# The 100-Digit Challenge Problems

**1.** Evaluate $\lim_{\epsilon \to 0} \int_{\epsilon}^{1} \frac{1}{x} \cos\left(\frac{\ln x}{x}\right) dx$.

**2.** A photon starts at $\left(\frac{1}{2}, \frac{1}{10}\right)$ and travels east with speed 1. If there are reflecting circles of radius $\frac{1}{3}$ centered at each integer lattice point, how far is the photon from the origin at time 10?

**3.** What is the norm of the following infinite matrix viewed as an operator on $\ell^2$, the space of infinite square-summable sequences:

$$\begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{4} & \frac{1}{7} & \frac{1}{11} & \cdots \\ \frac{1}{3} & \frac{1}{5} & \frac{1}{8} & \frac{1}{12} & \frac{1}{17} & \cdots \\ \frac{1}{6} & \frac{1}{9} & \frac{1}{13} & \frac{1}{18} & \frac{1}{24} & \cdots \\ \frac{1}{10} & \frac{1}{14} & \frac{1}{19} & \frac{1}{25} & \frac{1}{32} & \cdots \\ \frac{1}{15} & \frac{1}{20} & \frac{1}{26} & \frac{1}{33} & \frac{1}{41} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}?$$

**4.** What is the global minimum of
$\sin(60\, e^{y}) + \sin(70 \sin x) + \sin(\sin[80\, y]) - \sin[10\,(x+y)] + \frac{1}{4}\left(x^2 + y^2\right) + e^{\sin(50\, x)}$?

**5.** If $p(z)$ is the cubic polynomial that best approximates $1/\Gamma(z)$ on the unit disk in the sup-norm (i.e., worst case), what is the error in this approximation?

**6.** An unbiased random walk on the integer lattice returns to the origin with probability 1. For which $\epsilon$ is this probability 1/2, where $\epsilon$ biases the east-west portion of the walk (i.e., $\frac{1}{4} - \epsilon$, $\frac{1}{4} + \epsilon$, $\frac{1}{4}$, and $\frac{1}{4}$ are the east, west, north, and south probabilities)?

**7.** Let A be the $20\,000 \times 20\,000$ matrix having the primes $(2, 3, 5, \ldots, 224737)$ along the main diagonal, and with $a_{ij} = 1$ whenever $|i - j|$ is a power of 2, and 0 elsewhere. What is the upper left entry of the inverse of $A$?

**8.** When is the temperature at the center of a $2 \times 2$ square equal to 1, assuming the starting temperature is 0 everywhere except on one side, where it is 5, and the temperatures on the sides do not change? Use $u_t = u_{xx} + u_{yy}$ where $u(x, y, t)$ is the temperature at time t.

**9.** Which $\alpha$ in $[0, 5]$ maximizes $(2 + \sin(10\,\alpha)) \int_0^2 x^{\alpha} \sin\left(\frac{\alpha}{2-x}\right) dx$?

**10.** What is the probability that an infinitely small Brownian motion particle in the plane that starts at the center of a $10 \times 1$ rectangle hits one of the short sides before one of the long sides? "Brownian motion" means that the particle follows a 2-dimensional

random walk with infinitesimal step length until it hits the boundary of the rectangle.

## Brief Solutions

Here are solutions to the problems based on the work of the Macalester team of Danny Kaplan and Stan Wagon, with additional notes regarding better solutions when we know them. In most cases there are better solutions than ours to the 10-digit problem (it is arguable, of course, but our approach was the right one, or close to it, for #1, #2, #8, and #9). One hundred digits are given in all cases but one; thanks to F. Bornemann and J. Boersma's team) for the extra digits in cases we did not have them. *Mathematica* code giving a complete solution is included in all cases except #2, which, while not difficult, requires the most programming. As far as our own solutions go, we can say that we had 14 digits to all the problems, with 14 or 15 occurring on #3, #5, #7, and #10. Using our work and that of others, all problems except #3 yield 100 digits with no problem (when one uses the proper algorithm, which can be hard to find). Problem 3 is more difficult.

On the other hand, it is tricky to come up with a good measure of difficulty, as different people have different expertise. For our team, the last four problems to be solved were #8, #6, #10, and #5, in that order, with #5 being last (Trefethen called #5 "exceptionally difficult"; three of the teams scoring 99 missed the tenth digit on #5). The other six we could handle without much research or code, but the last four required library work or heavy programming.

As far as getting 500 digits in reasonable time, that can be done for #2, #4, and #8 by our methods; for #5, #6, #7, and #10 by others (and probably #1 and #9). But we must not get too caught up in the search for digits; the real point of this exercise is the search for algorithms. Note that all problems can be solved using machine precision, except #2, for which it is probably impossible to avoid high precision.

### 1 Divide and Conquer

The `Oscillatory` method to *Mathematica*'s `NIntegrate` solves this, once the integral is transformed using Lambert's $W$-function (also known as `ProductLog`) to $\int_0^\infty \frac{\cos u}{u\left(1+\frac{1}{W(u)}\right)}\,du$. Now the option is not needed. This method evaluates the integral between zeroes and extrapolates the partial sums to the limit. One can get more by summing an initial segment by hand and then using the method just described on the tail. Our technique can be used to obtain 60 digits. Bornemann, using fancier techniques to accelerate convergence, obtained 100 digits. Boersma and Jansen used contour integration to transform the integral to two integrals that can be easily computed numerically. The first 100 digits:

0.32336743167777876139937008795217044665104662572546966168103644343179033721067289443193037046410245 14

```
NIntegrate[Cos[u]/(u (1 + 1/ProductLog[u])), {u, 0, ∞}, WorkingPrecision → 20]
```

```
0.32336743167744626415
```

## 2 A Sea of Mirrors

Program the geometry and start with the initial conditions to, say, 50 digits. Then *Mathematica*'s automatic precision control will show how precision is lost (about 3 digits per reflection). Starting with 50 digits is enough to get the answer to 10 digits. We can get 500 digits easily. The answer is 0.99526291944335416089031180942672162102946692273415434980320885807298617962283063209917498189761876 0.

## 3 A Singular Problem

The value sought is the largest singular value of the matrix. Approximate the matrix by the upper-left $2048 \times 2048$ matrix and use *Mathematica*'s `SingularValues` function (which uses the QR algorithm) or use Matlab's `norm` function. Confidence in the answer is obtained by using extrapolation on the squence of values one gets by looking at all upper-left submatrices whose dimension is a power of 2. We got 13 digits. Bornemann obtained 20 digits. R. Strebel of ETH-Zurich obtained 40 digits: 1.2742241528212281882123406397250780994 72

```
f[i_, j_] := 2./(2 - i + i^2 - 3 j + 2 i j + j^2);
SingularValueDecomposition[Array[f, {2048, 2048}]][[2, 1, 1]] // InputForm
```

```
1.2742241522691704
```

Here is the extrapolation method. We look at 13 powers of two and then extrapolate. There are ways (using higher precision) to learn that the extrapolated result is likely correct to 13 digits. In fact, the extrapolated result below turns out to be correct when rounded to 15 digits.

```
mat[n_] := Array[f, {n, n}]
data = Table[{n, SingularValueDecomposition[mat[2^n]][[2, 1, 1]]}, {n, 0, 12}];
TableForm[data]
0     1.
1     1.18335
2     1.25254
3     1.27005
4     1.27353
5     1.27412
6     1.27421
7     1.27422
8     1.27422
9     1.27422
10    1.27422
11    1.27422
12    1.27422

NumericalMath`NSequenceLimit[Last /@ data] // InputForm
1.2742241528212164
```

## 4 How Low Can You Go?

It is not hard to see that the global minimum is at least as low as –3.24, and it then follows from the structure of the function (two of the six terms have easy-to-measure positive values; the sines are never less then –1 or, in one case, –0.84) that the true minimum lies inside the unit circle. Then we used a root-finding method to find all the critical points and choose the correct one. The root-finding method was developed by Wagon in 1996 for the `VisualDSolve` project in differential equations, and uses data from a contour plot to find all solutions to a pair of equations. We can get 500 digits easily. The answer to 100 digits is

–0.330686864747523728007611377089851565716648236147628821750129308549 6-.
309199837888295035825488075283499

Here is a solution that finds all the critical points by using the `FindAllCrossings` function just alluded to.

```
f[x_, y_] := (x^2 + y^2) / 4 + e^Sin[50 x] +
   Sin[60 e^y] + Sin[70 Sin[x]] + Sin[Sin[80 y]] - Sin[10 (x + y)];
f[{x_, y_}] := f[x, y];
fx = ∂x f[x, y]; fy = ∂y f[x, y];
Min[Flatten[Table[ f[x, y], {x, -1, 1, 0.01}, {y, -1, 1, 0.01}]]]
-3.24646
```

We now find all critical points (there are 2720 in the unit square), keeping any for which the function value is under –3.2. We used a collection of 64 subsquares of the

2×2 square around the origin, to save memory. And by increasing the resolution and watching the results, we gain evidence for the completeness of the count. We include here the complete code for `FindAllRoots2D` (from the `VisualDSolve` package), which finds all solutions to two equations in an interval. The resolution is controlled by the `PlotPoints` option which controls the resolution of the contour plot used to get started. We did not attempt to prove that a certain resolution was enough for the problem at hand, but we did examine images of the crossing contours and that convinced us that we had all the roots.

```
Options[FindAllRoots2D] = {ContourData → Automatic, PlotPoints → Automatic,
    MaxRecursion → Automatic, SeedsOnly → False, SameTest → Automatic};


FindAllRoots2D[funcs_, {x_, a_, b_}, {y_, c_, d_}, opts___] :=
 Module[{fZero, seeds, signs, fy},
  fy = Compile[{x, y}, Evaluate[funcs[[2]]]];
  cPlot = (ContourData /. {opts} /. Options[FindAllRoots2D]);
  If[cPlot === Automatic,
   cPlot = ContourPlot[funcs[[1]] == 0, {x, a - (b - a) / 97, b + (b - a) / 103},
       {y, c - (d - c) / 98, d + (d - c) / 102},
    Evaluate[FilterRules[{opts}, Options[ContourPlot]]]]];
  fZero = Cases[Normal[cPlot], Line[z_] :> z, Infinity];
  seeds = Flatten[((signs = Sign[Apply[fy, #1, {1}]];
   #1[[1 + Flatten[Position[
              Rest[signs * RotateRight[signs]], -1 | 0]]]]) & ) /@ fZero, 1];
  If[(SeedsOnly /. {opts} /. Options[FindAllRoots2D]) || seeds == {},
   seeds,
   Select[Union[
       ({x, y} /. FindRoot[{funcs[[1]], funcs[[2]]}, {x, #1[[1]]}, {y, #1[[2]]},
    Evaluate[FilterRules[{opts}, Options[FindRoot]]]] & ) /@ seeds,
       SameTest → (SameTest /. {opts} /. Options[FindAllRoots2D])],
    a ≤ #1[[1]] ≤ b && c ≤ #1[[2]] ≤ d & ]]];

cps =
  FindAllRoots2D[{fx, fy}, {x, -1, 1}, {y, -1, 1}, PlotPoints → 160] // Quiet;
Length[cps]
```

```
2720
```

```
lowest = Sort[cps, f[#1] < f[#2] &][[1]]
f[lowest]
```

```
{-0.0244031, 0.210612}
```

```
-3.30687
```

And now it is easy to zoom in to a more precise value.

```
FindMinValue[f[x, y], {x, -0.024}, {y, 0.211}, WorkingPrecision → 20]
```

```
-3.3068686474752372801
```

Here is an alternative solution using a combinatorial search technique that we learned about for problem 5. This is not the ideal way, but it gets the answer very quickly; the contour-based root-finding given above is better, but the root-finder (code not given here) takes a little programming. The `SimulatedAnnealing` setting to `Method` also works for this problem.

```
NMinimize[{f[x, y], x^2 + y^2 ≤ 1}, {x, y}, Method → "DifferentialEvolution"]
```

```
{-3.30687, {x → -0.0244031, y → 0.210612}}
```

Our methods described here do not provide proof of correctness. The Wolfram Research team developed a method using interval arithmetic that does provide such proof (it is in our book).

## 5 Lost in Hyperspace

The best algorithm we found for this (which is far from the best algorithm that exists!) is the differential evolution method, a type of genetic algorithm included in *Mathematica*'s `NMinimize` function. Of course, one must first get an efficient way to get the error for a given cubic. Several observations help get that quickly, the most important of which is that the maximum error occurs on the upper half of the unit circle.

The code that follows produces 15 digits (but with no proof that they are correct). Note that the problem did not specify that p(z) was to have real coefficients. The Eindhoven team's solution (link given earlier) provides a proof. We did find another method that reduced the problem from a 4-dimensional optimization to a 1-dimensional optimization of the results of a 3-dimensional optimization, and that technique was good enough to get us 15 digits.

But now I see that experts (Laurie, Bornemann, Jansen & Boersma) use the method of Chebyshev approximation, which is efficient and provides a proof of correctness. Jansen and Boersma obtained 100 digits (and their method yields 500 with no problem), which are:

0.2143352345904596394615264001847493961134072877895167080739349859709876760740545981694798847888305330590347563 3.

```
maxerror[{a_, b_, c_, d_}] := Max[Abs[a - b + c - d], (FindMaximum[t = e^(i θ);
        Abs[a t^3 + b t^2 + c t + d - 1/Gamma[t]],
        {θ, #1 - 0.03, #1 + 0.03}, PrecisionGoal → 12][[1]] &) /@ {1.40, 2.26}];

maxerror[{-0.603, 0.625, 1.02, 0.006}]
0.214865

DifferentialEvolution[n_, seeds_] :=
  Module[{best, current, children, vals, childval},
   current = Table[Random[Real, #] & /@ seeds, {n}];
   vals = maxerror /@ current;
   oldbest = Min[vals];
   Do[children =
     Table[{1, 0.4, -0.4}.current[[Table[Random[Integer, {1, n}], {3}]]], {n}];
    Do[If[(childval = maxerror[children[[j]]]) < vals[[j]],
      {current[[j]], vals[[j]]} = {children[[j]], childval}], {j, n}];
    If[Mod[i, 10] == 0, best = Min[vals];
     If[Abs[best - oldbest] < 10^-14, Break[], oldbest = best]], {i, 300}];
   First[Sort[Transpose[{vals, current}]]]];

SeedRandom[1];
DifferentialEvolution[60, {{-2, 2}, {-2, 2}, {-2, 2}, {-2, 2}}]
{0.214335, {-0.603343, 0.625212, 1.01976, 0.00554195}}
```

We tried to get more digits. The best cubic we found is

$$-0.603343220479925996\,z^3 + 0.6252119162842364\,z^2 +, \\ 1.01976185283468839\,z + 0.00554195066098559$$

for which the maximum error is 0.21433 52345 90459 64277. It seemed likely that 16 or 17 of these digits are correct. It turns out that 16 are; 17 if one rounds.

An earlier dimenson-reduction approach of ours is interesting and did get us 15 digits. First one observes that, when one is close, the error curve has local maxima at 1.40, 2.26, and $\pi$, where the first two are approximate and the last is exact. Moreover, the error at $\pi$ is $a - b + c - d$. So we can specify e, the error at $\pi$, and look at the family of cubics of the form $a\,z^3 + b\,z^2 + c\,z\, +\, a - b + c + e$. We can try to find the cubic in this 3-dimensional family for which the overall max-error is least. Then it is simply a matter of finding the value of e for which this abc-minimum value is least, and `FindMini-`

`mum` is up to the task on this 3-dimensional problem (and then on the 1-dimensional problem).

This approach can be viewed as a coordinate transformation of the 4-dimensional problem. Simply using `FindMinimum` on the raw 4-dimensional problem did not work well. Trying to lower the dimensions directly (fixing d, say, and optimizing a, b, c) does not work. A possible reason for this difference is that as we change e a little bit the overall shape of the curve does not change much; this is analogous to using Bézier polynomials as a design tool. This means that the location of the other two bumps does not change much, and the seed for that search is good. We store the locations found by the abc problem and use them for the next value of e, thus adding efficiency to the 3-dimensional search.

The solution by Boersma and Jansen of the Eindhoven team is noteworthy, as they found a lemma of Vidensky in a 1968 book by Smirnov and Lebedev [*Functions of a Complex Variable, Constructive Theory*, p. 450-452] that reduces the problem in quite a simple way to six simultaneous equations that can be solved by standard root-finding. Here is their code, which easily yields 500 digits.

```
w[a_, b_, c_, d_, z_] := 1/Gamma[z] - a - b z - c z^2 - d z^3;

det[u_, z_] := Det[{{1, z, z^2, z^3},
  {1, -1, 1, -1},
  {1, Conjugate[z], Conjugate[z]^2, Conjugate[z]^3},
  {1, Conjugate[u], Conjugate[u]^2, Conjugate[u]^3}}];

dw[z_, a_, b_, c_, d_] :=
  Im[z w[a, b, c, d, Conjugate[z]] (PolyGamma[z]/Gamma[z] + b + 2 c z + 3 d z^2)];

a - b + c - d /. FindRoot[{
   a - b + c - d == Abs[w[a, b, c, d, E^(I t1)]],
   a - b + c - d == Abs[w[a, b, c, d, E^(I t2)]],
   dw[E^(I t1), a, b, c, d] == 0,
   dw[E^(I t2), a, b, c, d] == 0,
   Arg[Conjugate[w[a, b, c, d, E^(I t1)]]/det[{E^(I t1), E^(I t2)}]] == 0,
   Arg[Conjugate[w[a, b, c, d, E^(I t2)]]/det[{E^(I t2), E^(I t1)}]] == 0},
  {a, {0.00555, 0.0059}}, {b, {1.0198, 1.0206}}, {c, {0.6252, 0.6261}},
  {d, {-0.6033, -0.602}}, {t1, {1.37, 1.40}}, {t2, {2.25, 2.26}},
  WorkingPrecision → 25, AccuracyGoal → 15]

{0.2143352345904596394615 26, 0.2143352345904596394615 26}
```

## 6 To Be Fair, It Must Be Biased

The return probability has the form $1 - \frac{1}{\sum_{n=0}^{\infty} p(n,\epsilon)}$ where $p\,(n,\ \epsilon)$ is the probability of return to the origin after $2\,n$ steps; this can be written in terms of binomial coefficients as $p(n,\epsilon) = 16^{-n}\binom{2\,n}{n}\sum_{k=0}^{n}\binom{n}{k}^2\left(1-16\,\epsilon^2\right)^k$. The sum can be estimated by using a partial sum or, for more digits, using an extrapolation technique to estimate the tail. Then standard root-finding techniques tell when the expression is 2. The answer is 0.0619139544739909428481752164732121769996387749983620760614672588599310102975961584590710564575208 7861.

There is also a closed form for this sum in terms of elliptic functions. Several people observed this. Here is how to use the elliptic function.

```
ϵ /. FindRoot[π Sqrt[2 + 16 ϵ^2 - 2 Sqrt[1 - 16 ϵ^2]] / EllipticK[2 Sqrt[1-16 ϵ^2] / Sqrt[1-16 ϵ^2 -1-8 ϵ^2]] == 2, {ϵ, 0.1}]
```

```
0.061914
```

And there is a well-known connection between this elliptic function and the very-fast-to-compute arithmetic-geometric mean. Thus, as observed by Bornemann, it is faster and more elegant to use the following, which gives the same result, and gives 500 digits in under a second.

```
ϵ /. FindRoot[
  Sqrt[2] ArithmeticGeometricMean[Sqrt[1 + 8 ϵ^2 - Sqrt[1 - 16 ϵ^2]], Sqrt[1 + 8 ϵ^2 + Sqrt[1 - 16 ϵ^2]]] ==
   1, {ϵ, 0.1}, WorkingPrecision → 500]
```

```
0.061913954473990942848175216473212176999638774998362076061467258859931010⋯
 297596158459071056457520878613716777621646035477035216576197862389207676 9⋯
 265210054246222936461060219831086664501115514448011690543835708233305223 5⋯
 585247093496178675886579320261925322981842755398323065953839669902970037 9⋯
 201450293339528637915513428413173819551547081604079876047944966252673904 0⋯
 704857090917636453642527587617501936819066435426651304107692758964293794 5⋯
 735889870031373692899494980937812909549989657672408260042853941 9
```

## 7 Inverting an Almost-Zero Matrix

The conjugate gradient method solves this in about 2000 iterations, and is easy to program. *Matlab* had this built-in in a form that could solve the problem. The answer is 0.72507834626840. The *Mathematica* code that follows uses compilation for speed. Bornemann obtained 100 digits. Here they are:

0.7250783462684011674686877192511609688691805944795089578781647692077731899945962835735923927864782020. D. Laurie reports that even Jacobi's iterative method is able to solve this.

```
diag[dim_] := diag[dim] = N[Prime[Range[dim]]]

mult = Compile[{{w, _Real, 1}},
    Module[{v = diag[Length[w]] w, d = Length[w]}, Do[v[[i]] +=
       Plus @@ w[[i + Join[2^Range[0,Log[2,d-i+1-0.1]], -2^Range[0,Log[2,i-0.1]]]]], {i, d}];
     v], {{diag[_], _Real, 1}}];

conjgrad = Compile[{{dim, _Integer}, {start, _Real, 1}, stop}, Module[
     {x = start, rold, r2, a, r = Prepend[Table[0., {dim - 1}], 1.], p, temp},
     p = r -= mult[x];
     r2 = Plus @@ r^2;
     Do[temp = mult[p];
      a = r2/p.temp;
      x += a p;
      rold = r2;
      r -= a temp;
      r2 = Plus @@ r^2;
      p = r + r2 p/rold,
       {stop}];
     x[[1]]], {{mult[_], _Real, 1}}];

conjgrad[20 000, Table[1, {20 000}], 3000]
0.725078
```

## 8 It Gets Hot, but When?

The standard technique of separation of variables and Fourier series works. Many texts explain how to use these techniques for the case that the boundary is held constant at 0 and an initial temperature is given for the interior, and for the case where the boundary is nonconstant and the steady-state solution is sought. It is not hard to combine these two methods to solve the problem at hand. The answer is:

0.42401138703368836379743366859325645124776209066427476219711249591331017695756369229707244229447701 12.

The following is a very terse solution, where some numerical work was used to deduce that the steady state solution at the center is exactly 5/4 (I discovered this numerically, but it is a special case of a known family of series (I. J. Zucker, 1984, see Problem 10),

and also follows from the symmetry of the problem. Only three terms of the series were used, which additional computation shows is adequate. Using more terms yields 500 digits easily. By using the explicit partial sum in the code and simplifying the equation a bit, the final expression is nicely simple (essentially a ninth-degree polynomial equation).

```
s = e^(-1/2 π^2 t);
t /. FindRoot[80 s^9 - 480 s^5 + 720 s == 9 π^2, {t, 1}, AccuracyGoal → 9]
```

```
0.424011
```

Here is the series that converges to 5/4 (the result alluded to above, with a multiplicative constant stripped off, so the sum is different).

$$\sum_{k=0}^{\infty} \frac{(-1)^k}{2k+1} \operatorname{sech}\left[\frac{\pi}{2}(2k+1)\right] = \frac{\pi}{8}$$

F. Bornemann observes that this series is a special case of a more general one which can be expressed in terms of the modular $\lambda$ elliptic function (see #10).

## 9 Surprise

The integral has a closed form in terms of the Meijer G function. Since routines to compute that are included in *Mathematica*, it is easy to use a standard optimization method to find the answer. We can get 50 digits, but beyond that ran into a problem with the G computation. Bornemann obtained 100 digits. They are:

0.78593367435037145456524398632754558296239545906186681758123180709891039714941236511677063376599449 53.

For the solution, we first discover the closed form, and then optimize it.

```
obj = Simplify[(2 + Sin[10 α]) ∫_(1/2)^∞ (2 u - 1)^α Sin[α u]/u^(2+α) du, α > 0]
```

$$4\sqrt{\pi}\,\text{Gamma}[\alpha]\;\text{MeijerG}\left[\left\{\{\},\left\{\frac{3+\alpha}{2},\frac{4+\alpha}{2}\right\}\right\},\left\{\left\{1,1,\frac{3}{2}\right\},\left\{\frac{1}{2}\right\}\right\},\frac{\alpha^2}{16}\right](2+\text{Sin}[10\alpha])$$

```
FindMaximum[obj, {α, 78/100, 79/100}, PrecisionGoal → 12][[2, 1, 2]]
```

```
0.785934
```

## 10 Think Outside the Box

We found that a very similar problem could be easily solved and that it agrees with the given problem to within $10^{-21}$. Since the answer is about $10^{-7}$, this is enough. The

variation is to consider the region with no right boundary and ask for the probability that a particle hits the left before hitting the top or bottom. This turns out to have the simple and exact answer: $\frac{4}{\pi}\arctan\left(e^{-5\pi}\right)$. Doubling this, and doing some very rough estimation on the discrepancy between our variation and the given problem, leads to the slightly better formula $\frac{4}{\pi}\arctan\left(e^{-5\pi}\right)\left(1+\frac{1}{1+\frac{4}{\pi}\arctan\left(e^{-10\pi}\right)}\right)$, which gives the answer to the problem posed to 14 digits. The answer to 100 digits is:
0.00000003837587979251226103407133186204839100793005594072509569030022799173436606852743276500842845647269910.

So, given the mathematical work we did, one can easily get 14 digits as follows.

```
N[4/π ArcTan[e^(-5π)] (1 + 1/(1 + 4/π ArcTan[e^(-10π)]))]
```

$3.83759\times10^{-7}$

Or 13 digits from:

```
N[8/π ArcTan[e^(-5π)], 13]
```

$3.837587979251\times10^{-7}$

```
N[8/π ArcTan[e^(-5π)], 20]
```

$3.8375879792513132589\times10^{-7}$

Because the argument is small, one gets the same digits even if the arctangent is ignored. We were lucky, since our method would fail if the rectangle was closer to a square!

Bornemann showed that the exact answer to the given question is $\frac{4}{\pi}\sum_{k=0}^{\infty}\frac{(-1)^k}{2k+1}\,\mathrm{sech}[5\pi(2k+1)]$, which is very similar to the steady-state series in #8. One term of this series gives 14 digits. Moreover, Bornemann observed that this series equals $\frac{8}{\pi}\sum_{k=0}^{\infty}(-1)^k\arctan\left(e^{-(2k+1)5\pi}\right)$ (the first term of which is our arctan approximation) and it is well known (Borwein & Borwein, *Pi and the AGM*, §3.2, exercise 12) that these series equal $\frac{2}{\pi}\arccos\sqrt{\lambda(i/10)}$, where $\lambda$ is the modular $\lambda$ elliptic function. This formula is fast, taking only a small fraction of a second to get 500 digits.

```
N[2/π ArcCos[Sqrt[ModularLambda[i/10]]], 20]
```

$3.8375879792512261034\times10^{-7}$