47890 61707 73413
14873 08193 82986 64978 78555 13348 82961 68868 13041 73985 ...

#8: 0.42401 13870 33688 36386 00554 97491 78847 36916 11522 03308
52761 90704 19058 66929 35897 34609 72420 30869 13703 48256
90452 11017 74514 11730 18132 98199 14302 75904 59857 59542
15148 80912 34299 49498 93903 93273 51198 35355 99308 88037
60040 97427 67264 00097 89312 20432 83724 41732 18630 94141
10075 88475 32740 10288 02516 89598 61347 31654 33718 94006
41849 66328 34431 07341 87027 21434 41711 29235 53042 92356
48802 21242 56584 76721 48204 59947 90700 27485 85247 12734
92538 62679 12831 89545 97431 31916 32548 89121 79494 06135
25382 28776 15419 63761 88538 62966 68371 68954 37299 02585 ...

#9: 0.78593 36743 50371 45456 52439 86327 54558 29623 95459 06186
68175 6 ...

#10: 0.00000 03837 58797 92512 26103 40713 31862 04839 10079 30055
94072 50956 90300 22799 17343 66068 52743 27650 08428 45647
26991 01535 33872 77831 78038 31155 44660 24291 38550 88504
10344 13877 48160 87592 95735 79676 47773 07732 92951 74018
39843 03931 38016 47910 07249 51181 20094 15206 80531 95973
52138 09853 45213 13987 70077 65078 26197 31493 90177 57468
88998 84444 68098 63262 64599 20822 27863 11389 99322 39658
25814 54130 47046 19981 09019 02912 17606 45911 21308 90215
28990 60149 07236 37566 97817 52076 16848 06801 11749 56830
49448 04326 50358 23341 42694 66258 80372 82616 20508 82606
38809 53322 87999 04415 23303 44651 38660 46460 42111 24375
19753 86985 73260 64959 23857 38647 75109 18783 62605 26095
18090 30770 83878 24030 65485 92522 28160 74543 10473 22744
23855 15420 96778 70238 51279 87386 94200 19901 07441 27977
15216 62115 50202 82266 38041 67331 53869 36093 58255 39024
08902 17619 91649 72194 71784 63260 25863 37758 10741 91924
13974 68174 51581 81231 84320 22034 16063 89932 32591 30536
84127 73582 48845 31580 31073 18121 30696 08951 03874 17482
49482 77934 39666 78093 39607 25683 53941 09595 08124 06441
09257 59930 38188 87286 21944 46236 42574 99260 09229 47153
36328 5 ...

Up to 10,000 digits are given in a file named `SIAM10`.

References

1. http://swox.com/gmp/
2. http://mathworld.wolfram.com/LambertsW-Function.html
3. W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery: *Numerical Recipes in C*, 2nd ed., Cambridge University Press, 1992.
4. B. N. Datta: *Numerical Linear Algebra*, Brooks/Cole Publishing Company, Pacific Grove, 1995.
5. A. P. Prudnikov, Yu. A. Brychkov, O. I. Marichev: *Integrals and Series*, Elementary functions, Gordon and Breach Science Publ., 1992.
6. L. N. Trefethen: *Near-Circularity of the Error Curve In Complex Chebyshev Approximation*, J. Approx. Theory, **31**, 344–367 (1981).
7. Ching-Yih Tseng: *A Multiple-Exchange Algorithm for Complex Chebyshev Approximation by Polynomials on the Unit Circle*, SIAM J. Numer. Anal., **33**, no. 5, 2017–2049 (1996).
8. L. N. Trefethen: *Chebyshev Approximation by Polynomials in the Complex Plane*, Undergraduate thesis, Applied Mathematics Committee, Harvard College, May 1977.
9. R. N. Bhattacharya, E. C. Waymire: *Stochastic processes with applications*, John Wiley & Sons, Inc., New York, 1990.
10. W. Feller: *An Introduction to Probability Theory and Its Applications*, John Wiley & Sons, Inc., New York, 1968.
11. http://www.jjj.de/mtommila/apfloat/2.33/
12. J. A. George, J. Liu: *Computer Solution of Large Sparse Positive Definite Systems*, Prentice Hall, Englewood Cliffs, N. J., 1981.
13. I. N. Sneddon: *Fourier Transforms*, McGraw-Hill, Inc., New York, Toronto, London, 1951.
14. Hibbitt, Karlsson & Sorenson, Inc.: ABAQUS/Standard, version 6.1, 2000.

Acknowlegdements

We like to thank Prof. Andreas Engel, Otto-von-Guericke University, Magdeburg and Richard Metzler, University Würzburg for their helpful discussions.