# A new stable method to compute mean value coordinates

Chiara Fuda · Kai Hormann

**Abstract**

The generalization of barycentric coordinates to arbitrary simple polygons with more than three vertices has been a subject of study for a long time. Among the different constructions proposed, mean value coordinates have emerged as a popular choice, particularly due to their suitability for the non-convex setting. Since their introduction, they have found applications in numerous fields, and several equivalent formulas for their evaluation have been presented in the literature. However, so far, there has been no study regarding their numerical stability. In this paper, we aim to investigate the numerical stability of the algorithms that compute mean value coordinates. We show that all the known methods exhibit instability in some regions of the domain. To address this problem, we introduce a new formula for computing mean value coordinates, explain how to implement it, and formally prove that our new algorithm provides a stable evaluation of mean value coordinates. We validate our results through numerical experiments.

**Citation Info**

*Journal*
Computer Aided Geometric Design

*Volume*
111, June 2024

*Article*
102310, 16 pages

*Note*
Proceedings of GMP

*DOI*
10.1016/j.cagd.2024.102310

## 1 Introduction

Mean value coordinates were initially introduced as a generalization of barycentric coordinates to polygons and polyhedra [2, 5, 8, 11]. Since then, they have emerged as a valuable tool in a wide range of domains, such as interpolation, curve and surface modelling in computer graphics, mesh parameterization, the finite element method, and various other fields. Moreover, they stand out for their capability to extend barycentric coordinates to the non-convex setting, unlike other commonly used coordinates. For more details about generalized barycentric coordinates, we refer to Hormann and Sukumar [9].

Let $P \subset \mathbb{R}^2$ be a simple planar polygon with $n \geq 3$ vertices $v_1, \ldots, v_n$ arranged in anticlockwise ordering and $v \in \mathbb{R}^2$ be an arbitrary point in the interior of $P$. The *mean value coordinates* [2] of $v$ with respect to $P$ are defined as

$$\lambda_i(v) = \frac{w_i(v)}{\sum_{j=1}^{n} w_j(v)}, \qquad w_i(v) = \frac{1}{r_i}\left(\tan\frac{\alpha_{i-1}}{2} + \tan\frac{\alpha_i}{2}\right), \qquad i = 1, \ldots, n, \tag{1}$$

with $\alpha_i \in (-\pi, \pi)$ denoting the signed angle at $v$ in the triangle $[v, v_i, v_{i+1}]$ and $r_i = \|v - v_i\|$. Note that indices are considered cyclically with respect to the range $[1, 2, \ldots, n]$; for example, $v_{n+1} = v_1$ and $\alpha_0 = \alpha_n$.

Floater [2] shows that the coordinate functions $\lambda_i$ form a *partition of unity*,

$$\sum_{i=1}^{n} \lambda_i(v) = 1$$

and can be used to express $v$ as an affine combination of the vertices of $P$,

$$\sum_{i=1}^{n} \lambda_i(v) v_i = v,$$

which is also referred to as the *barycentric property*. Besides these defining properties of generalized barycentric coordinates, they can be extended continuously to the boundary of $P$ and this extension satisfies the *Lagrange property*

$$\lambda_i(v_j) = \delta_{ij} = \begin{cases} 1, & i = j \\ 0, & i \neq j, \end{cases} \qquad i, j = 1, \ldots, n,$$

which makes them particularly useful for interpolating data given at the vertices of $P$. Moreover, they are actually well-defined for all $v \in \mathbb{R}^2 \setminus \partial P$, positive inside the kernel of $P$, invariant to similarity transformations of $P$, and their extension is linear along the edges of $P$ and smooth except at the vertices $v_i$, where it is only $C^0$ [8].

While many alternative formulas that are mathematically equivalent to (1) have been proposed in the literature over the years, a comprehensive investigation of their numerical stability is currently lacking. The aim of this paper is to change this and to discuss the numerical stability of algorithms that compute mean value coordinates. After reviewing the related work on mean value coordinates and recalling the various methods that have been introduced for their computation (Section 2), we show that each formulation can exhibit numerical instability in certain situations (Appendix C). To address this issue, we introduce a new formula for expressing mean value coordinates and explain how to properly implement it, so as to prevent potential numerical issues (Section 3). We then recall the mathematical definition of numerical stability, specifically in the case of mean value coordinates, and we prove that our new formula provides a stable way to compute the functions $\lambda_i$ (Section 4 and Appendix B). Finally, we validate our results with numerical experiments and compare the various methods both in terms of numerical stability and efficiency (Section 5).

## 2 Existing methods for computing the mean value coordinates

Floater et al. [4] note that mean value coordinates are a particular member of a family of *three-point coordinates* for convex polygons, which can be derived by normalizing a set of weight functions $w_i$ that each depend on three consecutive vertices of $P$. In this context, they show that mean value coordinates can be expressed as

$$\lambda_i(v) = \frac{w_i(v)}{\sum_{j=1}^{n} w_j(v)}, \qquad w_i(v) = \frac{r_{i-1}A_{i,i+1} - r_i A_{i-1,i+1} + r_{i+1}A_{i-1,i}}{2A_{i-1,i}A_{i,i+1}}, \qquad i = 1,\dots,n, \tag{2}$$

where $A_{i,j} = \det(v_i - v, v_j - v)/2$ denotes the signed area of the triangle $[v, v_i, v_j]$. The advantage of this formula over the original definition in (1) is that it avoids the computation of the angles $\alpha_i$ and that it gets by without the use of trigonometric functions.

While mean value coordinates were initially considered only for points inside the kernel of star-shaped polygons [2], Hormann and Floater [8] prove that they are well-defined for any $v \in \mathbb{R}$ and (sets of) arbitrary planar polygons without self-intersection. They also propose another way of evaluating mean value coordinates that avoids trigonometric functions. In particular, denoting the dot product of $v_i - v$ and $v_j - v$ by $D_{i,j} = (v_i - v)\cdot(v_j - v)$, using the half-angle formula for the tangent, $\tan(\alpha_i/2) = (1-\cos\alpha_i)/\sin\alpha_i$, and recalling that $D_{i,i+1} = r_i r_{i+1}\cos\alpha_i$ and $2A_{i,i+1} = r_i r_{i+1}\sin\alpha_i$, they conclude that the mean value coordinates in (1) can be written as

$$\lambda_i(v) = \frac{w_i(v)}{\sum_{j=1}^{n} w_j(v)}, \qquad w_i(v) = \frac{1}{r_i}\left(\frac{r_{i-1}r_i - D_{i-1,i}}{2A_{i-1,i}} + \frac{r_i r_{i+1} - D_{i,i+1}}{2A_{i,i+1}}\right), \qquad i = 1,\dots,n. \tag{3}$$

The advantage of implementing this formula over (2) is that it allows to easily "catch" the case when $v$ is on the boundary of $P$, say $v = (1-\mu)v_k + \mu v_{k+1}$ for some $\mu \in [0,1]$ and some $k \in \{1,\dots,n\}$, as this happens if and only if $A_{k,k+1} = 0$ and $D_{k,k+1} \le 0$. In this case, the mean value coordinates of $v$ are just $\lambda_k(v) = 1-\mu$, $\lambda_{k+1}(v) = \mu$, and $\lambda_i(v) = 0$ for $i \neq k, k+1$.

One potential problem with the formulas in (2) and (3) is that the coordinates $\lambda_i(v)$ are not well-defined if $A_{k,k+1} = 0$ for some $k$, that is, if $v$ is on the line supporting the edge $[v_k, v_{k+1}]$ of $P$. We can overcome this problem by using the alternative half-angle formula for the tangent, $\tan(\alpha_i/2) = \sin\alpha_i/(1+\cos\alpha_i)$, to obtain

$$\lambda_i(v) = \frac{w_i(v)}{\sum_{j=1}^{n} w_j(v)}, \qquad w_i(v) = \frac{1}{r_i}\left(\frac{2A_{i-1,i}}{r_{i-1}r_i + D_{i-1,i}} + \frac{2A_{i,i+1}}{r_i r_{i+1} + D_{i,i+1}}\right), \qquad i = 1,\dots,n. \tag{4}$$

This formula gives rise to an implementation that has the same advantages as the one derived from (3), but is well-defined even if $A_{k,k+1} = 0$ for some $k$.

All the formulas above have the limitation that they are not well-defined on the boundary of the polygon $P$ and, moreover, (2) and (3) can be used only if all $A_{i,i+1} \neq 0$. This motivated Floater [3] to introduce yet another formula for mean value coordinates, which is also valid on the boundary, namely

$$\lambda_i(v) = \frac{\hat{w}_i(v)}{\sum_{j=1}^{n} \hat{w}_j(v)}, \qquad \hat{w}_i(v) = \sigma_i \sqrt{r_{i-1}r_{i+1} - D_{i-1,i+1}} \prod_{j \neq i-1,i} \sqrt{r_j r_{j+1} + D_{j,j+1}}, \qquad i = 1,\dots,n, \tag{5}$$

where $\sigma_i \in \{+1,-1\}$ is a sign related to the weight function $\hat{w}_i$. Initially, this formula was presented without $\sigma_i$, which limits its applicability to points $v$ inside convex polygons, but Anisimov [1, Section 3.2.4] demonstrates

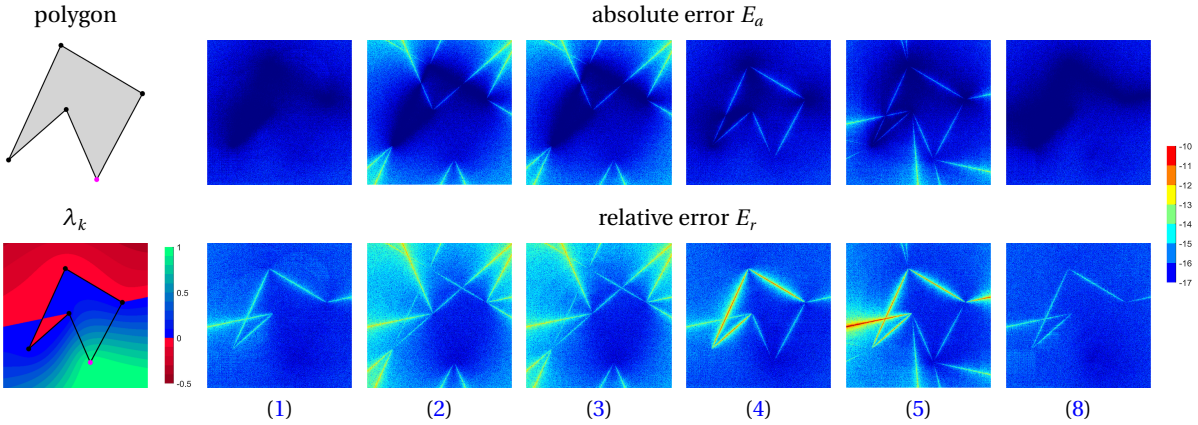


**Figure 1:** Plots of the absolute and relative errors on a $\log_{10}$ scale made by the algorithms that implement formulas (1)–(5) and (8) to evaluate the mean value coordinate $\lambda_k$ related to the vertex $v_k$ (magenta dot) for an arbitrary pentagon.

how to define $\sigma_i$, such that it can be used for any $v \in \mathbb{R}^2$ and arbitrary simple polygons. The disadvantage of this formula is that its implementation requires $O(n^2)$ instructions, while the formulas (1)–(4) give rise to $O(n)$ algorithms.

Let us now focus on understanding the circumstances under which the implementations of the formulas above may exhibit stability problems. One potential problem is the fact that they are rational, which can lead to the issue of vanishing denominators. This is actually not a problem for the $\lambda_i$, since the sum of the weights $w_i$ in (1)–(4) never vanishes for any $v \in \mathbb{R}^2 \setminus \partial P$ [8] and likewise for the sum of the weights $\hat{w}_i$ in (5). But what about the weights themselves? Considering some fixed $k \in \{1,\dots,n\}$, the weight $w_k$ in (1) or (4) is not well-defined, if either $\alpha_{k-1}$ or $\alpha_k$ is equal to $\pm\pi$, or if $v = v_k$, which happens only if $v$ lies on the edges $[v_{k-1}, v_k]$ or $[v_k, v_{k+1}]$. On the other hand, when computing $w_k$ with (2) or (3), we could potentially have problems even inside the polygon. In fact, the areas $A_{k-1,k}$ and $A_{k,k+1}$ vanish not only on the edges $[v_{k-1}, v_k]$ or $[v_k, v_{k+1}]$, but also on the entire lines supporting them. Based on this initial analysis, it is reasonable to expect that the computation of mean value coordinates is sensitive to rounding errors near the regions where they are not well-defined mathematically. Regarding instead the weight $\hat{w}_k$ in (5), even though it is well-defined for any $v \in \mathbb{R}^2$, problems can still arise, for example when subtracting two nearby numbers. This may happen if $D_{k-1,k+1}$ is approximately equal to $r_{k-1}\, r_{k+1}$ or if $D_{j,j+1}$ is close to $-r_j\, r_{j+1}$, for some $j \neq k-1, k$, that is, whenever $\alpha_{k-1} + \alpha_k$ is close to zero or some $\alpha_j$ approaches $\pm\pi$. In other words, we expect the weights $\hat{w}_k$ to be unstable when $v$ approaches the set $Z_k = \{v \in \mathbb{R}^2 : \hat{w}_k(v) = \lambda_k(v) = 0\}$, which consists of the edges that are not adjacent to $v_k$ and the line through $v_{k-1}$ and $v_{k+1}$, except for the (open) segment $(v_{k-1}, v_{k+1})$ itself.

To determine if such scenarios can indeed occur in practice, we examine the behaviour of the mean value coordinates for a specific polygon and visualize the numerical errors introduced by each of the previously mentioned formulas. For a given index $k \in \{1,\dots,n\}$, we compute the absolute error

$$E_a(v) = |\mathrm{fl}(\lambda_k(v)) - \lambda_k(v)|, \tag{6}$$

where $\lambda_k(v)$ is the "exact" value computed in multiple-precision (1024 bit) floating-point arithmetic using the MPFR library [6] and $\mathrm{fl}(\lambda_k(v))$ is the result of the standard double precision implementation. Let us consider the pentagon in Figure 1 and the index $k \in \{1,2,3,4,5\}$ of the vertex $v_k$ marked by the magenta dot. We examine the values $E_a(v)$ in (6) across a uniform grid of dimension $500 \times 500$ containing the polygon. The results are obtained for $\lambda_k(v)$ computed with all the formulas (1)–(5) and with our new formula (8), which will be introduced in Section 3. If $E_a(v)$ is on the order of the machine epsilon, which is approximately $10^{-16}$ in double precision, then it means that the method is stable for $v$, otherwise it suggests a potential instability.

The plots in Figure 1 show that the original formula in (1) seems to be the only one among the already known formulas that is stable everywhere. This outcome is particularly surprising, because it suggests that this method can effectively handle the division by small numbers in the weights $w_i$, $i = 1,\dots,5$, contrary to our initial expectations. Instead, computing $\lambda_k(v)$ with (2) or (3), we observe numerical issues around the lines supporting the edges, but not close to the edges themselves. While this partially aligns with our prediction of encountering issues along the entire lines, the theoretical analysis on the stability of these formulas explains why we do not have any problems near the edges. In particular, it turns out that the numerical errors introduced by (2) and (3) are bounded when $v$ approaches the edges, that is, when some $\alpha_i$

is close to $\pm\pi$ (see Corollaries 9 and 10). Then, as expected, the method resulting from (4) has numerical problems near the boundary of the polygon. Finally, formula (5) appears to be the least stable as it exhibits a substantial absolute error close to all sets $Z_i$, $i=1,\dots,5$, which also aligns with our initial considerations.

We further extend the stability analysis and also consider the relative error

$$E_r(v)=\frac{|\mathrm{fl}(\lambda_k(v))-\lambda_k(v)|}{|\lambda_k(v)|}, \tag{7}$$

which, although not mathematically defined on the set $Z_k$, provides valuable insights into the stability compared to the actual magnitude of $|\lambda_k|$. This metric offers a zoomed-in view of the domain region where $|\lambda_k|$ becomes notably small and indicates how close we can approach $Z_k$ before encountering significant relative errors. In fact, examining the values of $E_r$ in Figure 1, we observe that all methods exhibit relatively high errors in the vicinity of the set $Z_k$, but with different divergence rate. In particular, it appears that (1) may have potential instability over a wider region near $Z_k$ compared to (2) and (3), while (4) and (5) confirm to be the worst also in relative terms. Additionally, we observe that the relative error aligns with the information provided by the absolute error in the remaining part of the domain.

To summarize, the original formula (1) is the most robust in terms of numerical stability. In fact, despite some suggestions of instability close to $Z_k$ given by the relative error plot, the formula yields absolutely stable results across the entire domain. However, there exist situations where even the original formula (1) can be unstable, and we give an example in Section 5.

## 3 A new stable formula for mean value coordinates

After observing that all known methods for computing mean value coordinates have some flaw in terms of numerical stability, the goal of our work is to derive a new formula that is potentially stable everywhere. Like Floater [3], we aim to ensure that this new method is defined not only in the interior, but also along the boundary of the polygon. To achieve this, we first employ a similar trick and multiply both the numerator and denominator of the $\lambda_i$ in (1) by a common constant, which is then included into the redefined weights. In addition, we focus on minimizing operations that are more likely to introduce instability in the results, such as square roots and summations. We now present the new formula and explain how to implement it in a stable way.

**Theorem 1.** *The mean value coordinates can be expressed as*

$$\lambda_i(v)=\frac{\tilde{w}_i(v)}{\sum_{j=1}^n \tilde{w}_j(v)}, \qquad \tilde{w}_i=\sin\frac{\alpha_{i-1}+\alpha_i}{2}\prod_{j\neq i} r_j \prod_{j\neq i-1,i}\cos\frac{\alpha_j}{2}, \qquad i=1,\dots,n, \tag{8}$$

*and this formula is well-defined for all $v\in\mathbb{R}^2$, as long as the signed angle $\alpha_i$ is defined as $\pi$ or $-\pi$ for $v\in(v_i,v_{i+1})$ and in some arbitrary way for $v\in\{v_i,v_{i+1}\}$.*

*Proof.* Starting from (1), using the fact that $\tan(\alpha_i/2)=\sin(\alpha_i/2)/\cos(\alpha_i/2)$, and applying the angle sum identity for the sine function, we have

$$w_i=\sin\frac{\alpha_{i-1}+\alpha_i}{2}\Big/\left(r_i\cos\frac{\alpha_{i-1}}{2}\cos\frac{\alpha_i}{2}\right).$$

We can now eliminate the zeros in the denominator by multiplying all $w_i$ by $F=\prod_{i=1}^n r_i\cos(\alpha_i/2)$ and, denoting the result by $\tilde{w}_i$, we obtain the new formula (8).

This formula is well-defined for any $v\in\mathbb{R}^2\setminus\partial P$, because both the sum of the $w_i$ and $F$ do not vanish, and the denominator of $\lambda_i(v)$ in (8) is just $\sum_{j=1}^n \tilde{w}_j(v)=F\sum_{j=1}^n w_j(v)\neq 0$. Moreover, the formula also works if $v\in\partial P$. On the one hand, if $v$ is a vertex of $P$, that is, $v=v_k$ for some $k\in\{1,\dots,n\}$, then $r_k=0$ and $r_j\neq 0$ for $j\neq k$, so the only non-vanishing weight is $\tilde{w}_k$ and consequently $\lambda_k(v)=1$ and $\lambda_i(v)=0$ for $i\neq k$. On the other hand, if $v$ lies on an (open) edge of $P$, say $v=(1-\mu)v_k+\mu v_{k+1}$ for some $\mu\in(0,1)$ and some $k\in\{1,\dots,n\}$, then $\alpha_k=\pm\pi$, so that $\sin(\alpha/2)=\pm 1$ as well as $\cos(\alpha_k/2)=0$ and $\cos(\alpha_j/2)\neq 0$ for $j\neq k$. Therefore, all $\tilde{w}_i$ vanish, except for $\tilde{w}_k$ and $\tilde{w}_{k+1}$, which turn out to be

$$\tilde{w}_k=r_{k+1}S, \qquad \tilde{w}_{k+1}=r_k S, \qquad S=\sin\frac{\alpha_k}{2}\prod_{j\neq k,k+1} r_j\prod_{j\neq k}\cos\frac{\alpha_j}{2}.$$

Since $r_k=\mu e_k$ and $r_{k+1}=(1-\mu)e_k$, where $e_k=\|v_{k+1}-v_k\|$, it follows that $\lambda_k(v)=1-\mu$, $\lambda_{k+1}(v)=\mu$ and $\lambda_i(v)=0$ for $i\neq k,k+1$. □

**Algorithm 1** Stable implementation of formula (8) for computing the mean value coordinates $\lambda_1, \ldots, \lambda_n$

1: **function** MVC($v, v_1, \ldots, v_n$)
2: $W := 0$
3: **for** $i = 1, \ldots, n$ **do** ▷ indices are defined cyclically over $[1, \ldots, n]$, e.g., $v_{n+1} = v_1$
4: $\beta_i := \text{ANGLE}(v_{i+1} - v_i, v - v_i)$ ▷ ANGLE$((a_1, a_2), (b_1, b_2))$ returns ATAN2$(a_1 b_2 - a_2 b_1, a_1 b_1 + a_2 b_2)$
5: $\gamma_i := \text{ANGLE}(v_i - v_{i+1}, v - v_{i+1})$
6: $s_i := \beta_i + \gamma_i$
7: $r_i := \|v_i - v\|$
8: **for** $i = 1, \ldots, n$ **do**
9: $\alpha_{i-1,i+1} := \text{ANGLE}(v_{i-1} - v, v_{i+1} - v)$
10: $s_{i-1,i+1} := \pi \cdot [\text{sign}(s_{i-1}) + \text{sign}(s_i)] - s_{i-1} - s_i$ ▷ $s_{i-1,i+1} = \alpha_{i-1} + \alpha_i$
11: **if** $\text{sign}(\alpha_{i-1,i+1}) \neq \text{sign}(s_{i-1,i+1})$ **then** ▷ in this case, $\alpha_{i-1,i+1} = s_{i-1,i+1} - 2\pi \cdot \text{sign}(s_{i-1,i+1})$
12: $\alpha_{i-1,i+1} := -\alpha_{i-1,i+1}$ ▷ $\sin((\alpha_{i-1} + \alpha_i)/2) = \sin(-\alpha_{i-1,i+1}/2)$
13: $w_i := r_{i-1} \cdot \sin(\alpha_{i-1,i+1}/2)$
14: **for** $j = 1, \ldots, n$ **do**
15: **if** $j \neq i-1, i$ **then**
16: $w_i := w_i \cdot r_j \cdot \sin(|s_j|/2)$
17: $W := W + w_i$
18: **for** $i = 1, \ldots, n$ **do**
19: $\lambda_i := w_i / W$
20: return $\lambda_1, \ldots, \lambda_n$

Comparing our new formula in (8) to the one in (5), we observe that it also leads to an $O(n^2)$ algorithm for computing mean value coordinates, but we successfully eliminated all square roots, which can compromise the precision and the efficiency of the method, and we minimized the use of sum operations, as they can introduce numerical cancellation errors. In fact, the only sum in (8) is $\alpha_{i-1} + \alpha_i \in [-2\pi, 2\pi]$, but we can actually avoid computing this sum by noting that it is equal to the angle at $v$ in the triangle $[v, v_{i-1}, v_{i+1}]$, denoted by $\alpha_{i-1,i+1}$.

In our implementation (see Algorithm 1), we compute the signed angle $\theta$ between two vectors $a = (a_x, a_y)$ and $b = (b_x, b_y)$ using the ATAN2 function as $\theta = \text{ATAN2}(a_x b_y - a_y b_x, a_x b_x + a_y b_y) \in [-\pi, \pi]$. This is fine for all angles $\alpha_i$, but a bit more care is needed in the case of $\alpha_{i-1,i+1}$. Indeed, if $|\alpha_{i-1} + \alpha_i| > \pi$, then the ATAN2 function returns $\alpha_{i-1,i+1} = \alpha_{i-1} + \alpha_i - 2\pi \cdot \text{sign}(\alpha_{i-1} + \alpha_i)$. However, this "mismatch" by $\pm 2\pi$ is detected easily, because the signs of $\alpha_{i-1} + \alpha_i$ and $\alpha_{i-1,i+1}$ differ whenever it happens. And since $\sin((\alpha_{i-1} + \alpha_i)/2) = \sin(-\alpha_{i-1,i+1}/2)$ in this case, we can resolve this problem by changing the sign of $\alpha_{i-1,i+1}$ (cf. lines 11 and 12 in Algorithm 1). Note that the same problem can occur if $|\alpha_{i-1} + \alpha_i| = \pi$, because ATAN2 may return $\pi$ or $-\pi$ in this case, but it can be fixed in the same manner.

Yet, there might still be concerns related to the instability of the cosine function for arguments near zero. To prevent this, denoting by $\beta_i$ and $\gamma_i$ the signed angles at $v_i$ and $v_{i+1}$, respectively, in the triangle $[v, v_i, v_{i+1}]$, we use the fact that

$$\cos\frac{\alpha_i}{2} = \sin\frac{\pi - |\alpha_i|}{2} = \sin\frac{|\beta_i + \gamma_i|}{2}$$

(cf. line 16 in Algorithm 1) and recall that the sine function is stable for arguments near $\pi/2$. Note that computing the sum $s_i = \beta_i + \gamma_i$ is not a problem, because both angles are guaranteed to have the same sign, so that there is no risk of cancellation errors. The price for the improved stability is that we have to compute the $2n$ angles $\beta_i$ and $\gamma_i$ and their sums $s_i$. Note that we still need the angles $\alpha_i$ for determining whether it is necessary to change the sign of $\alpha_{i-1,i+1}$ or not, but once we know $\beta_i$ and $\gamma_i$ we can compute them as $\alpha_i = \pi \cdot \text{sign}(s_i) - s_i$ (cf. line 10 in Algorithm 1).

The numerical stability of this algorithm can be observed in Figure 1, which confirms that our new formula performs best, even if compared to the result using (1), especially close to the region $Z_k$.

So far, we have discussed the numerical stability of the different formulas and supported our claims only with empirical evidence. In the next section, we conduct a mathematical analysis on the numerical stability of mean value coordinates and provide a more formal explanation for our observations.

## 4 Theoretical analysis of the numerical stability

A common procedure to theoretically analyse the numerical stability of an algorithm is to establish an upper bound on the relative forward error and to examine its magnitude. In the specific context of mean value coordinates, we need to study the error $E_r$ in (7). To this end, we recall a result by Fuda et al. [7, Theorem 1], regarding an upper bound for any function that can be expressed in the form

$$r(x) = \frac{\sum_{i=0}^{n} a_i(x) f_i}{\sum_{j=0}^{m} b_j(x)} \tag{9}$$

for some data values $f_i$ and functions $a_i$ and $b_j$, $i = 0,\dots,n$ and $j = 0,\dots,m$. It is worth noting that bounding $E_r$ from above also gives an upper bound on the absolute error in (6), because

$$E_a(v) = E_r(v)|\lambda_k(v)|.$$

**Theorem 2.** *Suppose that there exist $\alpha_0,\dots,\alpha_n \in \mathbb{R}$ with*

$$\mathrm{fl}(a_i(x)) = a_i(x)(1+\alpha_i), \qquad |\alpha_i| \le A\epsilon + O(\epsilon^2), \qquad i = 0,\dots,n$$

*and $\beta_0,\dots,\beta_m \in \mathbb{R}$ with*

$$\mathrm{fl}(b_j(x)) = b_j(x)(1+\beta_j), \qquad |\beta_j| \le B\epsilon + O(\epsilon^2), \qquad j = 0,\dots,m$$

*for some constants $A$ and $B$. Then, assuming that the data $f_i$ are given as floating-point numbers, the relative forward error of $r$ in (9) satisfies*

$$\frac{|\mathrm{fl}(r(x)) - r(x)|}{|r(x)|} \le (n+2+A)\alpha(x)\epsilon + (m+B)\beta(x)\epsilon + O(\epsilon^2),$$

*where*

$$\alpha(x) = \frac{\sum_{i=0}^{n} |a_i(x) f_i|}{\left|\sum_{i=0}^{n} a_i(x) f_i\right|} \qquad \text{and} \qquad \beta(x) = \frac{\sum_{j=0}^{m} |b_j(x)|}{\left|\sum_{j=0}^{m} b_j(x)\right|},$$

*for $\epsilon$ small enough.*

We can use this result also for the mean value coordinates

$$\lambda_i(v) = \frac{w_i(v)}{\sum_{j=1}^{n} w_j(v)}, \qquad i = 1,\dots,n, \tag{10}$$

as their formula fits the expression in (9) for $n = 0$, $a_0 = w_i$, $f_0 = 1$, $m = n-1$ and $b_j = w_{i+1}$. Before proceeding, we note that this stability analysis does not account for any errors arising from the initial rounding of the given values to floating-point numbers.

**Corollary 3.** *Assume that there exist $\delta_1,\dots,\delta_n \in \mathbb{R}$ with*

$$\mathrm{fl}(w_i(v)) = w_i(v)(1+\delta_i), \qquad |\delta_i| \le D\epsilon + O(\epsilon^2), \qquad i = 1,\dots,n \tag{11}$$

*for some constant $D$. Then, assuming that the input values $v_i$ and $v$ are given as floating-point numbers, the relative forward error of the mean value coordinates in (10) satisfies*

$$\frac{|\mathrm{fl}(\lambda_i(v)) - \lambda_i(v)|}{|\lambda_i(v)|} \le (1+D)\epsilon + (n-1+D)W(v)\epsilon + O(\epsilon^2), \tag{12}$$

*where*

$$W(v) = \frac{\sum_{i=1}^{n} |w_i(v)|}{\left|\sum_{i=1}^{n} w_i(v)\right|}, \tag{13}$$

*for $\epsilon$ small enough.*

Hence, the numerical stability of the mean value coordinates depends on the constant $D$ and the function $W$. As the latter is the same for all the different formulas, what distinguishes their performance in terms of numerical stability is the upper bound $D$ on the relative error associated with the weights $w_i$. Considering the new formula, it can be proven that the constant $D$ related to the weights $\tilde{w}_i$ is always small (see Appendix B). In contrast, for all the other formulas, the related $D$ can be large (see Appendix C), which agrees with what we observed in Section 2.

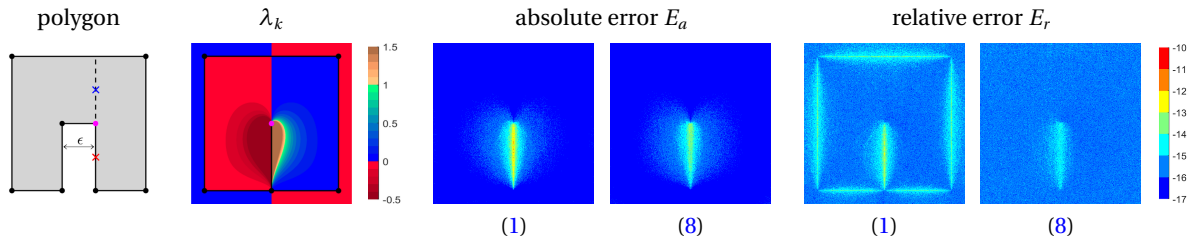


**Figure 2:** Plots of the absolute and relative errors on a $\log_{10}$ scale made by the original (1) and the new formula (8) to evaluate the mean value coordinate $\lambda_k$ related to the vertex $v_k$ (magenta dot) for the polygon on the left with $\epsilon = 0.0001$.

## 5 Numerical experiments

We investigated various examples to compare the different approaches for computing mean value coordinates based on the formulas in (1)–(5) and (8). Overall, we found that our new formula (8) consistently provides the most stable results, followed by the original formula (1), which usually performs much better than the other formulas and is often almost as stable as (8). However, as shown in Section 5.1, there are specific cases where our new Algorithm 1 beats the implementation of the original formula by a considerable margin. In Section 5.2, we further provide a comprehensive study of the efficiency of all methods. We implemented all algorithms with double precision in C++ and computed the "exact" values of $\lambda_k$ in multiple-precision (1024 bit) floating-point arithmetic using the MPFR library [6] for determining the relative and the absolute errors. All tests were run on a Windows 10 laptop with 1.8 GHz Intel Core i7-10510U processor and 16 GB RAM.

### 5.1 Stability comparison

Let us begin by comparing the performance of the original and the new formula for the 8-vertex polygon with vertices $(1,1)$, $(-1,1)$, $(-1,-1)$, $(-\epsilon,-1)$, $(-\epsilon,0)$, $(\epsilon,0)$, $(\epsilon,-1)$, and $(1,-1)$, shown in Figure 2 (left), where $\epsilon$ indicates the distance between the two vertical edges in the middle. Specifically, we investigate the case $\epsilon = 0.0001$ and turn our focus on the coordinate $\lambda_k$ associated with the vertex marked by the magenta dot. In the plots of the absolute error $E_a$ and the relative error $E_r$, which were computed on a uniform $500 \times 500$ grid that contains the polygon, we observe that problematic regions with numerical instability exist near the edges $[v_{k-1}, v_k]$ and $[v_k, v_{k+1}]$, but that Algorithm 1 handles them better. One reason for the relatively big errors is the function $W$ in (13), which influences the upper bound on the relative error in (12) for both formulas and obtains values on the order of $10^3$ in this region. The other reason is the constant $D$, which is about two orders of magnitude bigger for the formula in (1) than for our new formula in (8).

We also analysed the performance of the other formulas for this example, and Figure 3 (left) shows that

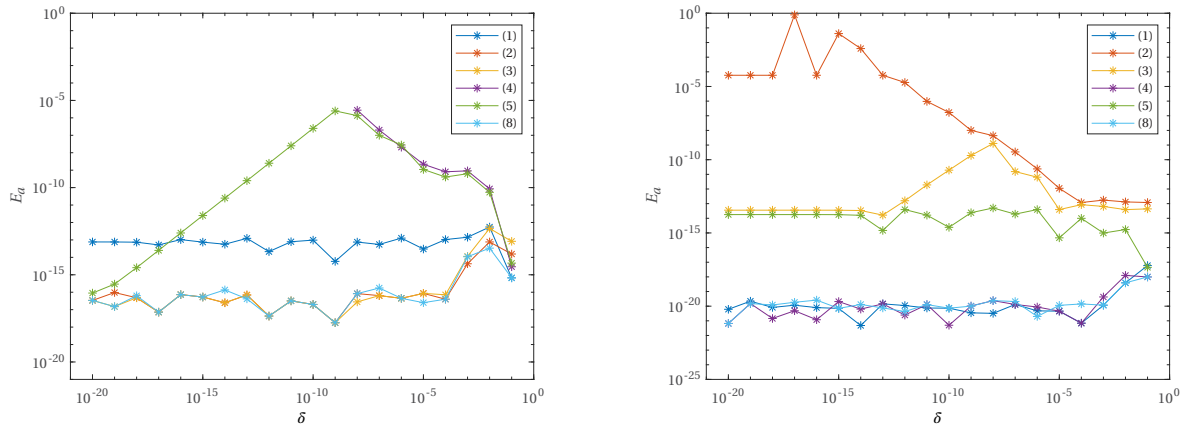


**Figure 3:** Comparison of the absolute errors on a log-log scale for computing $\lambda_k$ with the formulas (1)–(5) and (8) close to the points marked by the red cross (left) and the blue cross (right) in Figure 2. The plots show $E_a(v)$ for the different algorithms for $v$ at a horizontal distance of $\delta = 10^{-20}, 10^{-19}, \ldots, 10^{-1}$ from the considered points. Some values are not shown for very small $\delta$, because the algorithms return NaN as a result.

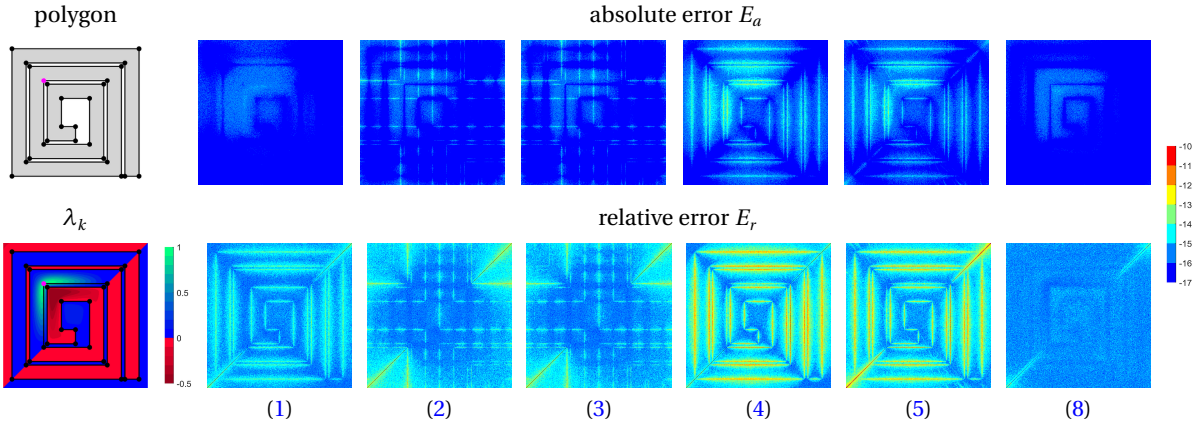


**Figure 4:** Same as Figure 1, but for a square spiral polygon.

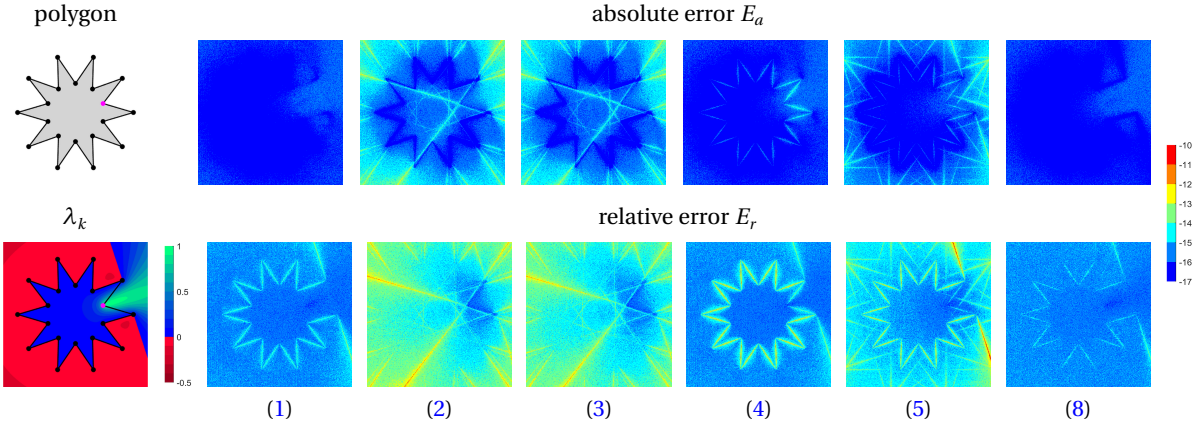


**Figure 5:** Same as Figure 1, but for a star-shaped polygon.

the implementations of (2) and (3) are as stable as Algorithm 1 close to the edge $[v_k, v_{k+1}]$. However, both formulas are very unstable close to the extension of this line, where instead the implementations of (1) and (4) are stable (see Figure 3, right). Interestingly, the worst case for our new formula, in terms of stability, does not occur extremely close to the edge $[v_k, v_{k+1}]$, but at a distance of about $10^{-2}$ to $10^{-3}$, which is again due to the behaviour of the function $W$ in (13), and similar for the formulas in (2) and (3). In contrast, the worst case for the formula in (5) happens at a distance of $10^{-8}$ to $10^{-9}$, that is, at roughly $\sqrt{\epsilon}$.

Figure 4 compares the errors of the different evaluation procedures for a square spiral polygon. As before, the plots show the absolute errors $E_a$ and the relative errors $E_r$ sampled on a uniform grid of $500 \times 500$ points that contains the polygon. Note that the black pixels in the lower left and the upper right of the relative error plots indicate points $v$ for which $E_r(v)$ is not well-defined, because $\lambda_k(v) = 0$. Otherwise, these plots confirm the behaviour that we already observed in Figure 1: the new formula (8) achieves the best result and the original one (1) is second-best, except close to the boundary of the polygon in relative terms. However, since $\lambda_k$ is very small in these regions, it makes more sense to focus on the absolute errors. These indicate that (1) and (8) produce very similar results, but still the new Algorithm 1 is better near the edges $[v_{k-1}, v_k]$ and $[v_k, v_{k+1}]$. As in Figure 1, we further note that (2) and (3) exhibit numerical instability along the extensions of the polygon's edges, especially for those related to $[v_{k-1}, v_k]$ and $[v_k, v_{k+1}]$, while (4) behaves similarly to (1), but with bigger errors close to the boundary. Finally, (5) is unstable in the vicinity of all sets $Z_i$. Figure 5 shows very similar results for a star-shaped polygon.

### 5.2 Efficiency comparison

To compare the efficiency of the different implementations, we conducted a first experiment using a set of 20 concave polygons, with an increasing number of vertices $n$, specifically with $n = 6i + 2$ for $i = 1, \ldots, 20$. The pattern of the polygons is shown in Figure 6 (left) for $i = 1, 2, 3$. The timings are obtained by evaluating the

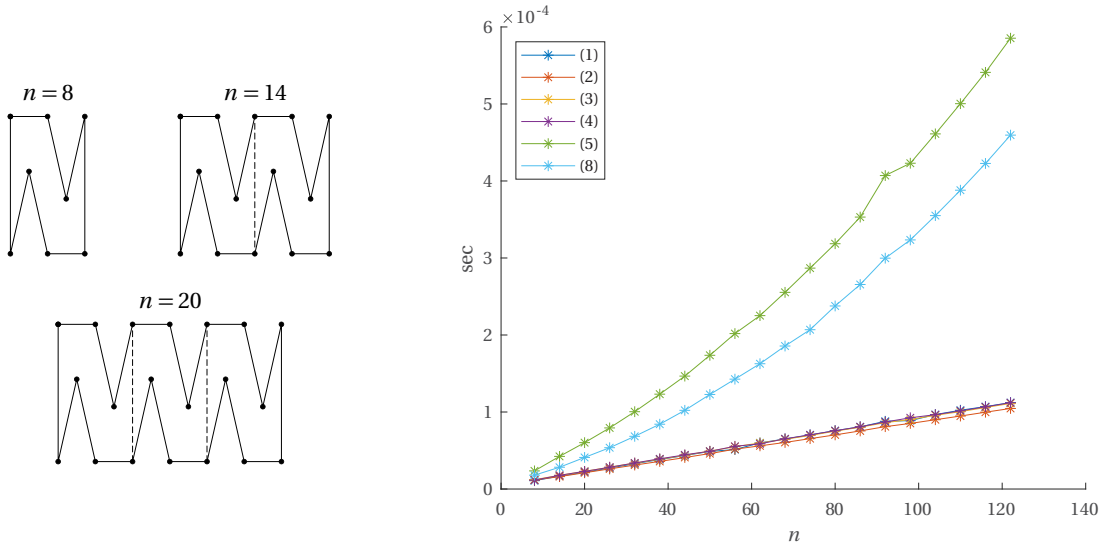


**Figure 6:** Average time in seconds (right) needed by the implementations of the formulas in (1)–(5) and (8) to evaluate all $n$ mean value coordinates for a concave test polygon (left) with $n = 6i + 2$ vertices for $i = 1, \ldots, 20$.

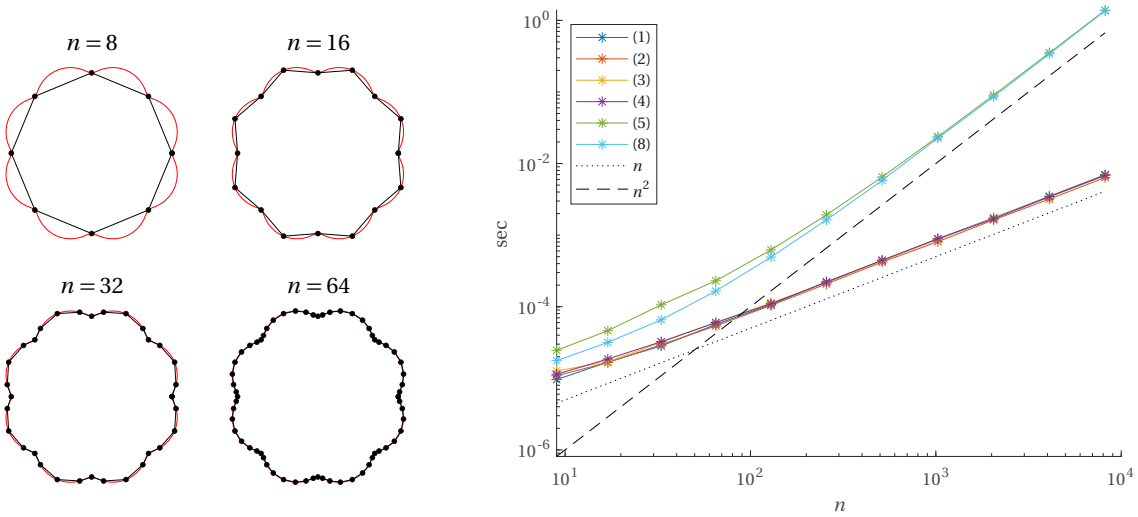


**Figure 7:** Average time in seconds (right) on a log-log scale for the implementations of the formulas in (1)–(5) and (8) to evaluate all $n$ mean value coordinates for a test polygon (left) inscribed to an epitrochoid (red curve) with $n = 2^i$ vertices for $i = 3, \ldots, 13$.

coordinates $\lambda_1, \ldots, \lambda_n$ at 90000 points and taking the average. The plots in Figure 6 (right) clearly indicate the linear time complexity of the algorithms derived from the formulas in (1)–(4) and the quadratic time complexity of the one that implements formula (5) as well as the new Algorithm 1, with the latter being roughly 25% faster. However, despite the unfavourable time complexity, the stable Algorithm 1 is at most twice as expensive as the linear-time algorithms for $n \leq 30$ and only about four times slower for $n = 100$.

In a second experiment, we focus on comparing the efficiency of the different methods for significantly larger values of $n$. Specifically, we construct the test polygons in Figure 7 (left) by sampling an epitrochoid curve at $n = 2^i$ points, $i = 3, \ldots, 13$, and then measure and plot (right) the average running time of all algorithms for computing all $n$ mean value coordinates at 100 evaluation points. In this setting, we gain a more comprehensive understanding of the asymptotic computational cost associated with the implementation of the various formulas, further confirming our previous observations. In fact, formulas (1)–(4) demonstrate a computational complexity of $O(n)$, while (5) and Algorithm 1 exhibit an asymptotic running time on the order of $O(n^2)$. To conclude, although (5) and Algorithm 1 have similar behaviour, our new implementation consistently proves to be faster in practice, especially for polygons with less than a thousand vertices.

## 6 Conclusion

Our investigations regarding the stable and efficient evaluation of mean value coordinates reveal the following, partially surprising insights. First, among the four formulas in (1)–(4), which give rise to efficient $O(n)$ algorithms, the original expression in (1) generally performs best in terms of stability and is as fast as the others. This is contrary to the common belief that using the ATAN2 function (for computing the angles $\alpha_i$) and the TAN function (for evaluating $\tan(\alpha_i/2)$) is slow. At least on our platform, we did not notice any computational disadvantage.

Second, the implementation of the original formula works well, even if $v$ is on one of the edges of the polygon, say $v = (1-\mu)v_k + \mu v_{k+1}$ for some $k$ and $\mu \in (0,1)$, despite the fact that $\alpha_k = \pm\pi$ in this case, hence $\tan(\alpha_k/2)$ is mathematically not well-defined. Since common floating-point implementations cannot represent $\pm\pi/2$ exactly, the TAN function does not return NAN in this case, but rather a number that is extremely big in absolute value, and the mean value coordinates $\lambda_i$ happen to be correct, up to machine precision, in the end. However, we observed major numerical problems for polygons with edges that are very close to each other (see Figure 2). In the vicinity of such edges, two of the values $\tan(\alpha_i/2)$ are very big, which eventually leads to a loss of precision.

Third, our new Algorithm 1 handles even such extreme cases and is generally the most stable of all methods. It also works if $v$ is a vertex of the polygon, a case that needs to be detected in the linear-time algorithms by checking if some $r_i$ equals zero. The only other method that does not require any exceptions for handling points on the boundary of the polygon is the one based on (5), but it turns out to be slower and less stable than our approach, especially near the sets $Z_i$.


### Acknowledgements

This work was supported by the Swiss National Science Foundation (SNSF) under project number 188577.


### References


[1] D. Anisimov. *Analysis and new constructions of generalized barycentric coordinates in 2D*. PhD thesis, Faculty of Informatics, Università della Svizzera italiana, May 2017.

[2] M. S. Floater. Mean value coordinates. *Computer Aided Geometric Design*, 20(1):19–27, Mar. 2003.

[3] M. S. Floater. Wachspress and mean value coordinates. In G. E. Fasshauer and L. L. Schumaker, editors, *Approximation Theory XIV: San Antonio 2013*, pages 81–102. Springer, Cham, 2014.

[4] M. S. Floater, K. Hormann, and G. Kós. A general construction of barycentric coordinates over convex polygons. *Advances in Computational Mathematics*, 24(1–4):311–331, Jan. 2006. [PDF]

[5] M. S. Floater, G. Kós, and M. Reimers. Mean value coordinates in 3D. *Computer Aided Geometric Design*, 22(7):623–631, Oct. 2005.

[6] L. Fousse, G. Hanrot, V. Lefèvre, P. Pélissier, and P. Zimmermann. MPFR: A multiple-precision binary floating-point library with correct rounding. *ACM Transactions on Mathematical Software*, 33(2):Article 13, 15 pages, June 2007.

[7] C. Fuda, R. Campagna, and K. Hormann. On the numerical stability of linear barycentric rational interpolation. *Numerische Mathematik*, 152(4):761–786, Dec. 2022. [PDF]

[8] K. Hormann and M. S. Floater. Mean value coordinates for arbitrary planar polygons. *ACM Transactions on Graphics*, 25(4):1424–1441, Oct. 2006. [PDF]

[9] K. Hormann and N. Sukumar, editors. *Generalized Barycentric Coordinates in Computer Graphics and Computational Mechanics*. Taylor & Francis, CRC Press, Boca Raton, 2017. ISBN 978-1-4987-6359-2.

[10] IEEE Computer Society. *IEEE Standard for Floating-Point Arithmetic*, July 2019. IEEE Std 754-2019 (Revision of IEEE Std 754-2008).

[11] T. Ju, S. Schaefer, and J. Warren. Mean value coordinates for closed triangular meshes. *ACM Transactions on Graphics*, 24(3):561–566, July 2005.

[12] S. Loosemore, R. M. Stallman, R. McGrath, A. Oram, and U. Drepper. Known maximum errors in math functions. In *The GNU C Library Reference Manual*, chapter 19.7, pages 561–601. 2023.

[13] NVIDIA. Mathematical functions. In *CUDA C++ Programming Guide, Release 12.4*, chapter 16, pages 373–383. Mar. 2024.

[14] S. M. Rump. Error bounds for computer arithmetics. In *26th IEEE Symposium on Computer Arithmetic*, ARITH-26, pages 1–14, Kyoto, June 2019.

[15] L. N. Trefethen and D. Bau. *Numerical Linear Algebra*. SIAM, Philadelphia, 1997. ISBN 978-0-89871-361-9.

## Appendix

This appendix provides the mathematical proof that computing mean value coordinates with the new formula (8) is always stable, as long as the function $W$ in (13) does not affect the upper bound in (12). In other words, we demonstrate that the constant $D$ in (11) related to the weights $\tilde{w}_i$ is always small (Appendix B). In addition, we show that the constants $D$ related to the weights $w_i$ in (1)–(4) and $\hat{w}_i$ in (5) cannot be bounded (Appendix C). Before going into these details, we present some preliminary facts that we later use in our analysis.

## A Preliminaries

We consider a computer that uses a set $\mathbb{F}$ of *floating-point numbers* with the corresponding *machine epsilon* $\epsilon$ and let $\mathrm{fl}\colon \mathbb{R} \to \mathbb{F}$ be the *rounding function* that maps each $x \in \mathbb{R}$ to the closest floating-point approximation $\mathrm{fl}(x) \in \mathbb{F}$. We recall [15, Lecture 13] that for any $x \in \mathbb{R}$, $x \neq 0$, the relative error is bounded from above by the machine epsilon $\epsilon$, or, equivalently, we can always find some $\delta \in \mathbb{R}$ with $|\delta| < \epsilon$, such that

$$\mathrm{fl}(x) = x(1+\delta).$$

The same holds for any arithmetic operation $* \in \{+, -, \times, \div\}$ between two arbitrary floating-point numbers $x, y \in \mathbb{F}$, that is, there exists some $\delta \in \mathbb{R}$ with $|\delta| < \epsilon$, such that

$$\mathrm{fl}(x * y) = (x * y)(1+\delta).$$

This property can also be extended to cases involving multiple operations, such as sums or products, where the upper bound on $|\delta|$ depends on the number of operations performed; for more detailed information, we refer the interested reader to Fuda et al. [7, Section 2].

Regarding instead the elementary function implementations in standard libraries, such as the trigonometric functions, we do not have general results on their numerical stability, but some libraries give information about the maximum relative errors in their specific implementations, such as the CUDA programming model [13] and the GNU library [12]. In our analysis, we investigate the numerical stability of the values $\tilde{w}_i$ by assuming that we have stable algorithms to evaluate the square root, the sine, and the arctangent functions. In other words, we assume that, for any $x \in \mathbb{F}$, there exist some $\delta_{\mathrm{sqrt}}, \delta_{\sin}, \delta_{\arctan}, \delta_{\tan} \in \mathbb{R}$, such that

$$\mathrm{fl}(\sqrt{x}) = \sqrt{x}(1+\delta_{\mathrm{sqrt}}), \qquad |\delta_{\mathrm{sqrt}}| \leq D_{\mathrm{sqrt}}\epsilon + O(\epsilon^2), \tag{14}$$

$$\mathrm{fl}(\sin x) = \sin x(1+\delta_{\sin}), \qquad |\delta_{\sin}| \leq D_{\sin}\epsilon + O(\epsilon^2), \tag{15}$$

$$\mathrm{fl}(\arctan x) = \arctan x(1+\delta_{\arctan}), \qquad |\delta_{\arctan}| \leq D_{\arctan}\epsilon + O(\epsilon^2) \tag{16}$$

$$\mathrm{fl}(\tan x) = \tan x(1+\delta_{\tan}), \qquad |\delta_{\tan}| \leq D_{\tan}\epsilon + O(\epsilon^2) \tag{17}$$

for some constants $D_{\mathrm{sqrt}}$, $D_{\sin}$,$D_{\arctan}$, and $D_{\tan}$. For example, for the IEEE standard 754 floating-point arithmetic [10], it is known [14] that $|\delta_{\mathrm{sqrt}}| \leq 1 - 1/\sqrt{1+2\epsilon}$, hence, by Taylor expansion, $D_{\mathrm{sqrt}} = 1$.

While the bounds in (14)–(16) assume that the argument $x$ is a floating-point number, let us now derive the bounds for an arbitrary argument $y \in \mathbb{R}$, which is first rounded to a floating-point value $z = \mathrm{fl}(y)$.

**Lemma 4.** *Let $z = y(1+\gamma) \in \mathbb{F}$, where $y \in \mathbb{R}$ and $\gamma \in \mathbb{R}$ satisfies $|\gamma| \leq C\epsilon$, for some $C > 0$, and $f$ be a differentiable function at $y$. If $f(y) \neq 0$, then there exists some $\gamma' \in \mathbb{R}$ such that*[1]

$$f(z) = f(y)(1+\gamma'), \qquad |\gamma'| \leq \frac{|f'(y)y|}{|f(y)|} C\epsilon + O(\epsilon^2).$$

*Proof.* The statement follows immediately from the Taylor expansion of $f$ around $y$, that is,

$$f(z) = f(y) + f'(y)y\gamma + O(\epsilon^2) = f(y)\left(1 + \frac{f'(y)y}{f(y)}\gamma + O(\epsilon^2)\right).$$

□

[1] Note that the quantity $|f'(y)y|/|f(y)|$ is the relative condition number $\kappa_f$ [15] of the function $f$.

**Corollary 5.** *Let $z = y(1+\gamma) \in \mathbb{F}$, where $y \in \mathbb{R}$ and $\gamma \in \mathbb{R}$ satisfies $|\gamma| \le C\epsilon$, for some $C > 0$. If $\sin y \neq 0$ and $\arctan y \neq 0$, then there exist some $\delta'_{\sin}, \delta'_{\arctan} \in \mathbb{R}$, such that*

$$\mathrm{fl}(\sin z) = \sin y(1+\delta'_{\sin}), \qquad |\delta'_{\sin}| \le (|\cot y||y|C + D_{\sin})\epsilon + O(\epsilon^2), \tag{18}$$

$$\mathrm{fl}(\arctan z) = \arctan y(1+\delta'_{\arctan}), \qquad |\delta'_{\arctan}| \le (C + D_{\arctan})\epsilon + O(\epsilon^2). \tag{19}$$

*If $y \notin \{(2k+1)\pi/2, k \in \mathbb{Z}\}$ and $\tan y \neq 0$, then there exists some $\delta'_{\tan} \in \mathbb{R}$, such that*

$$\mathrm{fl}(\tan z) = \tan y(1+\delta'_{\tan}), \qquad |\delta'_{\tan}| \le (2|y|/|\sin 2y|C + D_{\tan})\epsilon + O(\epsilon^2). \tag{20}$$

*If $y > 0$, then there exists some $\delta'_{\mathrm{sqrt}} \in \mathbb{R}$, such that*

$$\mathrm{fl}(\sqrt{z}) = \sqrt{y}(1+\delta'_{\mathrm{sqrt}}), \qquad |\delta'_{\mathrm{sqrt}}| \le (C/2 + D_{\mathrm{sqrt}})\epsilon + O(\epsilon^2). \tag{21}$$

*Proof.* Equations (18), (20) and (21) follow directly from (15), (17) and (14), respectively, and Lemma 4. Regarding (19), Lemma 4 and (16) give

$$\mathrm{fl}(\arctan z) = \arctan y(1+\delta'_{\arctan}), \qquad |\delta'_{\arctan}| \le \left(\left|\frac{y}{(1+y^2)\arctan y}\right| C + D_{\arctan}\right)\epsilon + O(\epsilon^2).$$

We note that $g(y) = y/[(1+y^2)\arctan y]$ is always positive, because $y$ and $\arctan y$ have the same sign. So, to complete the proof, it remains to show that $g(y) \le 1$ for all $y > 0$. The first derivative of $g$ is given by

$$g'(y) = \frac{\arctan y - y^2 \arctan y - y}{[(1+y^2)\arctan y]^2}.$$

Since $h(y) = \arctan y - y$ is a decreasing function, we have $h(y) < h(0) = 0$ and therefore $g'(y) < 0$. This means that $g$ is a strictly decreasing function. Additionally, we know that $\lim_{x\to 0} \arctan x/x = 1$ and conclude

$$g(y) < \lim_{x\to 0} g(x) = \lim_{x\to 0} \frac{x}{(1+x^2)\arctan x} = 1.$$

□

Finally, we consider two arbitrary vectors $a = (a_x, a_y)$ and $b = (b_x, b_y)$ and present the upper bounds on the relative forward errors of some quantities that we frequently use.

1. Considering the *radius* $r_i = \|v - v_i\|$ for some $i \in \{1, \ldots, n\}$, it follows from Theorem 2 and (21) that there exists some $\rho_i \in \mathbb{R}$, such that $\mathrm{fl}(r_i) = r_i(1+\rho_i)$ with

$$|\rho_i| \le (2 + D_{\mathrm{sqrt}})\epsilon + O(\epsilon^2), \qquad i = 1, \ldots, n. \tag{22}$$

2. Considering the *dot product* $D_{a,b} = a_x b_x + a_y b_y$ between $a$ and $b$, it follows from Theorem 2 that there exists some $\delta_{a,b} \in \mathbb{R}$, such that $\mathrm{fl}(D_{a,b}) = D_{a,b}(1+\delta_{a,b})$ with

$$|\delta_{a,b}| \le u(D_{a,b})\epsilon + O(\epsilon^2), \qquad u(D_{a,b}) = 4\frac{|a_x b_x| + |a_y b_y|}{|D_{a,b}|}. \tag{23}$$

   It is important to note that the relative forward error becomes unreliable when the computed quantity approaches zero, as dividing by a small value can result in a significantly large error. In such cases, the right quantity to consider is the absolute forward error, which is given by $|D_{a,b}\delta_{a,b}|$ and, since $|a_x b_x| + |a_y b_y| \le 2\|a\|\|b\|$, it is bounded from above by $8\|a\|\|b\|\epsilon + O(\epsilon^2)$. Hence, it is reasonable to expect that the computation of $D_{a,b}$ is generally stable, although its upper bound on the forward error may increase when the values $\|a\|$ and $\|b\|$ become large.

3. Considering the 2D *cross product* $C_{a,b} = (a_x b_y - a_x b_y)$ between $a$ and $b$, which is twice the signed area of the triangle $[0, a, b]$, it follows from Theorem 2 that there exists some $\gamma_{a,b} \in \mathbb{R}$, such that $\mathrm{fl}(C_{a,b}) = C_{a,b}(1+\gamma_{a,b})$ with

$$|\gamma_{a,b}| \le u(C_{a,b})\epsilon + O(\epsilon^2), \qquad u(C_{a,b}) = 4\frac{|a_x b_y| + |a_y b_x|}{|C_{a,b}|}. \tag{24}$$

   As in the case of the dot product, it may happen that this upper bound is big when the values $\|a\|$ and $\|b\|$ are large, but in general we assume that the computation of $C_{a,b}$ is stable.

4. Considering the *signed angle* $\theta_{a,b} = \arctan(D_{a,b}/C_{a,b})$ between $a$ and $b$, it follows from Theorem 2, the previous observations, and (19) that there exists some $\sigma_{a,b} \in \mathbb{R}$, such that $\mathrm{fl}(\theta_{a,b}) = \theta_{a,b}(1+\sigma_{a,b})$ with

$$|\sigma_{a,b}| \le u(\theta_{a,b})\epsilon + O(\epsilon^2), \qquad u(\theta_{a,b}) = u(D_{a,b}) + u(C_{a,b}) + 1 + D_{\arctan}. \tag{25}$$

## B Error analysis of the formula in (8)

We now observe that the weights $\tilde{w}_i(v)$ can be written in the general form

$$w(v)=\prod_{j=1}^{J}\sum_{k=1}^{K}x_{j,k}(v), \tag{26}$$

for some $J,K\in\mathbb{N}$. Thus, we first derive a general bound on the relative forward error for the function $w$ in (26) and then apply this result in the specific case of the weights $\tilde{w}_i$ in (8). It is worth noting that, since the expressions of the weights $w_i$ in (1)–(4) and $\hat{w}_i$ in (5) are all of the type (26), the result presented below can be applied to these methods as well (see Appendix C).

**Theorem 6.** *Suppose that there exist $\chi_{j,k}\in\mathbb{R}$, $j=1,\dots,J$ and $k=1,\dots,K$, with*

$$\mathrm{fl}(x_{j,k}(v))=x_{j,k}(v)(1+\chi_{j,k}),\qquad |\chi_{j,k}|\le X_{j,k}\epsilon+O(\epsilon^2),$$

*for some positive constants $X_{j,k}$, $j=1,\dots,J$ and $k=1,\dots,K$. Then there exists some $\delta\in\mathbb{R}$, such that $w$ in (26) satisfies $\mathrm{fl}(w(v))=w(v)(1+\delta)$ and $|\delta|\le D\epsilon+O(\epsilon^2)$, where*

$$D=\sum_{j=1}^{J}\frac{\sum_{k=1}^{K}|x_{j,k}(v)|(K-1+X_{j,k})}{\left|\sum_{k=1}^{K}x_{j,k}(v)\right|}+J-1.$$

*Proof.* We first notice that $\mathrm{fl}(w(v))$ is given by

$$\mathrm{fl}(w(v))=\prod_{j=1}^{J}\sum_{k=1}^{K}\left[x_{j,k}(v)(1+\chi_{j,k})(1+\delta_{j,k}^{+})\right](1+\delta^{\times}),$$

where $\delta_{j,k}^{+}$ and $\delta^{\times}$ are the relative errors introduced by the $K-1$ sums and the $J-1$ products, respectively, so they satisfy

$$|\delta_{j,k}^{+}|\le(K-1)\epsilon+O(\epsilon^2)\qquad\text{and}\qquad|\delta^{\times}|\le(J-1)\epsilon+O(\epsilon^2). \tag{27}$$

Consequently, there exist some $\eta_{j,k}\in\mathbb{R}$ with

$$|\eta_{j,k}|\le(K-1+X_{j,k})\epsilon+O(\epsilon^2),\qquad j=1,\dots,J,\quad k=1,\dots,K, \tag{28}$$

such that

$$\begin{aligned}\mathrm{fl}(w(v))&=\prod_{j=1}^{J}\sum_{k=1}^{K}\left[x_{j,k}(v)(1+\eta_{j,k})\right](1+\delta^{\times})=\prod_{j=1}^{J}\left[\sum_{k=1}^{K}x_{j,k}(v)\left(1+\frac{\sum_{k=1}^{K}x_{j,k}(v)\eta_{j,k}}{\sum_{k=1}^{K}x_{j,k}(v)}\right)\right](1+\delta^{\times})\\&=w(v)\left(1+\sum_{j=1}^{J}\frac{\sum_{k=1}^{K}x_{j,k}(v)\eta_{j,k}}{\sum_{k=1}^{K}x_{j,k}(v)}+\delta^{\times}+O(\epsilon^2)\right).\end{aligned}$$

Therefore, $\delta=\sum_{j=1}^{J}\sum_{k=1}^{K}x_{j,k}(v)\eta_{j,k}\Big/\sum_{k=1}^{K}x_{j,k}(v)+\delta^{\times}+O(\epsilon^2)$, and the statement follows immediately by using the triangle inequality, (27), and (28). □

**Corollary 7.** *For any $v\in\mathbb{F}^2$ and $v_1,\dots,v_n\in\mathbb{F}^2$, there exist $\delta_1,\dots,\delta_n\in\mathbb{R}$, such that the $\tilde{w}_i$ in (8) satisfy $\mathrm{fl}(\tilde{w}_i(v))=\tilde{w}_i(v)(1+\delta_i)$ and $|\delta_i|\le D\epsilon+O(\epsilon^2)$ for $i=1,\dots,n$, where*

$$D=\max_{i=1,\dots,n}\frac{\pi}{2}\left(u(\alpha_{i-1,i+1})+\sum_{j\neq i-1,i}\left(\max\{u(\beta_j),u(\gamma_j)\}+1\right)\right)+(n-1)(2+D_{\mathrm{sqrt}}+D_{\mathrm{sin}})+2n-3.$$

*Proof.* We note that the $\tilde{w}_i$ in (8) can be written as in (26) for $J=2n-2$, $K=1$ and

$$x_{j,1}=\begin{cases}\sin\frac{\alpha_{i-1,i+1}}{2}, & j=1\\ r_{j-1}, & j=2,\dots,i,\\ r_j, & j=i+1,\dots,n,\\ \sin\frac{\beta_{j-n}+\gamma_{j-n}}{2}, & j=n+1,\dots,n+i-2,\\ \sin\frac{\beta_{j-n+2}+\gamma_{j-n+2}}{2}, & j=n+i-1,\dots,2n-2.\end{cases}$$

It then follows from (18), (22), and (25) that $\mathrm{fl}(x_{j,1}) = x_{j,1}(1+\chi_{j,1})$ with $|\chi_{j,1}| \le X_{j,1}\epsilon + O(\epsilon^2)$ and

$$X_{j,1} = \begin{cases} \left|\cot\frac{\alpha_{i-1,i+1}}{2}\right|\left|\frac{\alpha_{i-1,i+1}}{2}\right| u(\alpha_{i-1,i+1}) + D_{\sin}, & j = 1, \\ 2 + D_{\mathrm{sqrt}}, & j = 2,\dots,n, \\ \left|\cot\frac{\beta_{j-n}+\gamma_{j-n}}{2}\right|\left|\frac{\beta_{j-n}+\gamma_{j-n}}{2}\right|\left(\max\{u(\beta_{j-n}), u(\gamma_{j-n})\}+1\right) + D_{\sin}, & j = n+1,\dots,n+i-2 \\ \left|\cot\frac{\beta_{j-n+2}+\gamma_{j-n+2}}{2}\right|\left|\frac{\beta_{j-n+2}+\gamma_{j-n+2}}{2}\right|\left(\max\{u(\beta_{j-n+2}), u(\gamma_{j-n+2})\}+1\right) + D_{\sin}, & j = n+i-1,\dots,2n-2. \end{cases}$$

Therefore, we can use Theorem 6 to get $\mathrm{fl}(\tilde{w}_i(v)) = \tilde{w}_i(v)(1+\delta_i)$ with $|\delta_i| \le D_i\epsilon + O(\epsilon^2)$ and

$$D_i = X_{1,1} + (n-1)(2+D_{\mathrm{sqrt}}) + \sum_{j=n+1}^{2n-2} X_{j,1} + 2n - 3.$$

Since $|x|/|\sin x| \le \pi/2$ for any $x \in [-\pi/2, \pi/2]$ and $\alpha_{i-1,i+1}, \beta_j + \gamma_j \in [-\pi/2, \pi/2]$, we get

$$D_i \le \frac{\pi}{2}\left(u(\alpha_{i-1,i+1}) + \sum_{j\ne i-1,i}\left(\max\{u(\beta_j), u(\gamma_j)\}+1\right)\right) + (n-1)D_{\sin} + (n-1)(2+D_{\mathrm{sqrt}}) + 2n - 3,$$

which proves the statement. □

## C Error analysis of the formulas in (1)–(5)

As mentioned before, also the weights $w_i$ in (1)–(4) and $\hat{w}_i$ in (5) are of type (26), so that we can apply Theorem 6 in the specific case of these formulas.

**Corollary 8.** *For any $v \in \mathbb{F}^2$ and $v_1,\dots,v_n \in \mathbb{F}^2$, there exist $\delta_1,\dots,\delta_n \in \mathbb{R}$, such that the $w_i$ in (1) satisfy $\mathrm{fl}(w_i(v)) = w_i(v)(1+\delta_i)$ and $|\delta_i| \le D\epsilon + O(\epsilon^2)$ for $i = 1,\dots,n$, where*

$$D = \max_{i=1,\dots,n} F_i(1 + \pi u(\alpha_i) + D_{\tan}) + 5 + D_{\mathrm{sqrt}} \tag{29}$$

*and*

$$F_i = \max_{v\in\mathbb{F}^2}\left|\sin\frac{\alpha_{i-1}+\alpha_i}{2}\cos\frac{\alpha_{i-1}}{2}\cos\frac{\alpha_i}{2}\right|^{-1}.$$

*Proof.* We note that $w_i$ in (1) can be written as in (26) for $J = K = 2$ and

$$x_{1,1} = \tan\frac{\alpha_{i-1}}{2}, \qquad x_{1,2} = \tan\frac{\alpha_i}{2},$$
$$x_{2,1} = \frac{1}{r_i}, \qquad x_{2,2} = 0.$$

It then follows from (22), (25), and (20) that $\mathrm{fl}(x_{j,k}) = x_{j,k}(1+\chi_{j,k})$ with $|\chi_{j,k}| \le X_{j,k}\epsilon + O(\epsilon^2)$ and

$$X_{1,1} = \frac{|\alpha_{i-1}|}{|\sin\alpha_{i-1}|}u(\alpha_{i-1}) + D_{\tan}, \qquad X_{1,2} = \frac{|\alpha_i|}{|\sin\alpha_i|}u(\alpha_i) + D_{\tan},$$
$$X_{2,1} = 3 + D_{\mathrm{sqrt}}, \qquad X_{2,2} = 0.$$

Therefore, we can use Theorem 6 to obtain $\mathrm{fl}(w_i(v)) = w_i(v)(1+\delta_i)$ with $|\delta_i| \le D_i\epsilon + O(\epsilon^2)$ and

$$\begin{aligned} D_i &= \frac{|x_{1,1}|(1+X_{1,1}) + |x_{1,2}|(1+X_{1,2})}{|x_{1,1}+x_{1,2}|} + X_{2,1} + 2 \\ &= \frac{\sum_{j=i-1,i}|\tan(\alpha_j/2)|\left(1 + \frac{|\alpha_j|}{|\sin\alpha_j|}u(\alpha_j) + D_{\tan}\right)}{|\tan(\alpha_{i-1}/2) + \tan(\alpha_i/2)|} + 5 + D_{\mathrm{sqrt}} \\ &\le \frac{\sum_{j=i-1,i}\frac{|\tan(\alpha_j/2)|}{|\sin\alpha_j|}(1 + |\alpha_j|u(\alpha_j) + D_{\tan})}{|\tan(\alpha_{i-1}/2) + \tan(\alpha_i/2)|} + 5 + D_{\mathrm{sqrt}} \\ &\le \frac{\sum_{j=i-1,i}\frac{|\tan(\alpha_j/2)|}{|\sin\alpha_j|}}{|\tan(\alpha_{i-1}/2) + \tan(\alpha_i/2)|}\left(1 + \pi\max\{u(\alpha_{i-1}), u(\alpha_i)\} + D_{\tan}\right) + 5 + D_{\mathrm{sqrt}}. \end{aligned}$$

Finally, we use the double-angle formula for the sine function and get

$$\frac{\sum_{j=i-1,i}\frac{|\tan(\alpha_j/2)|}{|\sin\alpha_j|}}{|\tan(\alpha_{i-1}/2)+\tan(\alpha_i/2)|}=\frac{1}{2}\frac{\cos^2(\alpha_{i-1}/2)+\cos^2(\alpha_i/2)}{|\sin(\alpha_{i-1}+\alpha_i)/2\cos(\alpha_{i-1}/2)\cos(\alpha_i/2)|}\le F_i,$$

which gives $D_i \le D$ for $D$ in (29). □

For the following statements, let $d_k = v_k - v$, $k = 1,\dots,n$ and let $C_{i,j}$ denote the cross product of $d_i$ and $d_j$, which we denoted by $C_{d_i,d_j}$ in Appendix A.

**Corollary 9.** *For any $v \in \mathbb{F}^2$ and $v_1,\dots,v_n \in \mathbb{F}^2$, there exist $\delta_1,\dots,\delta_n \in \mathbb{R}$, such that the $w_i$ in (2) satisfy $\mathrm{fl}(w_i(v)) = w_i(v)(1+\delta_i)$ and $|\delta_i| \le D\epsilon + O(\epsilon^2)$ for $i = 1,\dots,n$, where*

$$D=\max_{i=1,\dots,n}\frac{3}{4}F_i\Big(5+D_{\mathrm{sqrt}}+\max\big\{u(C_{i,i+1}),u(C_{i-1,i+1}),u(C_{i-1,i})\big\}\Big)+u(C_{i,i+1})+u(C_{i-1,i})+8 \tag{30}$$

*and*

$$F_i=\max_{v\in\mathbb{F}^2}\left|\sin\frac{\alpha_{i-1}+\alpha_i}{2}\sin\frac{\alpha_{i-1}}{2}\sin\frac{\alpha_i}{2}\right|^{-1}.$$

*Proof.* We note that the $w_i$ in (2) can be written as in (26) for $J = 3$, $K = 3$ and

$$\begin{aligned}
&x_{1,1}=r_{i-1}A_{i,i+1}, && x_{1,2}=-r_iA_{i-1,i+1}, && x_{1,3}=r_{i+1}A_{i-1,i},\\
&x_{2,1}=\frac{1}{A_{i-1,i}}, && x_{2,2}=0, && x_{2,3}=0,\\
&x_{3,1}=\frac{1}{A_{i,i+1}}, && x_{3,2}=0, && x_{3,3}=0.
\end{aligned}$$

It then follows from (22) and (24) that $\mathrm{fl}(x_{j,k}) = x_{j,k}(1+\chi_{j,k})$ with $|\chi_{j,k}| \le X_{j,k}\epsilon + O(\epsilon^2)$ and

$$\begin{aligned}
&X_{1,1}=3+D_{\mathrm{sqrt}}+u(C_{i,i+1}), && X_{1,2}=3+D_{\mathrm{sqrt}}+u(C_{i-1,i+1}), && X_{1,3}=3+D_{\mathrm{sqrt}}+u(C_{i-1,i}),\\
&X_{2,1}=1+u(C_{i-1,i}), && X_{2,2}=0, && X_{2,3}=0,\\
&X_{3,1}=1+u(C_{i,i+1}), && X_{3,2}=0, && X_{3,3}=0.
\end{aligned}$$

Therefore, we can use Theorem 6 to obtain $\mathrm{fl}(w_i(v)) = w_i(v)(1+\delta_i)$ with $|\delta_i| \le D_i\epsilon + O(\epsilon^2)$ and

$$\begin{aligned}
D_i&=\frac{|x_{1,1}|(2+X_{1,1})+|x_{1,2}|(2+X_{1,2})+|x_{1,3}|(2+X_{1,3})}{|x_{1,1}+x_{1,2}+x_{1,3}|}+X_{2,1}+X_{3,1}+6\\
&\le\frac{\sum_{k=1,3}|x_{1,k}|}{\left|\sum_{k=1,3}x_{1,k}\right|}\Big(5+D_{\mathrm{sqrt}}+\max\big\{u(C_{i,i+1}),u(C_{i-1,i+1}),u(C_{i-1,i})\big\}\Big)+u(C_{i,i+1})+u(C_{i-1,i})+8.
\end{aligned}$$

Finally, we use some trigonometric identities to obtain

$$\begin{aligned}
\frac{\sum_{k=1,3}|x_{1,k}|}{\left|\sum_{k=1,3}x_{1,k}\right|}&=\frac{|\sin\alpha_{i-1}|+|\sin(\alpha_{i-1}+\alpha_i)|+|\sin\alpha_i|}{|\sin\alpha_{i-1}-\sin(\alpha_{i-1}+\alpha_i)+\sin\alpha_i|}\\
&\le\frac{3}{|\sin\alpha_{i-1}-\sin(\alpha_{i-1}+\alpha_i)+\sin\alpha_i|}\\
&=\frac{3}{4|\sin((\alpha_i+\alpha_{i-1})/2)\sin(\alpha_{i-1}/2)\sin(\alpha_i/2)|}=\frac{3}{4}F_i,
\end{aligned}$$

which gives $D_i \le D$ for $D$ in (30). □

**Corollary 10.** *For any $v \in \mathbb{F}^2$ and $v_1,\dots,v_n \in \mathbb{F}^2$, there exist $\delta_1,\dots,\delta_n \in \mathbb{R}$, such that the $w_i$ in (3) and (4) satisfy $\mathrm{fl}(w_i(v)) = w_i(v)(1+\delta_i)$ and $|\delta_i| \le D\epsilon + O(\epsilon^2)$ for $i = 1,\dots,n$, where*

$$D=\max_{i=1,\dots,n}\left(F_i\max_{j=i-1,i}\Big(1+\max\big\{7+2D_{\mathrm{sqrt}},2+u(D_{j,j+1})\big\}+u(C_{j,j+1})\Big)\right)+5+D_{\mathrm{sqrt}} \tag{31}$$

*with*

$$F_i = \max_{v\in\mathbb{F}^2} \begin{cases} \left|\sin\frac{\alpha_{i-1}+\alpha_i}{2}\sin\frac{\alpha_{i-1}}{2}\sin\frac{\alpha_i}{2}\right|^{-1}, & \text{for } w_i \text{ in (3)},\\ 2\left|\sin\frac{\alpha_{i-1}+\alpha_i}{2}\cos\frac{\alpha_{i-1}}{2}\cos\frac{\alpha_i}{2}(1+\cos\alpha_{i-1})(1+\cos\alpha_i)\right|^{-1}, & \text{for } w_i \text{ in (4)}. \end{cases}$$

*Proof.* The proof is carried out for the computation of $w_i(v)$ with formula (3), but similar arguments can be applied to the case of the weights $w_i(v)$ in (4).

We note that $w_i$ in (3) can be written as in (26) for $J = K = 2$ and

$$x_{1,1} = \frac{r_{i-1}r_i - D_{i-1,i}}{2A_{i-1,i}}, \qquad x_{1,2} = \frac{r_i r_{i+1} - D_{i,i+1}}{2A_{i,i+1}},$$
$$x_{2,1} = \frac{1}{r_i}, \qquad x_{2,2} = 0.$$

It then follows from Theorem 2 and (22)–(24) that $\mathrm{fl}(x_{j,k}) = x_{j,k}(1+\chi_{j,k})$ with $|\chi_{j,k}| \le X_{j,k}\epsilon + O(\epsilon^2)$ and

$$X_{1,1} = \frac{r_{i-1}r_i(7+2D_{\mathrm{sqrt}}) + |D_{i-1,i}|(2+u(D_{i-1,i}))}{|r_{i-1}r_i - D_{i-1,i}|} + u(C_{i-1,i}),$$
$$X_{1,2} = \frac{r_i r_{i+1}(7+2D_{\mathrm{sqrt}}) + |D_{i,i+1}|(2+u(D_{i,i+1}))}{|r_i r_{i+1} - D_{i,i+1}|} + u(C_{i,i+1}),$$
$$X_{2,1} = 3 + D_{\mathrm{sqrt}},$$
$$X_{2,2} = 0.$$

Therefore, we can use Theorem 6 to obtain $\mathrm{fl}(w_i(v)) = w_i(v)(1+\delta_i)$ with $|\delta_i| \le D_i\epsilon + O(\epsilon^2)$ and

$$\begin{aligned} D_i &= \frac{|x_{1,1}|(1+X_{1,1}) + |x_{1,2}|(1+X_{1,2})}{|x_{1,1}+x_{1,2}|} + X_{2,1} + 2\\ &= \frac{\displaystyle\sum_{j=i-1,i}\left|\frac{r_j r_{j+1} - D_{j,j+1}}{2A_{j,j+1}}\right|\left(1 + \frac{|r_j r_{j+1}|(7+2D_{\mathrm{sqrt}}) + |D_{j,j+1}|(2+u(D_{j,j+1}))}{|r_j r_{j+1} - D_{j,j+1}|} + u(C_{j,j+1})\right)}{|(r_{i-1}r_i - D_{i-1,i})/(2A_{i-1,i}) + (r_i r_{i+1} - D_{i,i+1})/(2A_{i,i+1})|} + 5 + D_{\mathrm{sqrt}}\\ &\le \frac{\displaystyle\sum_{j=i-1,i}\frac{r_j r_{j+1} + |D_{j,j+1}|}{2|A_{j,j+1}|}\Big(1 + \max\{7+2D_{\mathrm{sqrt}}, 2+u(D_{j,j+1})\} + u(C_{j,j+1})\Big)}{|(r_{i-1}r_i - D_{i-1,i})/(2A_{i-1,i}) + (r_i r_{i+1} - D_{i,i+1})/(2A_{i,i+1})|} + 5 + D_{\mathrm{sqrt}}\\ &= \frac{\displaystyle\sum_{j=i-1,i}\frac{1+|\cos\alpha_j|}{|\sin\alpha_j|}\Big(1 + \max\{7+2D_{\mathrm{sqrt}}, 2+u(D_{j,j+1})\} + u(C_{j,j+1})\Big)}{|(1-\cos\alpha_{i-1})/\sin\alpha_{i-1} + (1-\cos\alpha_i)/\sin\alpha_i|} + 5 + D_{\mathrm{sqrt}}. \end{aligned}$$

Finally, we use some trigonometric identities and observe that

$$\begin{aligned} \frac{\displaystyle\sum_{j=i-1,i}\frac{1+|\cos\alpha_j|}{|\sin\alpha_j|}}{|(1-\cos\alpha_{i-1})/\sin\alpha_{i-1} + (1-\cos\alpha_i)/\sin\alpha_i|} &\le \frac{4}{|\sin\alpha_{i-1} + \sin\alpha_i - \sin(\alpha_{i-1}+\alpha_i)|}\\ &= \frac{1}{|\sin((\alpha_i+\alpha_{i-1})/2)\sin(\alpha_{i-1}/2)\sin(\alpha_i/2)|} = F_i, \end{aligned}$$

which gives $D_i \le D$ for $D$ in (31). □

**Corollary 11.** *For any $v \in \mathbb{F}^2$ and $v_1, \dots, v_n \in \mathbb{F}^2$, there exist $\delta_1, \dots, \delta_n \in \mathbb{R}$, such that the $\hat{w}_i$ in (5) satisfy $\mathrm{fl}(\hat{w}_i(v)) = \hat{w}_i(v)(1+\delta_i)$ and $|\delta_i| \le D\epsilon + O(\epsilon^2)$ for $i = 1, \dots, n$, where*

$$D = \max_{i=1,\dots,n}\left(F_i \max\left\{7+2D_{\mathrm{sqrt}}, 2+u(D_{i-1,i+1}), 2+\max_{j\ne i-1,i} u(D_{j,j+1})\right\}\right) + (n-1)D_{\mathrm{sqrt}} + n - 2 \tag{32}$$

*and*

$$F_i = \max_{v\in\mathbb{F}^2}\left(|1-\cos(\alpha_{i-1}+\alpha_i)|^{-1} + \sum_{j\neq i-1,i} |1+\cos\alpha_j|^{-1}\right).$$

*Proof.* We note that the $\hat{w}_i$ in (5), neglecting the signs $\delta_i$, can be written as in (26) for $J = n-1$, $K = 1$ and

$$x_{j,1} = \begin{cases} \sqrt{r_{i-1}r_{i+1} - D_{i-1,i+1}}, & j = 1 \\ \sqrt{r_{j-1}r_j + D_{j-1,j}}, & j = 2,\dots,i-1, \\ \sqrt{r_j r_{j+1} + D_{j,j+1}}, & j = i+1,\dots,n. \end{cases}$$

It then follows from Theorem 2 and (21)–(23) that $\mathrm{fl}(x_{j,1}) = x_{j,1}(1+\chi_{j,1})$ with $|\chi_{j,1}| \le X_{j,1}\epsilon + O(\epsilon^2)$ and

$$X_{j,1} = \begin{cases} \dfrac{r_{i-1}r_{i+1}(7+2D_{\mathrm{sqrt}}) + |D_{i-1,i+1}|(2+u(D_{i-1,i+1}))}{2|r_{i-1}r_{i+1} - D_{i-1,i+1}|} + D_{\mathrm{sqrt}}, & j = 1, \\ \dfrac{r_{j-1}r_j(7+2D_{\mathrm{sqrt}}) + |D_{j-1,j}|(2+u(D_{j-1,j}))}{2|r_{j-1}r_j + D_{j-1,j}|} + D_{\mathrm{sqrt}}, & j = 2,\dots,i-1, \\ \dfrac{r_j r_{j+1}(7+2D_{\mathrm{sqrt}}) + |D_{j,j+1}|(2+u(D_{j,j+1}))}{2|r_j r_{j+1} + D_{j,j+1}|} + D_{\mathrm{sqrt}}, & j = i+1,\dots,n. \end{cases}$$

Therefore, we can use Theorem 6 to get $\mathrm{fl}(\hat{w}_i(v)) = \hat{w}_i(v)(1+\delta_i)$ with $|\delta_i| \le D_i\epsilon + O(\epsilon^2)$ and

$$\begin{aligned} D_i &= \sum_{j=1}^{n} X_{j,1} + n - 2 \\ &\le \frac{r_{i-1}r_{i+1} + |D_{i-1,i+1}|}{2|r_{i-1}r_{i+1} - D_{i-1,i+1}|} \max\left\{7+2D_{\mathrm{sqrt}}, 2+u(D_{i-1,i+1})\right\} \\ &\quad + \sum_{j\neq i-1,i} \frac{r_j r_{j+1} + |D_{j,j+1}|}{2|r_j r_{j+1} + D_{j,j+1}|} \max\left\{7+2D_{\mathrm{sqrt}}, 2+u(D_{j,j+1})\right\} + (n-1)D_{\mathrm{sqrt}} + n - 2 \\ &= \frac{1+|\cos(\alpha_{i-1}+\alpha_i)|}{2|1-\cos(\alpha_{i-1}+\alpha_i)|} \max\left\{7+2D_{\mathrm{sqrt}}, 2+u(D_{i-1,i+1})\right\} \\ &\quad + \sum_{j\neq i-1,i} \frac{1+|\cos\alpha_j|}{2|1+\cos\alpha_j|} \max\left\{7+2D_{\mathrm{sqrt}}, 2+u(D_{j,j+1})\right\} + (n-1)D_{\mathrm{sqrt}} + n - 2 \\ &\le \frac{\max\left\{7+2D_{\mathrm{sqrt}}, 2+u(D_{i-1,i+1})\right\}}{|1-\cos(\alpha_{i-1}+\alpha_i)|} + \sum_{j\neq i-1,i} \frac{\max\left\{7+2D_{\mathrm{sqrt}}, 2+u(D_{j,j+1})\right\}}{|1+\cos\alpha_j|} + (n-1)D_{\mathrm{sqrt}} + n - 2, \end{aligned}$$

which proves the statement. □

The important difference between the constants $D$ in the upper bounds on the relative errors of the weights in Corollaries 8–11, compared to $D$ in Corollary 7, is that the former all depend on $F_i$. In all cases, $F_i$ is the maximum, over the finite set $\mathbb{F}^2$, of some function that diverges to infinity, either at the edges of $P$, along the lines that support them, or at the sets $Z_i$, which explains the big relative errors of the mean value coordinates $\lambda_i$ close to those regions. Surprisingly, for the original formula in (1), the potentially big relative errors of the weights usually cancel out "magically" during the normalization and do not affect the relative errors of the $\lambda_i$, except in the cases discussed in Section 5.