Two Algorithms for the Computation of π

Benjamin Dreyer

November 16, 2022

Contents

1 Introduction

This paper introduces two novel (to the knowledge of the author) algorithms to calculate the circle constant π.

For their relation to the volumes of hyperballs which will become apparent in this paper, we call the first algorithm the *Dimension Specific Algorithm (DSA)* and the second algorithm the *Trans-Dimensional Algorithm (TDA)*.

Both algorithms have a runtime complexity of $O(n(\log n)^3)$ and a space complexity of $O(n)$ to calculate π to a precision of n digits. Regarding their suitability for practical applications, the TDA is most likely always superior to the DSA because it requires less memory and is less subject to rounding errors.

The algorithms are unlikely to be fast enough to break world records, however they have certain advantages in their simplicity.

We cannot provide sufficient proof that the algorithms are correct. They have been discovered by finding regular patterns in certain series known to converge to π. However both algorithms have been used to calculate π to hundreds of millions of hexadecimal digits which gives confidence in the conjecture that they are indeed correct.

For lack of formal proof, this paper will give a brief summary of how the algorithms were discovered, reducing the gaps in reasoning to one conjecture per algorithm. We will also discuss the error boundaries for approximations of π.

We will touch upon some details about implementation strategies, but not cover them in depth. Generally, the methods to implement our algorithms efficiently are commonly known and not the core subject of this paper.

This paper is intended to be published alongside an example implementation at `https://github.com/Shakatir/picalc`.

1.1 Summary of the Dimension Specific Algorithm

The DSA can be summarized as follows:

Algorithm 1 (Dimension Specific Algorithm)**.** Given integers $n > 0$ and $k \geq 0$ compute

$$f_n(k) = 2 \times \frac{(2n)!!}{(2n-1)!!} \times \sum_{i=0}^{k} \frac{(2i-1)!!}{(2i+1)!} \prod_{j=0}^{i-1} (2j+1-2n).$$

We will use the convention $0!! = (-1)!! = 0^0 = 1$ throughout this paper.

We call n the *dimension* of f_n. As a shorthand we write $f_n(\infty) = \lim_{k\to\infty} f_n(k)$.

Since the techniques to compute these kinds of series are widely known, we won't go into detail about that here.

We make observations about the convergence and the complexity of the DSA which we summarize in theorem 2.

Theorem 2 (Convergence of DSA)**.** *The DSA has the following properties:*

1. *For every integer n there is a real number x such that $f_n(\infty) = x$.*
2. *For $n \leq k$ the following error boundaries hold:*

$$\frac{1}{(k+1)(2k+3)} \times \binom{k}{n}^{-1} < (-1)^n \left(f_n(\infty) - f_n(k)\right) < \frac{2k+4}{2k+3} \times \binom{k+1}{n}^{-1}.$$

Note: $\binom{a}{b}$ denotes the binomial coefficient "a choose b."

Notably, this theorem holds even if the algorithm does not converge to π. The only point of uncertainty is summarized in conjecture 3.

Conjecture 3 (Correctness of DSA)**.** *For any positive integer n, the following equation holds:*

$$\pi = 2 \times \frac{(2n)!!}{(2n-1)!!} \times \sum_{i=0}^{\infty} \frac{(2i-1)!!}{(2i+1)!} \prod_{j=0}^{i-1} (2j+1-2n).$$

In practical terms, one can compute n hexadecimal digits of π by computing the value $f_{2n}(4n)$. Calculating $f_n(k)$ to a precision of $O(n)$ digits takes $O(m(\log m)^3)$ time and $O(m)$ space on a proper implementation (where $m = max(n,k)$). Calculating the exact result without rounding takes $O(m \log m)$ space but does not hurt the time complexity.

1.2 Summary of the Trans-Dimensional Algorithm

The TDA can be described as follows:

Algorithm 4 (Trans-Dimensional Algorithm)**.** Given the integer $n \geq 0$, compute

$$a_n = 2 + \sum_{i=0}^{n-1} \left(\frac{-1}{4}\right)^i \left(\frac{2}{2i+1} - \frac{2(i+1)}{(2i+1)(4i+3)} - \frac{1}{2(4i+5)}\right).$$

This algorithm is just as easy to implement as the DSA. It should be obvious from looking at the series that it converges at a speed of roughly 2 bits per iteration, so it would be sufficient to conjecture that $a_\infty = \pi$. However we make a stronger conjecture that depends on the DSA instead.

Conjecture 5 (Correctness of TDA)**.** *For any positive integer n, the result a_n of the TDA is equal to the result $f_n(2n)$ of the DSA.*

While this conjecture has an additional dependency on the DSA, it gives us a few more properties for free. Theorem 2 assures roughly 2 bits of precision per iteration as well, and on top of that promises us that a_n alternates between upper and lower bounds of π according to the inequality:

$$a_{2n} < \pi < a_{2n+1}.$$

The runtime complexity of $O(n(\log n)^3)$ and the space complexity of $O(n)$ also trivially follow from this relationship. However, the TDA can be implemented with a smaller memory footprint and less rounding errors.

2 Construction of the Dimension Specific Algorithm

We will now go through the steps that led to the discovery of the DSA.

The original idea was to calculate π by finding a solution to the integral

$$\frac{\pi}{2} = \int_{-1}^{1} \sqrt{1-t^2}\, dt.$$

This integral has (obviously) no closed form solution, but the goal was to derive from it a series, that converges to π instead.

We're going to skip over the first step in which we derive a series for this particular integral and go straight to the generalization to calculate the volume of hyperballs in arbitrary dimensions. The method used to solve it remains the same as in the special case.

We introduce a function that defines hyperballs recursively. For $n \in \mathbb{N}_0$ we define the function $V_n : \mathbb{R} \to \mathbb{R}$ such that

$$V_0(r) = 1 \quad \text{and} \quad V_n(r) = \int_{-r}^{r} V_{n-1}\left(\sqrt{r^2 - t^2}\right) dt.$$

This is a commonly known definition that interprets an n-ball as a stack of infinitely many infinitely thin layers of $(n-1)$-balls.

Through induction we can show that $V_n(r) = V_n(1) \times r^n$. Thus calculating the volume of an n-ball can be reduced to the problem of $V_n(1)$ which only needs to be done once per n.

We write $\pi_n = V_n(1)$. Then our new problem becomes finding a solution to the recursive formula

$$\pi_0 = 1 \quad \text{and} \quad \pi_{n+1} = \pi_n \times \int_{-1}^{1} (1 - t^2)^{n/2}\, dt$$

which is more specifically the question how to solve the integral

$$\int_{-1}^{1} (1 - t^2)^{n/2}\, dt.$$

At this point one might get tempted into distinguishing cases since for even n this integral is almost trivially solvable and yields a rational result whereas for odd n, it cannot have a closed solution.

But this is where we make an unusual choice. Without distinguishing cases we find the Taylor series of the function we're trying to integrate.

Calculating the derivatives is tedious work, but it can be automated. We will not go into the details of said automation and jump straight to the conjecture that sprang from a careful look at the results.

Conjecture 6. *For $n \in \mathbb{N}_0$ and $x \in [-1, 1]$ the following equation holds:*

$$(1 - t^2)^{n/2} = \sum_{i=0}^{\infty} \left(t^{2i} \times \frac{(2i-1)!!}{(2i)!} \prod_{j=0}^{i-1} (2j - n) \right).$$

Note how the term $(2j - n)$ will eventually turn to 0 for some j if n is even. This causes almost all terms of the Taylor series to vanish, effectively turning it into a finite polynomial.

Given this conjecture we can then solve the above problem rather simply.

Corollary 7. *Let π_n denote the volume of an n-dimensional hyperball, then the following recursive formula holds:*

$$\pi_0 = 1 \quad \textit{and} \quad \pi_{n+1} = 2\pi_n \times \sum_{i=0}^{\infty} \frac{(2i-1)!!}{(2i+1)!} \prod_{j=0}^{i-1} (2j-n).$$

Remark. Outside of its purpose in this paper, it is noteworthy that this recursive formula (provided the conjecture is true) unifies steps from odd n to even and vice-versa without distinguishing cases. It shows how the steps from $\pi_0 = 1$ to $\pi_1 = 2$ to $\pi_2 = \pi = 3.1415\ldots$ to $\pi_3 = \frac{4}{3}\pi$ and beyond are all part of the same continuous relationship, even though their ratios alternate between rational and transcendental.

Given this result, one might assume that we have already found a formula for π. We can calculate

$$\pi = \pi_2 = 4 \times \sum_{i=0}^{\infty} \frac{(2i-1)!!}{(2i+1)!} \prod_{j=0}^{i-1} (2j-1).$$

However once you do that, you will be confronted with the bitter realization that the number of digits of π computed grows *logarithmically* with the number of iterations. That is not a good result to put it lightly. However, if you implement the series for higher dimensions, you will notice that it converges much quicker.

We will now make use of two well established formulas:

Theorem 8 (Volumes of Hyperballs)**.** *For $n \in \mathbb{N}_0$ the following equations hold:*

$$\pi_{2n} = \frac{\pi^n}{n!}, \qquad\qquad \pi_{2n+1} = \frac{2^{n+1} \times \pi^n}{(2n+1)!!}.$$

We can use these formulas to calculate the ratio

$$\frac{\pi_{2n}}{\pi_{2n-1}} = \frac{(2n-1)!!}{(2n)!!}\pi.$$

But we already have another formula for said ratio. And this allows us to finally arrive at the formula underlying the Dimension Specific Algorithm.

Corollary 9. *For any $n \in \mathbb{N}$:*

$$\begin{aligned} \pi &= 2 \times \frac{(2n)!!}{(2n-1)!!} \times \sum_{i=0}^{\infty} \frac{(2i-1)!!}{(2i+1)!} \prod_{j=0}^{i-1} (2j+1-2n) \\ &= 2 \times \prod_{i=1}^{n} \frac{2i}{2i-1} \times \sum_{i=0}^{\infty} \prod_{j=0}^{i-1} \frac{(2j+1)(2j+1-2n)}{(2j+2)(2j+3)}. \end{aligned}$$

If conjecture 6 is correct, this should serve if not as rigorous proof, at least as a convincing argument that the conjecture 3 and therefore the Dimension Specific Algorithm in general is correct.

Since the algorithm sprang from the relationship between the volumes of hyperballs in specific dimensions, we chose to call it the Dimension Specific Algorithm.

3 Construction of the Trans-Dimensional Algorithm

The DSA has two flaws:

1. Rather than being a single series that is suitable for all precisions, it is a family of series, each of which is suitable for its own range of precision.
2. For $k < n$ the result $f_n(k)$ can grow very large which hurts memory usage, makes rounding errors more severe and makes intermediate results virtually useless.

These issues could be resolved if we found some efficient way to go from $f_n(mn)$ to $f_{n+1}(m(n+1))$ for some positive integer m. Since this approach would allow us to jump across dimensions in the DSA we call the resulting algorithm Trans-Dimensional Algorithm.

After calculating the results of the DSA for several dimensions and iterations, another pattern emerged for which we unfortunately have no proof.

Conjecture 10 (Relation between Dimensions of DSA)**.** *For all $n \in \mathbb{N}$ the following identities hold for results of the DSA:*

$$f_{n+1}(n) = f_n(n) + \frac{2 \times (-1)^n}{(2n+1)}, \qquad f_{n+1}(2n) = f_n(2n) + \frac{(-1)^n}{2^{2n-1} \times (2n+1)}$$

If there exist similar formulas for other values of m, they are likely a lot less obvious and more complicated. It makes intuitive sense—given the structure of the DSA—that $f_n(n)$ and $f_n(2n)$ are two unique points of stability and that beyond that, no simple pattern exists.

From this relation we can then derive the recursive relationship:

$$\begin{aligned} f_{n+1}(2n+2) = {} & f_n(2n) + \frac{(-1)^n}{2^{2n-1}(2n+1)} \\ & + 2 \times \frac{(2n+2)!!}{(2n+1)!!} \times \frac{(4n+1)!!}{(4n+3)!} \times \prod_{j=0}^{2n} (2j+1-2n) \\ & + 2 \times \frac{(2n+2)!!}{(2n+1)!!} \times \frac{(4n+3)!!}{(4n+5)!} \times \prod_{j=0}^{2n+1} (2j+1-2n). \end{aligned}$$

Now we substitute each summand:

$$\begin{aligned}
b_n &= \frac{(-1)^n}{2^{2n-1}(2n+1)},\\
c_n &= 2 \times \frac{(2n+2)!!}{(2n+1)!!} \times \frac{(4n+1)!!}{(4n+3)!} \times \prod_{j=0}^{2n}(2j+1-2n)\\
&= \frac{(-1)^{n+1}(n+1)}{2^{2n-1}(2n+1)(4n+3)},\\
d_n &= 2 \times \frac{(2n+2)!!}{(2n+1)!!} \times \frac{(4n+3)!!}{(4n+5)!} \times \prod_{j=0}^{2n+1}(2j+1-2n)\\
&= \frac{(-1)^{n+1}}{2^{n+1}(4n+5)}.
\end{aligned}$$

So that we can then define the ultimate series

$$a_1 = f_1(2) = \frac{97}{30}, \qquad\qquad a_{n+1} = a_n + b_n + c_n + d_n.$$

And while there is no definition for $f_0(0)$, we can still extend the recursive formula such that $a_0 = 2$.

Now we turn this into its closed form:

$$\begin{aligned}
a_n &= a_0 + \sum_{i=0}^{n-1}(b_i + c_i + d_i)\\
&= 2 + \sum_{i=0}^{n-1}\left(\frac{-1}{4}\right)^i \left(\frac{2}{2i+1} - \frac{2(i+1)}{(2i+1)(4i+3)} - \frac{1}{2(4i+5)}\right).
\end{aligned}$$

And thus we have constructed the Trans-Dimensional Algorithm. It is only a small step from conjecture 10 to conjecture 5. They are essentially just rephrasing each other and therefore equivalent.

It is also possible to create an analogous algorithm that steps from $f_n(n)$ to $f_{n+1}(n+1)$ which theorem 2 assures us will also converge to π, but very slowly.

4 Convergence Behavior

We are now going to prove theorem 2. Throughout several iterations of attempting to write this paper, this part always proved both straight-forward and extremely tedious as we will juggle a lot of terms around.

For integers $n > 0$ and $k \geq 0$ we define

$$c_{n,k} = \sum_{i=k}^{\infty}\prod_{j=k}^{i-1}\frac{(2j+1)(2j+1-2n)}{(2j+2)(2j+3)},$$

which fulfills the recursive equation

$$c_{n,k} = 1 + \frac{(2k+1)(2k+1-2n)}{(2k+2)(2k+3)} c_{n,k+1}.$$

Obviously, this means that

$$f_n(k) = 2 \times \frac{(2n)!!}{(2n-1)!!} \times c_{n,0},$$

but also less obviously

$$f_n(\infty) - f_n(k) = 2 \times \frac{(2n)!!}{(2n-1)!!} \times \frac{(2k+1)!!}{(2k+3)!} \times \prod_{j=0}^{k} (2j+1-2n) \times c_{n,k+1}.$$

This happens to be the quantity which we are after in this section.

Before we determine error boundaries for the DSA itself, we first need to determine error boundaries for $c_{n,k}$.

For $k \geq n$ we know that $c_{n,k} > 0$ since there are no more negative factors in any of its summands. With this in mind, we can derive the inequality

$$\begin{aligned} c_{n,k} &= 1 + \frac{(2k+1)(2k+1-2n)}{(2k+2)(2k+3)} c_{n,k+1} < 1 + \frac{(2k+1)(2k-1)}{(2k+2)(2k+2)} c_{n,k+1} \\ &= 1 + \frac{4k^2-1}{(2k+2)^2} c_{n,k+1} < 1 + \frac{4k^2}{(2k+2)^2} c_{n,k+1} = 1 + \frac{k^2}{(k+1)^2} c_{n,k+1}. \end{aligned}$$

Now we can apply this inequality recursively.

$$\begin{aligned} c_{n,k} &< 1 + \frac{k^2}{(k+1)^2} c_{n,k+1} < 1 + \frac{k^2}{(k+1)^2} + \frac{k^2}{(k+2)^2} c_{n,k+2} \\ &< \ldots \\ &< 1 + k^2 \sum_{i=k+1}^{\infty} \frac{1}{i^2} < 1 + k^2 \sum_{i=k+1}^{\infty} \left(\frac{1}{i-1} - \frac{1}{i} \right) = 1 + \frac{k^2}{k} = 1 + k. \end{aligned}$$

Thus, we have found suitable error bounds for $c_{n,k}$

$$1 < c_{n,k} < 1 + k.$$

We also know for $k \geq n$:

$$\prod_{j=0}^{k} (2j+1-2n) = (-1)^n \times (2n-1)!! \times (2k+1-2n)!!$$

With these bounds in place, we can now look at the difference we actually want to estimate:

$$\begin{aligned}
f_n(\infty) - f_n(k) &= 2 \times \frac{(2n)!!}{(2n-1)!!} \times \frac{(2k+1)!!}{(2k+3)!} \times \prod_{j=0}^{k}(2j+1-2n) \times c_{n,k+1} \\
&= 2 \times \frac{(2n)!!}{(2n-1)!!} \times \frac{(2k+1)!!}{(2k+3)!} \times (-1)^n \times (2n-1)!! \times (2k+1-2n)!! \times c_{n,k+1} \\
&= (-1)^n \times \frac{2 \times (2n)!! \times (2k+1)!! \times (2k+1-2n)!!}{(2k+3)!} \times c_{n,k+1}
\end{aligned}$$

Before we continue, let's get the obvious out of the way: the sign of the difference $f_n(\infty) - f_n(k)$ is $(-1)^n$ as stated in theorem 2.

From now on, we will only look at the absolute difference.

$$\begin{aligned}
|f_n(\infty) - f_n(k)| &= \frac{2 \times (2n)!! \times (2k+1)!! \times (2k+1-2n)!!}{(2k+3)!} \times c_{n,k+1} \\
&= \frac{2 \times (2n)!! \times (2k+2)! \times (2k+1-2n)!!}{(2k+3)! \times (2k+2)!!} \times c_{n,k+1} \\
&= \frac{2 \times 2^n \times n! \times (2k+2)! \times (2k+1-2n)!!}{(2k+3)! \times 2^{k+1} \times (k+1)!} \times c_{n,k+1} \\
&= \frac{2^{n-k}}{2k+3} \times \frac{n! \times (2k+1-2n)!!}{(k+1)!} \times c_{n,k+1}.
\end{aligned}$$

Now we get for the upper bound:

$$\begin{aligned}
|f_n(\infty) - f_n(k)| &< \frac{2^{n-k}}{2k+3} \times \frac{n! \times (2k+2-2n)!!}{(k+1)!} \times (k+2) \\
&= \frac{2k+4}{2k+3} \times \frac{n! \times (k+1-n)!}{(k+1)!} \\
&= \frac{2k+4}{2k+3} \times \binom{k+1}{n}^{-1}.
\end{aligned}$$

And for the lower bound:

$$\begin{aligned}
|f_n(\infty) - f_n(k)| &> \frac{2^{n-k}}{2k+3} \times \frac{n! \times (2k-2n)!!}{(k+1)!} \\
&= \frac{1}{(k+1)(2k+3)} \times \frac{n! \times (k-n)!}{k!} \\
&= \frac{1}{(k+1)(2k+3)} \times \binom{k}{n}^{-1}.
\end{aligned}$$

And so we arrive at the final inequality:

$$\frac{1}{(k+1)(2k+3)} \times \binom{k}{n}^{-1} \; < \; |f_n(\infty) - f_n(k)| \; < \; \frac{2k+4}{2k+3} \times \binom{k+1}{n}^{-1}.$$

A more careful analysis might improve these boundaries further, but not by a lot. The boundaries we calculated already only differ by a factor of $O(k^2)$. An improvement to $O(k)$ seems feasible.

5 The Algorithms in Practice

To get a better understanding of the convergence behavior of the algorithms, consider table 1 which lists the first iterations of the DSA for select dimensions and the first iterations of the TDA.

k	$f_1(k)$	$f_5(k)$	$f_{10}(k)$	$f_{25}(k)$	a_k
0	4.000 000	8.126 984 127	11.350 927 710	17.813	2.000 000 000
1	3.333 333	−4.063 492 062	−24.593 676 704	−127.662	3.233 333 333
2	3.233 333	8.736 507 937	67.065 064 554	897.943	3.128 174 603
3	3.197 619	1.117 460 317	−96.611 259 122	−4 596.370	3.143 952 575
4	3.180 258	3.339 682 540	110.257 427 746	18 372.912	3.141 138 325
5	3.170 315	3.157 864 358	−75.924 390 435	−58 678.591	3.141 684 938
6	3.164 005	3.145 043 845	42.229 455 718	153 213.044	3.141 573 232
7	3.159 708	3.142 662 893	−8.970 544 281	−332 119.701	3.141 596 845
8	3.156 627	3.142 006 380	5.147 102 777	604 643.869	3.141 591 732
9	3.154 330	3.141 777 944	3.041 839 619	−931 977.075	3.141 592 859
10	3.152 564	3.141 684 938	3.137 077 714	1 222 950.868	3.141 592 607
11	3.151 171	3.141 642 479	3.141 030 283	−1 370 628.416	3.141 592 664
12	3.150 050	3.141 621 320	3.141 484 829	1 313 726.143	3.141 592 651
13	3.149 132	3.141 610 018	3.141 565 766	−1 076 190.665	3.141 592 654
14	3.148 368	3.141 603 628	3.141 584 605	751 565.909	3.141 592 653
15	3.147 725	3.141 599 843	3.141 589 892	−445 319.847	3.141 592 654
16	3.147 178	3.141 597 509	3.141 591 600	222 261.317	3.141 592 654
17	3.146 708	3.141 596 021	3.141 592 215	−92 455.517	3.141 592 654
18	3.146 300	3.141 595 043	3.141 592 458	31 588.280	3.141 592 654
19	3.145 943	3.141 594 384	3.141 592 561	−8 671.548	3.141 592 654
20	3.145 630	3.141 593 930	3.141 592 607	1 859.833	3.141 592 654
21	3.145 352	3.141 593 610	3.141 592 629	−291.927	3.141 592 654
22	3.145 105	3.141 593 380	3.141 592 640	35.183	3.141 592 654
23	3.144 884	3.141 593 213	3.141 592 646	1.141	3.141 592 654
24	3.144 685	3.141 593 090	3.141 592 649	3.182	3.141 592 654
25	3.144 505	3.141 592 997	3.141 592 651	3.142	3.141 592 654

Table 1: Various results of the DSA and TDA

What catches the eye are the early values for the DSA in higher dimensions. They don't converge to π immediately, but first alternate between extremely large positive and negative values before they finally begin to close in onto the actual limit. If one were to see the first million terms of f_{10^7} without context, they would be led to believe that the series simply diverges.

This is the big flaw of the DSA. One has to choose the dimension based on the precision they want to achieve. The value $f_{10}(20)$ already reaches a decent precision, but at the cost of $f_{10}(5)$ being complete garbage. And while f_1 doesn't produce such garbage values, it converges so slowly that even $f_1(25)$ is still a worse approximation of π than $^{22}/_3$. And on the opposite end, the DSA converges decently fast for a while, but for any dimension, the convergence will eventually taper off and the number of digits computed becomes asymptotically logarithmic in the number of iterations. Choosing the right dimension is crucial. If it is too high, the algorithm will take excessively long to converge to a usable result, and if it is too low, the same applies. There is a sweet spot for the rate of convergence around $f_n(2n)$. And this makes the TDA so good: it simply rides this wave of good convergence across dimensions to any arbitrary precision.

In practice, choosing the right dimension is not that big of an issue. With any algorithm one has to know what precision they want in order to prepare the appropriate amount of hardware resources for the task. Also the quality of intermediate result is rarely relevant. However there is one more problem that emerges: memory requirement.

The value of $f_{2n}(n)$ is roughly proportional to 2^n. This means that intermediate values will take up $O(n)$ additional storage and that you also have to dedicate another $O(n)$ storage in order to catch the rounding errors that would otherwise occur while the values are still growing. That means an additional $O(n)$ space both before and after the decimal point. It doesn't hurt the DSA's overall complexity, but is significant enough that the TDA will always outperform the DSA.

Table 1 suggests that the TDA converges rapidly in comparison, but that is not entirely a just comparison. It makes more sense to compare the value a_k with $f_n(2k)$. The TDA is ultimately a fusion of all the series of the DSA.

In tests computing up to hundreds of millions of digits of π with both algorithms, the TDA was consistently about 10% faster than the DSA and had a slightly lower memory footprint. Initial expectations that with a growing number of digits the advantage would be more extreme, did not turn out to be true.

5.1 Suitability for Practical Applications

Both algorithms have the same time and space complexity as algorithms that are usually used to compute new world records of π. In particular, the Ramanujan-Sato-Series and the Chudnovsky-Algorithm are two popular choices for this purpose.

The question how the DSA and TDA compare to those algorithms in practice is a matter that requires further investigation, however we deem it unlikely that they can break any new records.

The primary advantage of the DSA and TDA lies in their simplicity. They involve only the basic integer operations: addition, subtraction, multiplication and division. Square roots, exponential and trigonometric functions do not come into play. This makes the algorithms potentially easier to implement based on the big-integer facility that is used.

The most likely use-case is an unremarkable one: The algorithms can be included in collections of similar algorithms which are usually used to test the correctness of numerical facilities or test their performance.

If an algorithm to compute a number of digits of π is needed and maximum efficiency is not the highest concern, the DSA and TDA are both suitable candidates. We do however encourage to prefer the TDA over the DSA.

The algorithms are ultimately unlikely to disrupt the world of mathematics and computer science, but they may find their place in it.

6 Further Thoughts on Implementation Strategies and Variations

The following is a collection of facts and ideas about the algorithms that are not necessary for understanding them, but may be useful for their efficient implementation and for a deeper understanding of the subject matter as a whole.

Hybrid-Algorithm of DSA and TDA Given $k \geq n$, it is possible to calculate $f_n(k)$ by calculating $f_n(n)$ or $f_n(2n)$ using the TDA and then continue with the DSA to reach the final result. This bypasses the excessively large intermediate values of the DSA and reduces the rounding errors to a minimum as well. The resulting hybrid algorithm is essentially as good as the TDA.

Initial Factor of DSA To implement the DSA, we can consider the equation

$$\frac{(2n)!!}{(2n-1)!!} = 2^n \times \frac{n!}{(2n-1)!!} = 2^n \times \frac{n!! \times (n-1)!!}{(2n-1)!!}.$$

For $n = 2m$ this can be further simplified to

$$\frac{(2n)!!}{(2n-1)!!} = 2^{2m} \times \frac{(2m)!! \times (2m-1)!!}{(4m-1)!!} = 2^{3m} \times m! \times \frac{(2m-1)!!}{(4m-1)!!}.$$

For $n = 2m+1$ it simplifies to

$$\frac{(2n)!!}{(2n-1)!!} = 2^{2m+1} \times \frac{(2m+1)!! \times (2m)!!}{(4m+1)!!} = 2^{3m+1} \times m! \times \frac{(2m+1)!!}{(4m+1)!!}.$$

Merged into a single equation, it becomes

$$\frac{(2n)!!}{(2n-1)!!} = 2^{\left\lfloor \frac{3}{2}n \right\rfloor} \times \left\lfloor \frac{n}{2} \right\rfloor ! \times \frac{(n - ((n+1) \bmod 2))!!}{(2n-1)!!}.$$

This cuts the number of factors in half and also extracts a large power of 2 which both greatly improve the runtime of a potential implementation.

Alternatively, one can also express the initial factor as

$$\frac{(2n)!!}{(2n-1)!!} = \frac{((2n)!!)^2}{(2n)!} = 2^{2n} \times \frac{(n!)^2}{(2n)!} = 2^{2n} \times \binom{2n}{n}^{-1}.$$

which allows to calculate the binomial coefficient with any of many already existing methods.

TDA as 3 Separate Series From our construction earlier, we know that the TDA can be expressed as

$$a_n = 2 + \sum_{i=0}^{n-1} (b_i + c_i + d_i)$$

with

$$\begin{aligned}
b_n &= \frac{(-1)^n}{2^{2n-1}(2n+1)} \\
c_n &= \frac{(-1)^{n+1}(n+1)}{2^{2n-1}(2n+1)(4n+3)} \\
d_n &= \frac{(-1)^{n+1}}{2^{2n+1}(4n+5)}.
\end{aligned}$$

The series consists of 3 subseries which we call the *b*-series, the *c*-series and the *d*-series. These subseries converge just as fast as the TDA as a whole. Their separate computation is one potential angle of parallelization.

The subseries can be recursively defined as

$$\begin{aligned}
b_0 &= 2, & \frac{b_{n+1}}{b_n} &= -\frac{2n+1}{4(2n+3)}, \\
c_0 &= -\frac{2}{3}, & \frac{c_{n+1}}{c_n} &= -\frac{(n+2)(2n+1)(4n+3)}{4(n+1)(2n+3)(4n+7)}, \\
d_0 &= -\frac{1}{10}, & \frac{d_{n+1}}{d_n} &= -\frac{4n+5}{4(4n+9)}.
\end{aligned}$$

There is also the possibility to define them in a circular fashion:

$$\begin{aligned}
\frac{c_n}{b_n} &= -\frac{n+1}{4n+3}, \\
\frac{d_n}{c_n} &= \frac{(2n+1)(4n+3)}{4(n+1)(4n+5)}, \\
\frac{b_{n+1}}{d_n} &= \frac{4n+5}{2n+3}.
\end{aligned}$$

TDA as a Single Series The initial description of the TDA also lends itself to an alternative angle:

$$\begin{aligned}
a_n &= 2 + \sum_{i=0}^{n-1} \left(\frac{-1}{4}\right)^i \left(\frac{2}{2i+1} - \frac{2(i+1)}{(2i+1)(4i+3)} - \frac{1}{2(4i+5)}\right) \\
&= 2 + \sum_{i=0}^{n-1} \left(\frac{-1}{4}\right)^i \frac{40i^2+82i+37}{64i^3+160i^2+124i+30}.
\end{aligned}$$

This angle is suited for implementations that can evaluate small polynomials very fast.

Unfortunately, this is not as easily expressed recursively. If we write each term as

$$q_n = \left(\frac{-1}{4}\right)^n \frac{40n^2 + 82n + 37}{64n^3 + 160n^2 + 124n + 30},$$

then the recursive relationship essentially boils down to

$$\begin{aligned} q_0 &= \frac{37}{30}, \\ \frac{q_{n+1}}{q_n} &= -\frac{(40(n+1)^2 + 82(n+1) + 37)(64n^3 + 160n^2 + 124n + 30)}{4(64(n+1)^3 + 160(n+1)^2 + 124(n+1) + 30)(40n^2 + 82n + 37)} \\ &= -\frac{1280n^5 + 8384n^4 + 20528n^3 + 23364n^2 + 12288n + 2385}{5120n^5 + 38656n^4 + 113344n^3 + 160592n^2 + 109056n + 27972}. \end{aligned}$$

7 Conclusion

In this paper we introduced two algorithms to compute π to arbitrary precision. Due to their relationship with hyperballs in higher dimensions, we call these algorithms the *Dimension Specific Algorithm* and the *Trans-Dimensional Algorithm*.

Unfortunately—due to the author's lack of formal education—the core claims of this paper could not be proven. We instead focused on summarizing the thought processes that led to the discovery of the algorithms and reduced the gaps to a few contained conjectures to hopefully convince the reader of the overall validity of the algorithms. Both algorithms were successfully tested to compute hundreds of millions of digits of π which further supports the conjectures.

Alongside this document, a simple, reasonably efficient implementation for both algorithms will be published in order to serve as further means of understanding and verification.

Both algorithms are theoretically suitable for the task and decently efficient, however the Trans-Dimensional Algorithm tends to be slightly more efficient and has a convergence behavior that is easier to reason about.

Closing Words *I want to make a personal statement. This paper is the result of dedicated work spanning almost a decade. From the initial idea to calculate the volumes of hyperballs, every so often a new breakthrough would ignite my enthusiasm to continue. My work turned from a funny little series that is useless in practice into two algorithms that are comparable with the best (even if inferior in practice).*

While a lot of progress has been made in analyzing the convergence behavior and in optimizing the implementation, it pains me that after all my efforts, I am unable to prove my conjectures.

I have no academic degree and am not connected in the world of academia which makes it difficult for me to discuss my findings with peers. I don't know how much of the contents of this paper is already common knowledge and I don't know how to find out. It would be rather embarrassing if all the conjectures have already been proven and my work brings nothing new to the table.

Regardless, even though the work is incomplete and even if it is not a revolutionary breakthrough, it is my desire to make it accessible to the world hoping that my efforts are not all for nothing.