# Computing the Lambert W function in arbitrary-precision complex interval arithmetic

Fredrik Johansson

# Computing the Lambert W function in arbitrary-precision complex interval arithmetic

Fredrik Johansson

INRIA Bordeaux, Talence, France
fredrik.johansson@gmail.com

**Abstract**

We describe an algorithm to evaluate all the complex branches of the Lambert $W$ function with rigorous error bounds in arbitrary-precision interval arithmetic or ball arithmetic. The classic 1996 paper on the Lambert $W$ function by Corless *et al.* provides a thorough but partly heuristic numerical analysis of the Lambert $W$ function which needs to be complemented with some explicit inequalities and practical observations about managing precision and branch cuts. An implementation is provided in the Arb library.

## 1 Introduction

The Lambert $W$ function $W(z)$ is the inverse function of $f(w) = we^w$, meaning that $W(z)e^{W(z)} = z$ holds for any $z$. Since $f$ is not injective, the Lambert $W$ function is multivalued, having an infinite number of branches $W_k(z)$, $k \in \mathbb{Z}$, analogous to the branches $\ln_k(z) = \log(z) + 2\pi ik$ of the natural logarithm which inverts $g(w) = e^w$.

The study of the equation $we^w = z$ goes back to Lambert and Euler in the 18th century, but a standardized notation for the solutions of this equation only appeared in the 1990s with the introduction of `LambertW` in the Maple computer algebra system, along with the paper [4] by Corless, Gonnet, Hare, Jeffrey and Knuth which collected and proved the function's main properties. The Lambert $W$ function is now widely known as a standard mathematical function [15, §4.13] with a vast literature on applications, and in 2016 a conference was held to celebrate the first 20 years of the Lambert $W$ function [7].

Several authors have provided approximations or software implementations of the Lambert $W$ function, among them [5, 2, 4, 12, 3, 17, 6, 1]. The state of the art for computing the real branches in machine (single or double) precision is the work by Fukushima [6]. The Corless *et al.* paper [4] sketches how $W_k(z)$ can be computed for any $z \in \mathbb{C}$ and any $k$, using a combination of series expansions and iterative root-finding. Both the `LambertW` function in Maple [13] and the `ProductLog` function in Wolfram Mathematica [18] allow computing all the complex branches with arbitrary precision and also permit symbolic manipulation. An open source arbitrary-precision implementation can also be found in the mpmath Python library [10].

However, there is no published work to date addressing evaluation of the Lambert $W$ function in interval arithmetic [14] or discussing a complete rigorous implementation of the complex branches.

Solutions of the equation $we^w - z = 0$ can naturally be isolated rigorously using standard interval root-finding methods such as subdivision and the interval Newton method [14]. Another possibility, suggested by Corless *et al.*, is to use a posteriori error analysis to bound the error of an approximate solution. The Lambert $W$ function can also be characterized in terms of the nonlinear ordinary differential equation $z(1+W)W' = W$, allowing the use of validated ODE solvers. Regardless of the approach, an important issue is to maintain both correctness and efficiency near singularities and branch cuts.

This work gives an algorithm for rigorous evaluation of the Lambert $W$ function in complex interval arithmetic, which has been implemented in the Arb library [9]. The following goals were pursued:

- At high precision, $W(z)$ is only a constant factor more expensive to compute than elementary functions like $\log(z)$ or $\exp(z)$. For rapid, rigorous computation of elementary functions with arbitrary precision in Arb, the methods by Johansson [8] are used.
- The output enclosures are reasonably tight.
- All the complex branches $W_k$ are supported, with a stringent treatment of branch cuts.
- It is possible to compute derivatives $W^{(n)}(z)$ efficiently, for arbitrary $n$.

The main contribution of this work is to derive bounds with explicit constants for a posteriori certification and for the truncation error in certain series expansions, in cases where previous publications give big-O estimates. We also discuss the implementation of the complex branches in detail.

The presentation will occasionally rely on the implementation details of interval arithmetic in Arb, but the algorithms can easily be adapted to other formats. Arb uses real intervals of the form $[m \pm r] = [m-r, m+r]$, where the midpoint $m$ is an arbitary-precision floating-point number and the radius $r$ is an unsigned fixed-precision floating-point number. This form of interval arithmetic is known as ball arithmetic [16]. In Arb, the exponents of $m$ and $r$ are bignums which can be arbitrarily large (this is useful for asymptotic problems, and removes edge cases with underflow or overflow). Complex numbers in Arb are represented in rectangular form $x + yi$ using pairs of real balls. For such a complex number $z = [m_1 \pm r_1] + [m_2 \pm r_2]i$, we write $\operatorname{mid}(z) = m_1 + m_2 i$ and $\operatorname{rad}(z) = \sqrt{r_1^2 + r_2^2}$.

## 2 Complex branches

In this work, $W_k(z)$ always refers to the standard $k$-th branch as defined by Corless *et al.* [4]. We sometimes write $W(z)$ when referring to the multivalued Lambert $W$ function or a branch implied by the context. Before we proceed to describe any algorithms, we summarize the branch structure of $W$. We refer to Corless *et al.* for a more detailed explanation including illustrations.

Figure 1 demonstrates evaluation of the Lambert $W$ function in the two real-valued regions. The *principal branch* $W_0(z)$ is real-valued and monotone increasing for real $z \geq -1/e$, with the image $[-1, \infty)$, while $W_{-1}(z)$ is real-valued and monotone decreasing for real $-1/e \leq z < 0$, with the image $(-\infty, -1]$. Everywhere else, $W_k(z)$ is complex. There is a square root-type singularity at the branch point $z = -1/e$ connecting the real segments, where $W_0(-1/e) = W_{-1}(-1/e) = -1$. The principal branch contains the root $W_0(0) = 0$, which is the only root of $W$. For all $k \neq 0$, the point $z = 0$ is a branch point with a logarithmic singularity.

Figure 2 shows the branch cuts and connections for the complex branches $W_k(z)$, which may be summarized as follows:

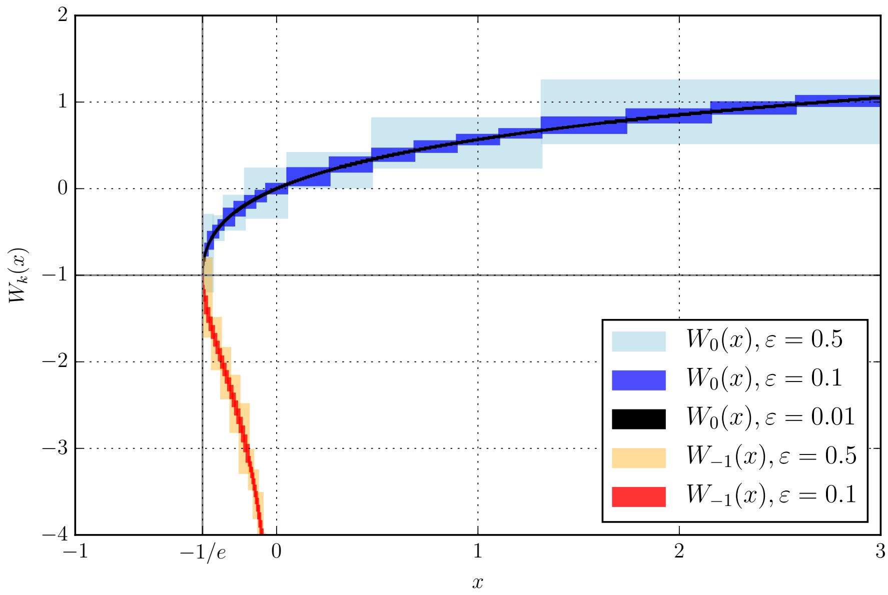


Figure 1: Plot of the real branches $W_0(x)$ and $W_{-1}(x)$ computed with Arb. The boxes show the size of the output intervals given wide input intervals. In this plot, the input intervals have been subdivided until the output radius is smaller than $\varepsilon$.

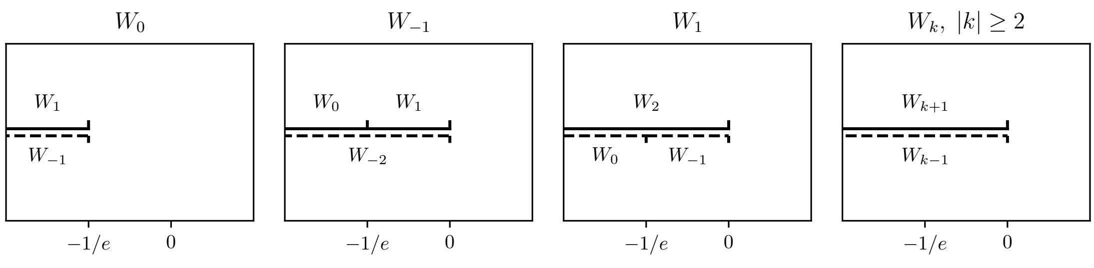


Figure 2: The solid and dashed lines show the branch cuts for $W_k(z)$ in the complex $z$-plane. The branch points are $-1/e$ and 0 (and $-\infty$). Solid lines indicate closure; that is, since the solid lines are on top of the branch cuts, the function value on the cut itself is defined to be continuous from above. In each plot, the label $W_j$ next to a branch cut indicates the connecting branch on the other side of the cut when $W$ is continued analytically.

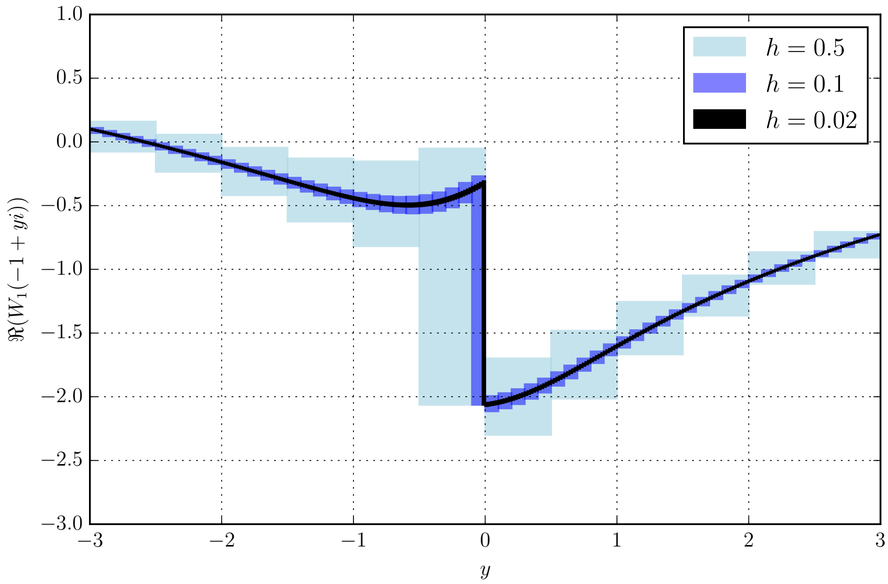


Figure 3: Plot of the real part of $W_1(z)$ on the vertical segment $z = -1 + yi, |y| \leq 3$ computed with Arb. The boxes show the range of the output intervals given input intervals $y = [a, a + h]$. The picture demonstrates continuity between the branch cut and the upper half plane: as intended, an imaginary part of $[-h, 0]$ (or $[-h/2, h/2]$, say, though not pictured here) in the input captures the jump discontinuity while $[0, h]$ does not. The output intervals are seen to converge as $h \to 0$ iff $y = [a, a + h]$ excludes regions $(-\varepsilon, 0], \varepsilon > 0$.

- The principal branch $W_0(z)$ has a single branch cut on $(-\infty, -1/e)$, while the branches $W_k(z)$ with $|k| \geq 2$ have a single branch cut on $(-\infty, 0)$.
- The branches $W_{\pm 1}$ are more complicated, with a set of adjacent branch cuts: in the upper half plane, $W_{-1}$ has a branch cut on $(-\infty, -1/e)$ and one on $(-1/e, 0)$; in the lower half plane, $W_{-1}$ has a single branch cut on $(-\infty, 0)$. $W_1$ is similar to $W_{-1}$, but with the sides exchanged.
- The branch cuts on $(-\infty, 0)$ or $(-\infty, -1/e)$ connect $W_k$ with $W_{k+1}$, while the branch cuts on $(-1/e, 0)$ connect $W_{-1}$ with $W_1$.

We follow the convention that the function value on a branch cut is continuous when approaching the cut in the counterclockwise direction around a branch point. For the standard branches $W_k(z)$, this is the same as continuity with the upper half plane, i.e. $W_k(x + 0i) = \lim_{y \to 0+} W_k(x + yi)$. When $\mathrm{Im}(z) \neq 0$, we have $W_k(z) = \overline{W_{-k}(\overline{z})}$. By the same convention, the principal branch of the natural logarithm is defined to satisfy $\mathrm{Im}(\log(z)) \in (-\pi, +\pi]$. The direction of continuity on the branch cut is illustrated in Figure 3.

We do not use signed zero in the sense of IEEE 754 floating-point arithmetic, which would allow preserving continuity from either side of a branch cut. This is a trivial omission since we could distinguish between $W(x+0i)$ and $W(x-0i)$ using $W_k(x-0i) = \overline{W_{-k}(x+0i)}$.

## 2.1 Alternative branch cuts

If the input $z$ is an exact floating-point number or represented by an interval with exact imaginary part, say $z = [-5, -4] + 0i$, then we can always pinpoint its location in relation to the standard branch cuts of $W$.

In interval arithmetic, we need to enclose the union of the images of $W(z)$ on both sides of the cut when the interval representing $z$ straddles a branch cut. If the input has been generated by an interval computation, it might be represented by an interval like $z = -5 + [\pm\varepsilon]i$ where the sign of $\mathrm{Im}(z)$ is ambiguous. The jump discontinuity between the cuts will prevent the output intervals from converging when the input intervals shrink.

This can be a problem if we want to enumerate all the solutions of $we^w = z$ or trace a specific solution when $z$ changes continuously across a branch cut. A typical application is to evaluate an integral or a solution of a differential equation involving $W$, say $\int_a^b f(z, W(g(z)))dz$, where we might need to consider paths that would cross the standard branch cuts.

It is instructive to consider the case of square roots and logarithms, where the branch cut can be moved from $(-\infty, 0)$ to $(0, \infty)$ quite easily. The solutions of $w^2 = z$ are given by $w = \sqrt{z}, -\sqrt{z}$, but switching to $w = i\sqrt{-z}, -i\sqrt{-z}$ gives continuity along paths crossing the negative real axis. Similarly, for the solutions of $e^w = z$, we can switch between $w = \log(z) + 2k\pi i$ and $w = \log(-z) + (2k+1)\pi i$, $k \in \mathbb{Z}$.

The Lambert $W$ function lacks any obvious functional equation that would allow us to simply negate $z$. We solve this problem by defining a complementary set of functions $\mathcal{W}_k^{\text{left}}(z)$ $(k \in \mathbb{Z})$ and $\mathcal{W}^{\text{middle}}(z)$ with branch cuts placed opposite those of $W_k(z)$.

A pictorial definition is given in Figure 4. More precisely, we define[1] (with $z = x + yi$)

$$\mathcal{W}_k^{\text{left}}(z) = \begin{cases} W_k(z) & \text{if } y > 0 \\ W_{k+1}(z) & \text{if } y < 0 \vee (y = 0 \wedge x > 0) \\ W_{-1-k}(z) & \text{if } k \in \{-1, 0\} \wedge (-1/e < x < 0) \\ W_k(z) & \text{otherwise,} \end{cases} \tag{1}$$

and

$$\mathcal{W}^{\text{middle}}(z) = \begin{cases} W_{-1}(z) & \text{if } y > 0 \vee (y = 0 \wedge x < 0) \\ W_1(z) & \text{otherwise.} \end{cases} \tag{2}$$

$\mathcal{W}_k^{\text{left}}(z)$ glues $W_k(z)$ for $z$ in the upper half plane with $W_{k+1}(z)$ in the lower half plane, with continuity through the *left* of the plane (and a branch cut accordingly extending to $+\infty$ instead of $-\infty$).

$\mathcal{W}^{\text{middle}}(z)$ glues $W_{-1}(z)$ in the upper half plane with $W_1(z)$ in the lower half plane, with continuity through the central segment $(-1/e, 0)$. This function extends the real analytic function $W_{-1}(x), x \in (-1/e, 0)$ to a complex analytic function on $z \in \mathbb{C} \setminus (-\infty, -1/e] \cup [0, \infty)$, unlike $W_{-1}(z)$ where the real-valued segment lies precisely on the branch cut.

[1] The case distinctions on the real line are made to preserve the principle of counter-clockwise continuity (absent use of signed zero), which here becomes equivalent to continuity with the lower half plane. This is ultimately not important since the purpose of introducing $\mathcal{W}_k^{\text{left}}$ is to *avoid* evaluation on the branch cuts, and we could just as well define a simplified version of $\mathcal{W}_k^{\text{left}}$ as

$$\mathcal{W}_k^{\text{left}}(z) = \begin{cases} W_k(z) & \text{if } y \geq 0 \\ W_{k+1}(z) & \text{otherwise.} \end{cases}$$

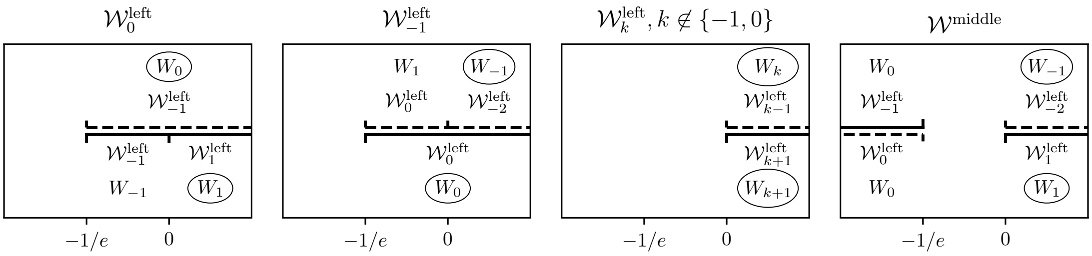


Figure 4: Locations of branch cuts and connecting branches for $\mathcal{W}_k^{\text{left}}(z)$ and $\mathcal{W}^{\text{middle}}(z)$. For each branch cut, the two labels show the branch of both $W_k$ and $\mathcal{W}_k^{\text{left}}$ that connects to the current branch at this cut. The circled label shows the standard branch $W_k$ that coincides with the current branch $\mathcal{W}_k^{\text{left}}(z)$ or $\mathcal{W}^{\text{middle}}(z)$ in the respective upper and lower half plane.

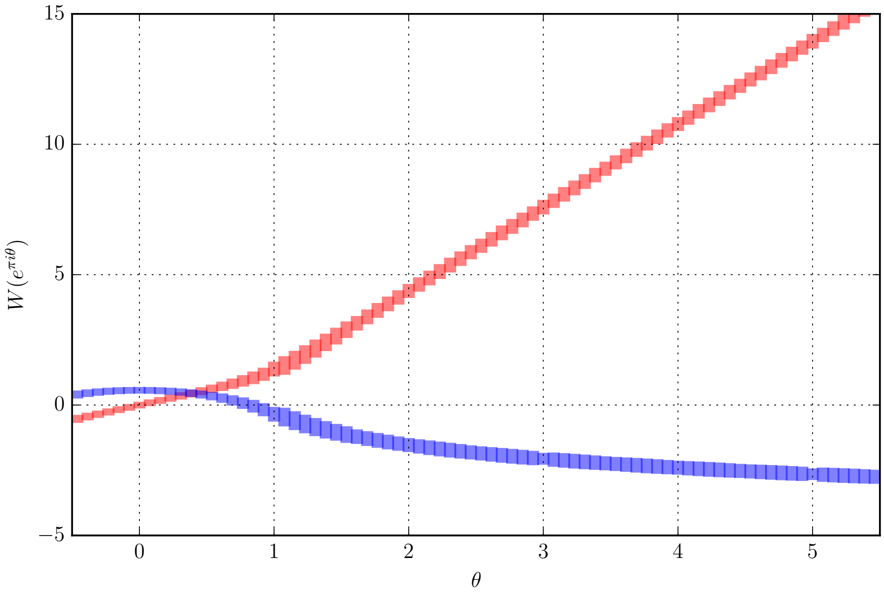


Figure 5: Plot of the real part (even function) and imaginary part (odd function) of $W(e^{\pi i\theta})$ with continuous analytic continuation on the Riemann surface of $W$. The branch used for evaluation is $W_0$ on $\theta \in [-0.5, 0.5]$, $\mathcal{W}_0^{\text{left}}$ on $[0.5, 1.5]$, $W_1$ on $[1.5, 2.5]$, $\mathcal{W}_1^{\text{left}}$ on $[2.5, 3.5]$, $W_2$ on $[3.5, 4.5]$, and $\mathcal{W}_2^{\text{left}}$ on $[4.5, 5.5]$. Thanks to the alternation between the complementary functions to avoid branch cuts, continuity is preserved whenever $\theta$ crosses an integer, that is, when $z = e^{\pi i\theta}$ crosses the real axis. The input intervals for $\theta$ have width $1/13$.

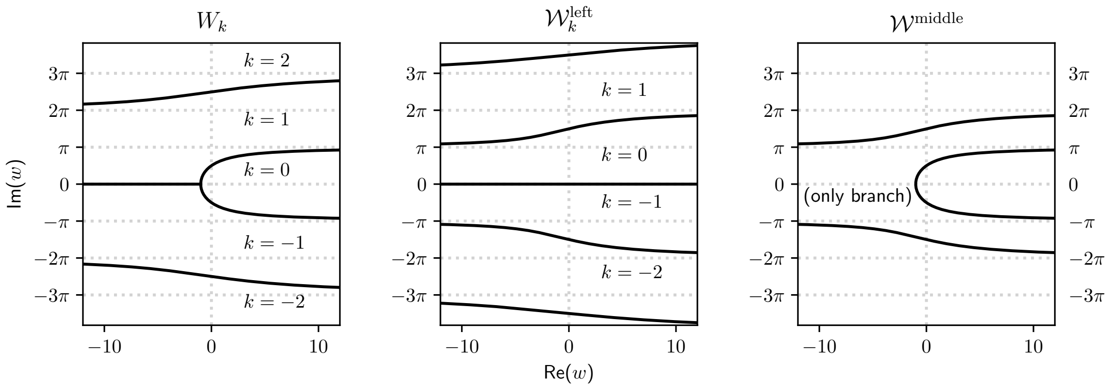


Figure 6: Alternative branch cuts of the Lambert $W$ function. The plots show how the branch cuts partition the $w$-plane, i.e. which image points $w = W(z)$ correspond to which branch.

At any point $(z, w)$ on the Riemann surface of the Lambert $W$ function other than the branch point $z = -1/e$, $w = -1$, at least one of $W_k$, $\mathcal{W}_k^{\text{left}}$ and $\mathcal{W}^{\text{middle}}$ will be analytic in a neighborhood of that point, so we can always choose a branch that maintains continuity on complex intervals around $z$.

**Example 1.** *Consider for example a counterclockwise circular trajectory through* $z = 1 \to i \to -1 \to -i \to 1$ *where we start on the* $W_0$ *branch (*$W_0(1) \approx 0.567$*) and end up on the* $W_1$ *branch (*$W_1(1) \approx -1.53 + 4.38i$*). If we start the numerical evaluation on* $W_0$*, switch to* $\mathcal{W}_0^{\text{left}}(z)$ *on complex intervals with* $\text{mid}(\text{Re}(z)) < 0$ *for passing through* $z = -1$*, and switch to* $W_1$ *for the final quarter-rotation when* $\text{mid}(\text{Re}(z)) > 0$*, then small complex intervals covering the path are well-separated from branch cuts and continuity is maintained (see Figure 5).*

**Example 2.** *An application involving both* $\mathcal{W}_k^{\text{left}}$ *and* $\mathcal{W}^{\text{middle}}$ *is given below in section 4.2. in the proof of Theorem 1 (see Figure 7).*

A good way to understand the branch cuts is to visualize how they partition the $w$-plane, as shown in Figure 6. The curves in the $w$-plane separating the branches are given by

$$C_{(a,b)} = \{-\eta \cot(\eta) + \eta i : a\pi < \eta < \pi b\}$$

for different integers $a, b$, together with the rays $(-\infty, -1)$ and $(-1, \infty)$.

- The ray $(-\infty, -1)$ in the $w$-plane corresponds to the middle segment $(-1/e, 0)$ in the $z$-plane connecting $W_{-1}$ and $W_1$. Note that $\mathcal{W}^{\text{middle}}$ removes this branch cut.
- The central curve $C_{(-1,1)}$ in the $w$-plane corresponds to the ray $(-\infty, -1/e)$ in the $z$-plane which connects $W_0$ to $W_{-1}$ and $W_1$.
- The curves $C_{\pm(2,3)}, C_{\pm(4,5)}, C_{\pm(6,7)}, \ldots$ in the $w$-plane correspond to the ray $(-\infty, 0)$ in the $z$-plane connecting the branches $W_{\pm 1}, W_{\pm 2}, W_{\pm 3}, \ldots$.
- The curves $C_{\pm(1,2)}, C_{\pm(3,4)}, C_{\pm(5,6)}, \ldots$ in the $w$-plane correspond to the ray $(0, +\infty)$ in the $z$-plane connecting the remaining branches $\mathcal{W}_k^{\text{left}}$ (in other words, cutting $W_{\pm 1}, W_{\pm 2}, \ldots$ in half along the positive real axis).

- The ray $(-1, +\infty)$, finally, cuts $W_0$ in half for $z$ crossing $(-1/e, \infty)$

We only work with the standard branches $W_k(z)$ in the algorithm for computing the Lambert $W$-function given below in sections 3 and 4. The alternative functions $\mathcal{W}_k^{\text{left}}$ and $\mathcal{W}^{\text{middle}}$ are implemented in terms of the standard branches $W_k(z)$ with a simple wrapper computation as explained in section 5.

# 3 The main algorithm

Our algorithm to evaluate the Lambert $W$ function has three main ingredients:

- *Asymptotic cases.* If $|z|$ is extremely small or large, or if $z$ is extremely close to the branch point at $-1/e$ when $W(z) \approx -1$, we use the respective Taylor, Puiseux or asymptotic series to compute $W(z)$ directly.
- *Approximation.* We use floating-point arithmetic to compute some approximation $\tilde{w} \approx W(\text{mid}(z))$ heuristically.
- *Certification.* Given an approximation $\tilde{w}$, we use interval arithmetic (or floating-point arithmetic with directed rounding) to determine a bound $r$ such that $|W(z) - \tilde{w}| \le r$. We then return $\tilde{w} + [\pm r] + [\pm r]i$, or simply $[\tilde{w} \pm r]$ when $W(z)$ is real-valued.

The special treatment of asymptotic cases is not necessary to construct a complete algorithm, but it improves performance since the error can be bounded directly without a separate certification step. We give error bounds for the truncated series expansions in section 4.

Computing a floating-point approximation with heuristic error control is a well understood problem, and we avoid going into too much detail here. Arb uses the Halley iteration

$$w_{j+1} = w_j - \frac{w_j e^{w_j} - \text{mid}(z)}{e^{w_j + 1} - \dfrac{(w_j + 2)(w_j e^{w_j} - \text{mid}(z))}{2w_j + 2}}$$

suggested by Corless *et al.* to solve $we^w - \text{mid}(z) = 0$. This roughly triples the number of accurate bits with each iteration when started from a well-chosen initial value. In the most common cases, machine `double` arithmetic is first used to achieve near 53-bit accuracy (with care to avoid overflow or underflow problems or loss of significance near $z = -1/e$). For typical accuracy goals of less than a few hundred bits, this leaves at most a couple of iterations to be done using arbitrary-precision arithmetic.

In the arbitrary-precision phase, we need $O(\log p)$ Halley iteration steps to obtain $p$-bit accuracy. The working precision is initially set low and then increases with each iteration to match the estimated number of accurate bits, ensuring that the total time complexity is equivalent to that of performing $O(1)$ exponential function evaluations at the final precision $p$.

## 3.1 Certification

To compute a certified error bound for $\tilde{w}$, we use backward error analysis, following the suggestion of Corless *et al.*. We compute $\tilde{z} = \tilde{w}e^{\tilde{w}}$ with interval arithmetic, and use

$$\tilde{w} = W(\tilde{z}) = W(z) + \int_z^{\tilde{z}} W'(t)dt. \tag{3}$$

to bound the error $W_k(\tilde{z}) - W_k(z)$. This approach relies on having a way to bound $|W_k'|$, which we address in section 4.

The formal identity (3) is only valid provided that the correct integration path is taken on the Riemann surface of the multivalued $W$ function. During the certification, we verify that the straight-line path $\gamma$ from $z$ to $\tilde{z}$ for $W_k$ is correct in (3), so that the error is bounded by $|z - \tilde{z}| \sup_{t \in \gamma} |W_k'(t)|$. This is essentially to say that we have approximated $W_k(z)$ for the right $k$, since a poor starting value (or rounding error) in the Halley iteration could have put $\tilde{w}$ on the wrong branch, or closer to a solution on the wrong branch than the intended solution.

---

**Algorithm 1** Compute certified enclosure of $W_k(z)$. The input is a complex interval $z$, a branch index $k \in \mathbb{Z}$, and a complex floating-point number $\tilde{w}$.

---

1. Verify that $\tilde{w} = x + yi$ lies in the range of the branch $W_k$:
   (a) Compute $t = x \operatorname{sinc}(y)$, $v = -\cos(y)$, $u = \operatorname{sgn}(k) y / \pi$ using interval arithmetic.
   (b) If $k = 0$, check $(|u| < 1) \wedge (t > v)$.
   (c) If $k \neq 0$, check $P_1 \wedge (P_2 \vee P_3 \vee P_4)$ where
   $$\begin{aligned} P_1 &= (u > 2|k| - 2) \wedge (u < 2|k| + 1) \\ P_2 &= (u > 2|k| - 1) \wedge (u < 2|k|) \\ P_3 &= (u < 2|k|) \wedge (t < v) \\ P_4 &= (u > 2|k| - 1) \wedge (t > v). \end{aligned}$$
   (d) If the check fails, return $[\pm\infty] + [\pm\infty]i$.
2. Compute $\tilde{z} = \tilde{w} e^{\tilde{w}}$ using interval arithmetic.
3. Compute a complex interval $U \supseteq z \cup \tilde{z}$ ($U$ will contain the straight line from $z$ to $\tilde{z}$).
4. Verify that $U$ does not cross a branch cut: check
   $$(\operatorname{Im}(U) \geq 0) \vee (\operatorname{Im}(U) < 0) \vee \begin{pmatrix} \operatorname{Re}(eU + 1) > 0 & \text{if } k = 0 \\ \operatorname{Re}(U) > 0 & \text{if } k \neq 0 \end{pmatrix}.$$
   If the check fails, return $[\pm\infty] + [\pm\infty]i$.
5. Compute a bound $C \geq |W_k'(U)|$ and return $\tilde{w} + [\pm r] + [\pm r]i$ where $r = C|z - \tilde{z}|$.

---

The complete certification procedure is stated in Algorithm 1. In the pseudocode, all pointwise predicates are extended to intervals in the strong sense; for example, $x \geq 0$ evaluates to true if all points in the interval representing $x$ are nonnegative, and false otherwise. A predicate that should be true for exact input in infinite precision arithmetic can therefore evaluate to false due to interval overestimation or insufficient precision.

In the first step, we use the fact that the images of the branches in the complex $w$-plane are separated by the ray $(-\infty, -1/e]$ together with the curves $\{-\eta \cot \eta + \eta i\}$ for $-\pi < \eta < \pi$ and $2k\pi < \pm\eta < (2k+1)\pi$. In the $k \neq 0$ case, the predicates $P_2, P_3, P_4$ cover overlapping regions, allowing the test to pass even if $\tilde{w}$ falls very close to one of the curves with $2k\pi < \pm\eta < (2k+1)\pi$ where a sign change occurs, i.e. when $z$ crosses the real axis to the right of the branch point.

The test in Algorithm 1 always fails when $z$ lies on a branch cut, or too close to a cut to resolve with a reasonable precision, say if $z = -2^{10^{10}} + 10i$ or $z = -10 + 2^{-10^{10}}i$. This problem could be solved by taking the location of $z$ into account in addition that of $\tilde{w}$. In

Arb, a different solution has been implemented, namely to perturb $z$ away from the branch cut before calling Algorithm 1 (together with an error bound for this perturbation). This solution is not elegant, but it works well in practice with the use of a few guard bits, and seemed to require less extra logic to implement.

Due to the cancellation in evaluating the residual $z - \tilde{z}$, the quantity $\tilde{z} = \tilde{w}e^{\tilde{w}}$ needs to be computed to at least $p$-bit precision in the certification step to achieve a relative error bound of $2^{-p}$. Here, a useful optimization is to compute $e^{w_j}$ with interval arithmetic in the last Halley update $\tilde{w} = w_{j+1} = H(w_j)$ and then compute $e^{\tilde{w}}$ as $e^{w_j}e^{\tilde{w}-w_j}$. Evaluating $e^{\tilde{w}-w_j}$ costs only a few series terms of the exponential function since $|\tilde{w} - w_j| \approx 2^{-p/3}$.

A different possibility for the certification step would be to guess an interval around $\tilde{w}$ and perform one iteration with the interval Newton method. This can be combined with the main iteration, simultaneously extending the accuracy from $p/2$ to $p$ bits and certifying the error bound. An advantage of the interval Newton method is that it operates directly on the function $f(w) = we^w - z$ and its derivative without requiring explicit knowledge about $W'$. This method was tested but ultimately abandoned in the Arb implementation since it seemed more difficult to handle the precision and make a good interval guess in practice, particularly when $z$ is represented by a wide interval. In any case the branch certification would still be necessary.

### 3.2 The main algorithm in more detail

Algorithm 2 describes the main steps implemented by the Arb function with signature

```
void acb_lambertw(acb_t res, const acb_t z,
                        const fmpz_t k, int flags, slong prec)
```

where `acb_t` denotes Arb's complex interval type, `res` is the output variable, `fmpz_t` is a bignum integer type, and `prec` gives the precision goal $p$ in bits.

In step 2, we switch to separate code for real-valued input and output (calling the function `arb_lambertw` which uses real `arb_t` interval variables). The real version implements essentially the same algorithm as the complex version, but skips most branch cut related logic.

In step 3, we reduce the working precision to save time if the input is known to less than $p$ accurate bits. The precision is subsequently adjusted in step 5, accounting for the fact that we gain accurate bits in the value of $W_k(z)$ from the exponent of $\mathrm{mid}(z)$ or $k$ when $|W_k(z)|$ is large. Step 5 is cheap, as it only requires inspecting the exponents of the floating-point components of $z$ and computing bit lengths of integers.

The constants $T, L, M, P$ appearing in steps 4, 6 and 7 are tuning parameters to control the number of series expansion terms allowed to compute $W$ directly instead of falling back to the generic root-finding case. These parameters could be made precision-dependent to optimize performance, but for most purposes small constants (of the order 10) work well.

Step 8 ensures that $z$ lies on one side of a branch cut, splitting the evaluation of $W_k(z)$ into two subcases if necessary. This step ensures that step 12 (which bounds the propagated error due to the uncertainty in $z$) is correct, since our bound for $W'$ does not account for the branch cut jump discontinuity (and in any case differentiating a jump discontinuity would give the output $[\pm\infty] + [\pm\infty]i$ which is needlessly pessimistic). We note that conjugation is used to get a continuous evaluation of $W_k(\mathrm{Re}(z) + (\mathrm{Im}(z) \cap (-\infty, 0))i)$, in light of our convention to work with closed intervals and make the standard branches $W_k$ continuous from above on the cut.

**Algorithm 2** Main algorithm for $W_k(z)$ implemented in `acb_lambertw`. The input is a complex interval $z$, a branch index $k \in \mathbb{Z}$, and a precision $p \in \mathbb{Z}_{\geq 2}$.

1. If $z$ is not finite or if $k \neq 0$ and $0 \in z$, return indeterminate ($[\pm\infty] + [\pm\infty]i$).
2. If $k = 0$ and $z \subset (-1/e, \infty)$, or if $k = -1$ and $z \subset (-1/e, 0)$, return $W_k(z)$ computed using dedicated code for the real branches.
3. Set the accuracy goal to $q \leftarrow \min(p, \max(10, -\log_2 \operatorname{rad}(z)/|\operatorname{mid}(z)|))$.
4. If $k = 0$ and $|\operatorname{mid}(z)| < 2^{-q/T}$, return $W_0(z)$ computed using $T$ terms of the Taylor series.
5. Compute positive integers $b_1 \approx \log_2(|\log(z) + 2\pi i k|)$, $b_2 \approx \log_2(b_1)$. If $|z|$ is near $\infty$, or near 0 and $k \neq 0$, adjust the goal to $q \leftarrow \min(p, \max(q + b_1 - b_2, 10))$.
6. Let $s = 2 - b_1$, $t = 2 + b_2 - b_1$. If $b_1 - \max(t + Ls, Mt) > q$, return $W_k(z)$ computed using the asymptotic series with $(L, M)$ terms.
7. Check if $z$ is near the branch point at $-1/e$: if $|ez + 1| < 2^{-2q/P-6}$, and $|k| \leq 1$ (and $\operatorname{Im}(z) < 0$ if $k = 1$, or $\operatorname{Im}(z) \geq 0$ if $k = -1$) return $W_k(z)$ computed using $P$ terms of the Puiseux series.
8. If $z$ contains points on both sides of a branch cut, set $z_a = \operatorname{Re}(z) + (\operatorname{Im}(z) \cap [0, \infty))i$ and $z_b = \operatorname{Re}(z) + (-\operatorname{Im}(z) \cap [0, \infty))i$. Then compute $w_a = W_k(z_a)$ and $w_b = \overline{W_{-k}(z_b)}$ and return $w_a \cup w_b$.
9. Let $x + yi = \operatorname{mid}(z)$. If $x$ lies to the left of a branch point (0 or $-1/e$) and $|y| < 2^{-q}|x|$, set $z' = \operatorname{Re}(z) + [\varepsilon \pm \varepsilon]i$ where $\varepsilon = 2^{-q}|x|$ (if $y < 0$ in this case, modify the following steps to compute $\overline{W_{-k}(z')}$ instead of $W_k(z')$). Otherwise, set $z' = z$.
10. Compute a floating-point approximation $\tilde{w} \approx W_k(\operatorname{mid}(z'))$ to a heuristic accuracy of $q$ bits plus a few guard bits.
11. Convert $\tilde{w}$ to a certified complex interval $w$ for $W_k(\operatorname{mid}(z'))$ by calling Algorithm 1.
12. If $z'$ is inexact, bound $|W_k'(z')| \leq C$ and add $[\pm r] + [\pm r]i$ to $w$, where $r = C \operatorname{rad}(z')$. Return $w$.

We perform step 8 *after* checking if the asymptotic series or Puiseux series can be used, since correctly implemented complex logarithm and square root functions take care of branch cuts automatically. If $z$ needs to be split into $z_a$ and $z_b$ in step 8, then the main algorithm can be called recursively, but the first few steps can be skipped. However, step 7 should be repeated when $k = \pm 1$ since the Puiseux series near $-1/e$ might be valid for $z_a$ or $z_b$ even when it is not applicable for the whole of $z$. This ensures a finite enclosure when $z$ contains the branch point $-1/e$.

# 4 Bounds and series expansions

We proceed to give the series expansions and inequalities needed for various error bounds in the algorithm.

## 4.1 Taylor series

Near the origin of the $k = 0$ branch, we have the Taylor series

$$W_0(z) = \sum_{n=1}^{\infty} \frac{(-n)^{n-1}}{n!} z^n.$$

Since $|n^{n-1}/n!| < e^n$, the truncation error on stopping before the $n = T$ term is bounded by $e^T|z|^T/(1-e|z|)$ if $|z| < 1/e$.

### 4.2 Puiseux series

Near the branch point at $-1/e$ when $W(z) \approx -1$, the Lambert $W$ function can be computed by means of a Puiseux series. This especially useful for intervals containing the point $-1/e$ itself, since we can compute a finite enclosure whereas enclosures based on $W'(z)$ blow up. If $\alpha = \sqrt{2(ez+1)}$, then provided that $|\alpha| < \sqrt{2}$, we have

$$W_k(z) = \begin{cases} B(\alpha) & \text{if } k = 0 \\ B(-\alpha) & \text{if } k = -1 \text{ and } \operatorname{Im}(z) \geq 0 \\ B(-\alpha) & \text{if } k = +1 \text{ and } \operatorname{Im}(z) < 0 \end{cases}$$

where

$$B(\xi) = W\left(\frac{\xi^2 - 2}{2e}\right) = \sum_{n=0}^{\infty} c_n \xi^n. \tag{4}$$

Note that $W_{\pm 1}$ have one-sided branch cuts on $(-\infty, 0)$ and $(-1/e, 0)$. In the opposite upper and lower half planes, there is only a single cut on $(-\infty, 0)$ so the point $-1/e$ does not need to be treated specially.

In (4), the appropriate branches of $W$ are implied so that $B(\xi)$ is analytic on $|\xi| < \sqrt{2}$. In terms of the standard branch cuts $W_k$, that is

$$k = \begin{cases} 0 & \text{if } -\pi/2 < \arg(\xi) \leq \pi/2 \\ 1 & \text{if } \pi < \arg(\xi) < -\pi/2 \\ -1 & \text{otherwise.} \end{cases}$$

The coefficients $c_n$ are rational numbers

$$c_0 = -1,\ c_1 = 1,\ c_2 = -\frac{1}{3},\ c_3 = \frac{11}{72},\ c_4 = -\frac{43}{540}, \ldots$$

which can be computed recursively. From singularity analysis, $|c_n| = O((1/\sqrt{2})^n)$, but we need an explicit numerical bound for computations. The following estimate is not optimal, but adequate for practical use.

**Theorem 1.** *The Puiseux series coefficients in* (4) *satisfy* $|c_n| < 2 \cdot (4/5)^n$*, or more simply,* $|c_n| \leq 1$.

*Proof.* Numerical evaluation of $W$ shows that $|2 + B(\xi)| < 2$ on the circle $|\xi| = 5/4$, so the Cauchy integral formula gives the result. □ □

The verification can of course be done using interval arithmetic, as demonstrated in Figure 7. We stress that there is no circular dependency on Theorem 1 in this computation since the Puiseux series is not used for evaluation that far from from the branch point.

### 4.3 Asymptotic series

The Lambert $W$ function has the asymptotic expansion

$$W_k(z) \sim L_1 - L_2 + \sum_{l=0}^{\infty} \sum_{m=1}^{\infty} c_{l,m} \sigma^l \tau^m \tag{5}$$

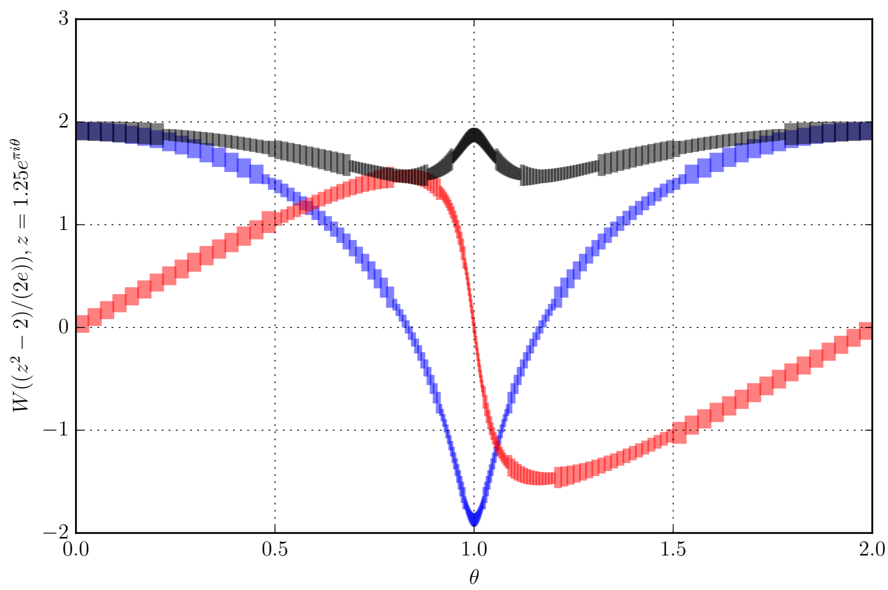


Figure 7: Plot of the real and imaginary parts as well as the absolute value of the function $2 + W((z^2 - 2)/(2e)), z = 1.25e^{\pi i\theta}$ with continuous analytic continuation. The function argument $(z^2-2)/(2e)$ traces two loops around the branch point at $-1/e$, passing through the standard branches $W_0, W_1, W_1$, and back to $W_0$. From left to right, the branch used for evaluation cycles through $W_0, \mathcal{W}_0^{\text{left}}, W_1, \mathcal{W}^{\text{middle}}, W_{-1}, \mathcal{W}_{-1}^{\text{left}}, W_0$. Input intervals have been subdivided adaptively to demonstrate the absolute value bound in Theorem 1.

where

$$L_1 = \log(z) + 2\pi k i, \quad L_2 = \log(L_1), \quad \sigma = 1/L_1, \quad \tau = L_2/L_1 \tag{6}$$

and

$$c_{l,m} = \frac{(-1)^m}{m!} \begin{bmatrix} l+m \\ l+1 \end{bmatrix} \tag{7}$$

where $\left[{n \atop k}\right]$ denotes an (unsigned) Stirling number of the first kind.

This expansion is valid for all $k$ when $|z| \to \infty$, and also for $k \neq 0$ when $|z| \to 0$. In fact, (5) is not only an asymptotic series but (absolutely and uniformly) convergent for all sufficiently small $|\sigma|, |\tau|$. These properties of the expansion (5) were proved by Corless *et al.*

The asymptotic behavior of the coefficients $c_{l,m}$ was studied further by Kalugin and Jeffrey [11], but they did not give explicit inequalities. We will give an explicit bound for $|c_{l,m}|$, which permits us to compute $W_k(z)$ directly from (5) with a bound on the error in the relevant asymptotic regimes.

**Lemma 2.** *For all $n, k \geq 0$,*

$$\begin{bmatrix} n \\ k \end{bmatrix} \leq \frac{2^n n!}{k!}.$$

*Proof.* This is easily proved using the recurrence relation

$$\begin{bmatrix} n+1 \\ k \end{bmatrix} = n \begin{bmatrix} n \\ k \end{bmatrix} + \begin{bmatrix} n \\ k-1 \end{bmatrix}$$

or by contour integration applied to the generating function

$$\sum_{n=k}^{\infty} (-1)^{n-k} \begin{bmatrix} n \\ k \end{bmatrix} \frac{z^n}{n!} = \frac{(\log(1+z))^k}{k!}. \qquad \square$$

$\square$

**Lemma 3.** *For all $l, m \geq 0$, $|c_{l,m}| \leq 4^{l+m}$.*

*Proof.* By the previous lemma,

$$|c_{l,m}| \leq \frac{2^{l+m}(l+m)!}{(l+1)!m!} \leq 2^{l+m} \binom{l+m}{m} \leq 4^{l+m}. \qquad \square$$

$\square$

We can now restate the asymptotic expansion (5) in the following computationally effective form.

**Theorem 4.** *With $\sigma, \tau, L_1, L_2$ defined as above, if $|\sigma| < 1/4$ and $|\tau| < 1/4$, and if $|z| > 1$ when $k = 0$, then*

$$W_k(z) = L_1 - L_2 + \sum_{l=0}^{L-1} \sum_{m=1}^{M-1} c_{l,m} \sigma^l \tau^m + \varepsilon_{L,M}(z)$$

*with*

$$|\varepsilon_{L,M}(z)| \leq \frac{4|\tau|(4|\sigma|)^L + (4|\tau|)^M}{(1 - 4|\sigma|)(1 - 4|\tau|)}.$$

*Proof.* Under the stated conditions, the analysis by Corless *et al.* shows that the series (5) converges to $W_k(z)$. We can bound the tail as

$$\left| \sum_{l=L}^{\infty} \sum_{m=M}^{\infty} c_{l,m} \sigma^l \tau^m \right| \leq \sum_{l=0}^{\infty} \sum_{m=M}^{\infty} (4|\sigma|)^l (4|\tau|)^m + \sum_{l=L}^{\infty} \sum_{m=1}^{\infty} (4|\sigma|)^l (4|\tau|)^m.$$

Evaluating the bivariate geometric series gives the result. $\square$ $\square$

### 4.4 Bounds for the derivative

We will now give a rigorous global bound for the magnitude of $W'$. Since we want to compute $W$ with small *relative* error, the estimate for $|W'(z)|$ should be close to optimal (within a small constant factor) anywhere, including near singularities. We did not obtain a single neat expression that covers $W_k(z)$ adequately for all $k$ and $z$, so a few case distinctions are made.

$W'$ like $W$ is a multivalued function, and whenever we fix a branch for $W$, we fix the corresponding branch for $W'$. Exactly on a branch cut, $W'$ is therefore finite (except at a branch point) and equal to the directional derivative taken along the branch cut, so we must deal with the branch cut discontinuity separately when bounding perturbations in $W$ if $z$ crosses the cut.

The derivative of the Lambert $W$ function can be written as

$$W'(z) = \frac{1}{(1 + W(z))e^{W(z)}} = \frac{1}{z} \frac{W(z)}{1 + W(z)}$$

where a limit needs to be taken in the rightmost expression for $W_0(z)$ near $z = 0$. The rightmost expression also shows that $W'(z) \approx 1/z$ when $|W(z)|$ is large. Bounding $|\operatorname{Im}(W_k(z))|$ from below gives the following.

**Theorem 5.** *For $|k| \geq 2$,*

$$|W_k'(z)| \leq \frac{1}{|z|}\frac{(2k-2)\pi}{(2k-2)\pi - 1} \leq \frac{1}{|z|}\frac{2\pi}{2\pi - 1} \leq \frac{1.2}{|z|}.$$

*Also, if $k = 1$ and $\mathrm{Im}(z) \geq 0$, or if $k = -1$ and $\mathrm{Im}(z) < 0$, then*

$$|W_k'(z)| \leq \frac{1}{|z|}\frac{\pi}{\pi - 1} \leq \frac{1.5}{|z|}.$$

For large $|z|$, the following two results are convenient.

**Theorem 6.** *If $|z| > e$, then for any $k$,*

$$|W_k'(z)| \leq \frac{1}{|z|}\frac{W_0(|z|)}{W_0(|z|) - 1}.$$

*Proof.* The inequality $|W_k(z)| \geq W_0(|z|)$ holds for all $z$ (this is easily proved from the inverse function relationship defining $W$), giving the result. □ □

**Theorem 7.** *If $|z| \geq \left(\frac{1}{2} + (2|k|+1)\pi\right) e^{-1/2}$, or more simply if $|z| \geq 4(|k|+1)$, then*

$$|W_k'(z)| \leq \frac{1}{|z|}.$$

*Proof.* Let $a = \mathrm{Re}(W_k(z))$. We have $|W_k(z)/(1 + W_k(z))| \leq 1$ when $a \geq -1/2$. If $a < -1/2$, then

$$|z| = |W_k(z)e^{W_k(z)}| < (|a| + (2|k|+1)\pi)e^a < (\tfrac{1}{2} + (2|k|+1)\pi)e^{-1/2}. \qquad \square$$

□

It remains to bound $|W_k'(z)|$ for $k \in \{-1, 0, 1\}$ in the cases where $z$ may be near the branch point at $-1/e$. This can be accomplished as follows.

**Theorem 8.** *For any $k$,*

$$|W_k'(z)| \leq \frac{1}{|z|} \max\left(3, \frac{1.5}{\sqrt{|ez+1|}}\right).$$

*Proof.* If $|W(z)+1| \geq 1/2$, then $|W(z)/(W(z)+1)| \leq 3$. Now consider the case $W(z)+1 = \varepsilon$ for some $|\varepsilon| \leq 1/2$. Then we must have $|ez+1| \leq |\varepsilon|^2$, due to the Taylor expansion

$$(-1+\varepsilon)e^{-1+\varepsilon} + e^{-1} = \frac{1}{e}\left(\frac{\varepsilon^2}{2} + \frac{\varepsilon^2}{3} + \frac{\varepsilon^2}{8} + \ldots\right).$$

This implies that

$$\left|\frac{W(z)}{W(z)+1}\right| = \frac{|\varepsilon - 1|}{|\varepsilon|} \leq \frac{1 + |\varepsilon|}{|\varepsilon|} \leq \frac{1.5}{\sqrt{|ez+1|}}. \qquad \square$$

□

Theorem 8 can be used in an implementation, provided that we use a different bound when $k = 0$ and $z \approx 0$ (also, when $z \approx -1/e$ and $W_k(z) \not\approx -1$). However, it is worth making a few case distinctions and slightly complicating the formulas to tighten the error propagation for $k = -1, 0, 1$. For these branches, we implement the following inequalities.

**Theorem 9.** *Let* $t = |ez + 1|$.

1. *If* $|z| \le 64$*, then*
$$|W_0'(z)| \le \frac{2.25}{\sqrt{t(1+t)}}.$$

2. *If* $|z| \ge 1$*, then*
$$|W_0'(z)| \le \frac{1}{|z|}.$$

3. *If* $\mathrm{Re}(z) \ge 0$*, or if* $\mathrm{Im}(z) < 0$ *when* $k = -1$ *(respectively* $\mathrm{Im}(z) \ge 0$ *when* $k = 1$*), then*
$$|W_{\pm 1}'(z)| \le \frac{1}{|z|}\left(1 + \frac{1}{4 + |z|^2}\right).$$

4. *For all* $z$*,*
$$|W_{\pm 1}'(z)| \le \frac{1}{|z|}\left(1 + \frac{23}{32}\frac{1}{\sqrt{t}}\right).$$

*Proof.* The inequalities can be verified by interval computations on a bounded region (since $1/|z|$ is an upper bound for sufficiently large $|z|$) excluding the neighborhoods of the branch points. These computations can be done by bootstrapping from Theorem 8: that is, we implement an initial version of the Lambert $W$ function using the weaker bound in Theorem 8, and then verify that the stronger bounds hold by repeated subdivision of relevant region. Close to $-1/e$, Theorem 1 applies, and an argument similar to that in Theorem 8 can be used close to 0. (We omit the computational details of such a verification which are straightforward but lengthy.) □ □

It is clearly possible to make the bounds sharper, not least by adding more case distinctions, but these formulas are sufficient for our purposes, easy to implement, and cheap to evaluate. The implementation requires only the extraction of lower or upper bounds of intervals and unsigned floating-point operations with directed rounding (assuming that $ez + 1$ has been computed using interval arithmetic).

# 5 Implementation of the alternative branch cuts

In the Arb implementation, the user can select the alternative functions $\mathcal{W}_k^{\mathrm{left}}(z)$ and $\mathcal{W}^{\mathrm{middle}}(z)$ by passing a special value in the *flags* field instead of the default value 0, namely

```
acb_lambertw(res, z, k, ACB_LAMBERTW_LEFT, prec)
acb_lambertw(res, z, k, ACB_LAMBERTW_MIDDLE, prec)
```

where $k = -1$ should be set in the second case.

We implement the alternative branch cuts essentially using the case distinctions in (1) and (2). More precisely, we split the input $z$ into $z_a = \mathrm{Re}(z) + (\mathrm{Im}(z) \cap [0, \infty))i$ and $z_b = \mathrm{Re}(z) + (-\mathrm{Im}(z) \cap [0, \infty))i$. If the standard branches to be taken below and above the cut have index $k$ and $k'$ respectively, then we compute $\mathcal{W}_k^{\mathrm{left}}(z)$ or $\mathcal{W}^{\mathrm{middle}}(z)$ as $W_k(z_a) \cup \overline{W_{-k'}(z_b)}$. Conjugation is used to get a continuous evaluation of $W_{k'}(\mathrm{Re}(z) + (\mathrm{Im}(z) \cap (-\infty, 0))i)$, in light of our convention to work with closed intervals and make the standard branches $W_k$ continuous from above on the cut.

We observe that for $\mathcal{W}^{\text{middle}}(z)$ the Puiseux expansion at $-1/e$ is valid in all directions, as is the asymptotic expansion at 0 with $L_1 = \log(-z)$ and $L_2 = \log(-L_1)$. Further, $\mathcal{W}_k^{\text{left}}(z)$ is given by the asymptotic expansion with $L_1 = \log(-z)+(2k+1)\pi i, L_2 = \log(L_1)$ when $|z| \to \infty$. These formulas could be used directly instead of case splitting where applicable.

# 6 Testing and benchmarks

| $z$ | 10 | 100 | 1000 | 10000 |
|---|---|---|---|---|
| 10 | $1.2 \cdot 10^{-6}$ (3.7) | $8.8 \cdot 10^{-6}$ (7.5) | $8.2 \cdot 10^{-5}$ (1.6) | 0.0042 (1.4) |
| $10^{10}$ | $1.2 \cdot 10^{-6}$ (3.6) | $8.1 \cdot 10^{-6}$ (6.8) | $8.0 \cdot 10^{-5}$ (1.6) | 0.0044 (1.4) |
| $10^{10^{20}}$ | $2.1 \cdot 10^{-6}$ (3.7) | $1.2 \cdot 10^{-5}$ (9.5) | $9.8 \cdot 10^{-5}$ (2.0) | 0.0047 (1.6) |
| $10i$ | $1.1 \cdot 10^{-5}$ (13.8) | $2.9 \cdot 10^{-5}$ (9.6) | $3.6 \cdot 10^{-4}$ (3.1) | 0.018 (2.9) |
| $-10^{10^{20}}$ | $4.7 \cdot 10^{-6}$ (3.9) | $1.0 \cdot 10^{-4}$ (32.0) | $5.7 \cdot 10^{-4}$ (7.4) | 0.017 (3.6) |
| $-1/e + 10^{-100}$ | $3.1 \cdot 10^{-6}$ (5.0) | $4.6 \cdot 10^{-6}$ (2.8) | $1.2 \cdot 10^{-4}$ (2.6) | 0.0057 (1.9) |
| $-1/e - 10^{-100}$ | $3.0 \cdot 10^{-6}$ (4.8) | $4.6 \cdot 10^{-6}$ (2.8) | $4.9 \cdot 10^{-4}$ (7.3) | 0.012 (3.2) |
| 10 (Mathematica) | $8.3 \cdot 10^{-5}$ (2.9) | $2.1 \cdot 10^{-4}$ (6.5) | 0.0013 (6.4) | 0.029 (3.3) |

Table 1: Time in seconds to compute $w = W_0(z)$ at a precision of 10, 100, 1000 and 10000 decimal digits with our implementation in Arb. The numbers in parentheses show the time relative to computing $e^w$ at the same precision. Bottom row: time for the Lambert $W$ function in Mathematica 10 (and time relative to the exponential function in Mathematica).

We have tested the implementation in Arb in various ways, most importantly to verify that correct inclusions are being computed, but also to make sure that output intervals are reasonably tight.

The test code included with the library generates overlapping random input intervals $z_1, z_2$ (sometimes placed very close to $-1/e$), computes $w_1 = W_k(z_1)$ and $w_2 = W_k(z_2)$ at different levels of precision (sometimes directly invoking the asymptotic expansion with a random number of terms instead of calling the main Lambert $W$ function implementation), checks that the intervals $w_1$ and $w_2$ overlap, and also checks that $w_1 e^{w_1}$ contains $z_1$. The conjugate symmetry is also tested. These checks together give a strong test of correctness.

We have also done separate tests to verify that the error bounds converge for exact floating-point input when the precision increases, and we have done further ad hoc tests for a variety of easy and difficult cases of the function.

Table 1 shows the time in seconds to evaluate $W_0(z)$ using `acb_lambertw` for different $z$ and for precision between 10 and $10^4$ digits. The table includes the relative timings compared to the exponential function (using `acb_exp`) with the output from the Lambert $W$ function as the argument. The timings were obtained using Arb 2.13 on an Intel i5-4300U CPU.

At low precision, the time to evaluate $W$ for a "normal" input $z$ is about $10^{-6}$ seconds when $W$ is real and $10^{-5}$ seconds when $W$ is complex. For instance, creating a 1000 by 1000 pixel domain coloring plot of $W_0(z)$ on $[-5, 5] + [-5, 5]i$ takes 12 seconds.

We observe that the Lambert $W$ function usually costs a small multiple of the time for an elementary function, as desired. The higher relative overhead of the Lambert

$W$ function in the complex-valued case compared to the real-valued case results in part from less optimized precision handling in the code for computing the initial floating-point approximation (which could be improved in a future version), and in part from overhead in the branch test.

The bottom row of Table 1 shows the time to evaluate $W_0(z)$ near $z = 10$ with Mathematica 10 on an Intel i7-4600U, using `N[ProductLog[z],d]` for $d$ digits. These timings were kindly provided by one of the referees. Arb is roughly an order of magnitude faster than Mathematica, mainly due to the underlying elementary functions. The fact that our algorithm for $W$ can make good use of the fast elementary functions in Arb [8] validates our approach.

### 6.1 Explicit values

We show the output (converted to decimal intervals using `acb_printn`) for a few of the test cases in the benchmark. For $z = 10$, the following results are computed at the respective levels of precision:

```
[1.745528003 +/- 3.82e-10]
[1.7455280027{...79 digits...}0778883075 +/- 4.71e-100]
[1.7455280027{...979 digits...}5792011195 +/- 1.97e-1000]
[1.7455280027{...9979 digits...}9321568319 +/- 2.85e-10000]
```

For $z = 10^{10^{20}}$, we get:

```
[2.302585093e+20 +/- 3.17e+10]
[230258509299404568354.9134111633{...59 digits...}5760752900 +/- 6.06e-80]
[230258509299404568354.9134111633{...959 digits...}8346041370 +/- 3.55e-980]
[230258509299404568354.9134111633{...9959 digits...}2380817535 +/- 6.35e-9980]
```

For $z = -1/e + 10^{-100}$, the input interval overlaps with the branch point at 10 and 100 digits, showing a potential small imaginary part in the output, but at higher precision the imaginary part is confirmed to be zero:

```
[-1.000 +/- 3.18e-5] + [+/- 2.79e-5]i
[-1.0000000000{...28 digits...}0000000000 +/- 3.81e-50] + [+/- 2.76e-50]i
[-0.9999999999{...929 digits...}9899904389 +/- 2.99e-950]
[-0.9999999999{...9929 digits...}9452369126 +/- 5.45e-9950]
```

## 7 Automatic differentiation

Finally, we consider the computation of derivatives $W^{(n)}$, or more generally $(W \circ f)^{(n)}$ for an arbitrary function $f$. That is, given a power series $f \in \mathbb{C}[[x]]$, we want to compute the power series $W(f)$ truncated to length $n + 1$.

The higher derivatives of $W$ can be calculated using recurrence relations as discussed by Corless *et al.*, but it is more efficient to use formal Newton iteration in the ring $\mathbb{C}[[x]]$ to solve the equation $we^w = f$. That is, given a power series $w_j$ correct to $n$ terms, we compute

$$w_{j+1} = w_j - \frac{w_j e^{w_j} - f}{e^{w_j} + w_j e^{w_j}}$$

which is correct to $2n$ terms. This approach allows us to compute the first $n$ derivatives of $W$ or $W \circ f$ (when the first $n$ derivatives of $f$ are given) in $O(\mathsf{M}(n))$ operations where

$\mathsf{M}(n)$ is the complexity of polynomial multiplication. With FFT based multiplication, we have $\mathsf{M}(n) = O(n \log n)$.

This algorithm has been implemented in Arb for real power series in the function `arb_poly_lambertw_series` and for complex power series in the function `acb_poly_lambertw_series`.

Since the low $n$ coefficients of $w_{j+1}$ and $w_j$ are identical mathematically, we simply copy these coefficients instead of performing the full subtraction (avoiding needless inflation of the interval enclosures). A further important optimization in this algorithm is to save the constant term $e_0 = [x^0]e^w$ so that $e^{w_j}$ can be computed as $e_0 e^{w_j - [x^0]w_j}$. This avoids a transcendental function evaluation, which is expensive and moreover can be ill-conditioned, leading to greatly inflated enclosures. The performance could be improved further by a constant factor by saving the partial Newton iterations done internally for power series division and exponentials.

The Newton iteration scheme appears to be reasonably numerically stable, permitting the evaluation of high order derivatives with modest extra precision. For example, computing $h(x) = W_0(e^{1+x})$ to order 10 001 at 256-bit precision takes 2.8 seconds, giving the coefficient $[x^{10000}]h(x)$ as

```
[-6.02283194399026390e-5717 +/- 5.56e-5735].
```

# 8 Discussion

We have presented a complete implementation of the Lambert $W$ function in complex interval arithmetic. A number of improvements could be pursued in future work.

Our algorithm is correct in the sense that it computes a validated enclosure for $W_k(z)$, absent any bugs in the code. It is also easy to see that the enclosures converge when the input intervals converge and the precision is increased accordingly (as long as a branch cut is not crossed), under the assumption that the floating-point approximation is computed accurately. However, we have made no attempt to prove that the floating-point approximation is computed accurately beyond the usual heuristic reasoning and experimental testing.

Although the focus is on interval arithmetic, we note that applying Ziv's strategy [19] allows us to compute floating-point approximations of $W_k(z)$ with certified correct rounding. This requires only a simple wrapper around the interval code without the need for separate analysis of floating-point rounding errors. Further error analysis would be needed to compute the Lambert $W$ function in floating-point arithmetic with correct rounding without relying on interval arithmetic internally. Such error analysis seems feasible for the real branches, but would likely be tedious for the complex branches.

The implementation in Arb is designed for arbitrary precision. For low precision, the main approximation is usually computed using `double` arithmetic, but the certification uses arbitrary-precision arithmetic which consumes the bulk of the time. Using validated `double` or `double-double` arithmetic for the certification would be significantly faster.

Our implementation uses the third-order Halley iteration for the numerical root-finding. Some speedup should be possible using a higher-order iterations. For example, Fukushima [6] uses a fifth-order method.

We use a first order bound based on $|W'(z)|$ for error propagation when $z$ is inexact. For wide $z$, more accurate bounds could be achieved using higher derivatives. Simple and tight bounds for $|W^{(n)}(z)|$ for small $n$ along the lines that we have presented for $|W'(z)|$ would be a useful addition to the literature. For very wide intervals $z$, optimal enclosures could be determined by evaluating $W$ at two or more points to find the extreme values. This is most easily done in the real case, but suitable monotonicity conditions could presumably be determined for complex variables as well.

## Acknowledgements

The author thanks the two referees, whose input greatly improved the article.

## References

[1] N. Agrawal, A. Bikineev, P. A. Bristow, M. Guazzone, C. Kormanyos, H. Holin, B. Lalande, J. Maddock, J. Murphy, J. Råde, G. Sewani, B. Sobotta, N. Thompson, T. van den Berg, D. Walker, and X. Zhang. BOOST C++ libraries: Lambert W function. `https://www.boost.org/doc/libs/develop/libs/math/doc/html/math_toolkit/lambert_w.html`, 2017.

[2] D. A. Barry, P. J. Culligan-Hensley, and S. J. Barry. Real values of the W-function. *ACM Transactions on Mathematical Software*, 21(2):161–171, 1995.

[3] F. Chapeau-Blondeau and A. Monir. Numerical evaluation of the Lambert W function and application to generation of generalized Gaussian noise with exponent 1/2. *IEEE Transactions on Signal Processing*, 50(9):2160–2165, 2002.

[4] R. M. Corless, G. H. Gonnet, D. E. G. Hare, D. J. Jeffrey, and D. E. Knuth. On the Lambert W function. *Advances in Computational Mathematics*, 5(1):329–359, 1996.

[5] F. N. Fitsch, R. E. Shafer, and W. P. Crowley. Algorithm 443: Solution of the transcendental equation $we^w = x$. *Communications of the ACM*, 16(2):123–124, 1973.

[6] T. Fukushima. Precise and fast computation of Lambert W-functions without transcendental function evaluations. *Journal of Computational and Applied Mathematics*, 244:77–89, 2013.

[7] D. J. Jeffrey. Celebrating 20 years of the Lambert W function. `http://www.apmaths.uwo.ca/~djeffrey/LambertW/LambertW.html`, 2016.

[8] F. Johansson. Efficient implementation of elementary functions in the medium-precision range. In *22nd IEEE Symposium on Computer Arithmetic*, ARITH22, pages 83–89, 2015.

[9] F. Johansson. Arb: efficient arbitrary-precision midpoint-radius interval arithmetic. *IEEE Transactions on Computers*, 66:1281–1292, 2017.

[10] F. Johansson et al. mpmath: a Python library for arbitrary-precision floating-point arithmetic (version 1.1). `http://mpmath.org`, 2018.

[11] G. A. Kalugin and D. J. Jeffrey. Convergence in C of series for the Lambert W function. *arXiv preprint arXiv:1208.0754*, 2012.

[12] P. W. Lawrence, R. M. Corless, and D. J. Jeffrey. Algorithm 917: complex double-precision evaluation of the Wright $\omega$ function. *ACM Transactions on Mathematical Software*, 38(3):20, 2012.

[13] Maplesoft. ProductLog – Maple Programming Help. `https://www.maplesoft.com/support/help/maple/view.aspx?path=LambertW`, 2019.

[14] R. E. Moore. *Methods and applications of interval analysis*. SIAM, 1979.

[15] F. J. Olver, D. W. Lozier, R. F. Boisvert, and C. W. Clark. *NIST Handbook of Mathematical Functions*. Cambridge University Press, 2010.

[16] J. van der Hoeven. Ball arithmetic. Technical report, HAL, 2009. `http://hal.archives-ouvertes.fr/hal-00432152/fr/`.

[17] D. Veberič. Lambert W function for applications in physics. *Computer Physics Communications*, 183(12):2622–2628, 2012.

[18] Wolfram Research. ProductLog – Wolfram Language Documentation. `https://reference.wolfram.com/language/ref/ProductLog.html`, 2019.

[19] A. Ziv. Fast evaluation of elementary mathematical functions with correctly rounded last bit. *ACM Transactions on Mathematical Software*, 17(3):410–423, 1991.