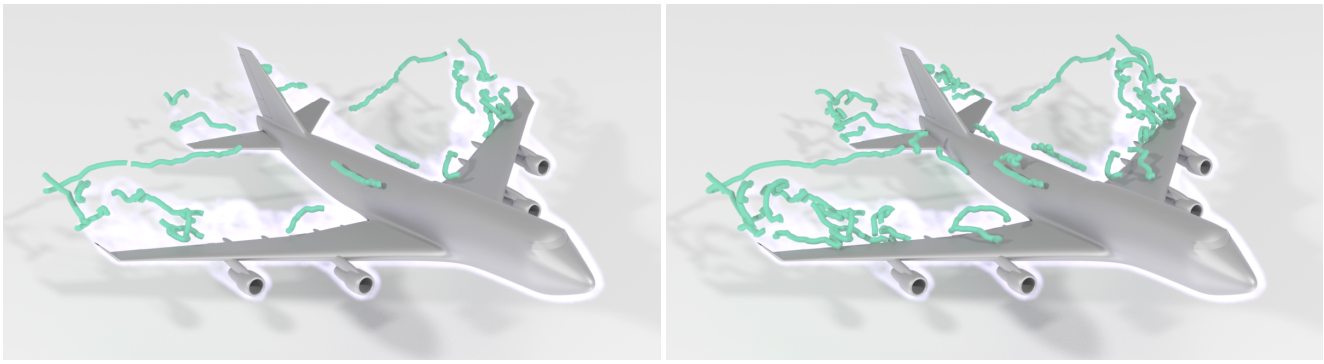
Eurographics Conference on Visualization (EuroVis) 2026
N. Andrienko, M. Chen, and B. Wang
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 45 (2026), Number 3

Parallel Vectors Extraction using Bézier Clipping – Additional Material –

Nico Daßler and Tobias Günther

Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany

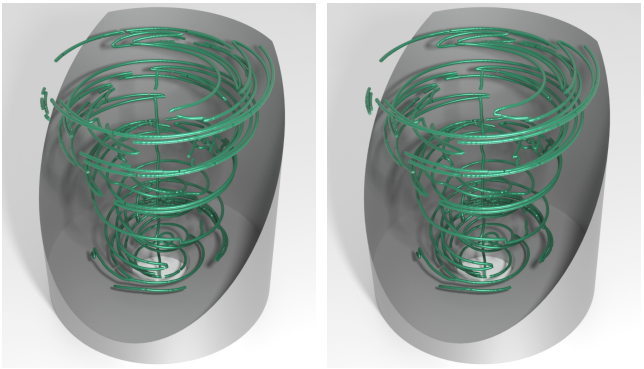


(a) *Newton-Raphson method.* **(b)** *Our Bézier clipping.*

Figure 1: *Results on the* BOEING *dataset using Newton-Raphson only [PR99] (left) and our Bézier Clipping approach (right). Both methods use the same bicubic interpolant on cell faces. The Newton-Raphson method misses 15% of the solutions due to insufficient initial guesses.*

1. Additional Performance Measurements

Table 1 and Table 2 list additional performance measurements for the independent bicubic interpolant and for the bicubic interpolant with derived acceleration. The behavior is overall the same as with the bilinear case. The component-based Bézier clipping consistently outperforms the bisection method, and the projection-based Bézier clipping consistently outperforms the component-based clipping. Due to the higher polynomial degree of the root finding problem, the bicubic interpolant with derived acceleration is the slowest, followed by bicubic interpolation. The bilinear interpolant remains to be the fastest. Across all datasets, it is by a factor of about 8.2× faster than the bicubic interpolant. We also observe a slightly larger speedup when using one of our clipping solvers (8.31×) compared to the bisection solver (8.01×). The bicubic interpolation is about 3.47× faster than the same with derived acceleration. For the solve runtime, we see an average speedup of about 7.28× of the bilinear interpolant over the bicubic interpolant and a 3.05× speedup of the bicubic interpolant over the bicubic interpolant with derived acceleration. The bisection solver again scales slightly worse in this scenario with a speedup of only about 2.63×, compared to the projection-based clipping solver with a speedup factor of about 3.76×.



(a) *Newton-Raphson method.* **(b)** *Our Bézier clipping.*

Figure 2: *Results on the* SWIRLING JET *dataset using Newton-Raphson and our Bèzier Clipping approach with bicubic interpolant and derived acceleration. The Newton-Raphson method misses* 0.49% *of solutions, resulting in minor visual differences.*

2. Additional Comparisons

Table 1 shows that the Newton-Raphson solver misses about 15% of the solutions compared to the Bézier clipping solver on the BOE-

Dataset	Algorithm	Runtime (Serial)	Runtime (Parallel)	Solve Runtime	# Duplicates	# Solutions	⌀ Sol. Depth	⌀ Clip Size
DELTA WING	Bisection	736.70 ± 6.61	50.68 ± 0.27	104.12 ± 0.19	924	2398	45.78	0.500
	Clipping (C)	672.01 ± 16.91	47.49 ± 1.24	49.66 ± 0.31	95	2398	17.04	0.723
	Clipping (P)	672.87 ± 0.19	43.74 ± 0.95	18.15 ± 0.03	0	2398	7.92	0.866
	Newton-Raphson	654.80 ± 2.46	41.83 ± 0.49	54.83 ± 0.18	0	2299	—	—
BORROMEAN	Bisection	1766.03 ± 5.00	115.25 ± 0.28	351.34 ± 0.56	3862	7002	42.95	0.500
	Clipping (C)	1574.61 ± 3.93	99.90 ± 0.18	156.21 ± 0.38	555	7002	18.39	0.692
	Clipping (P)	1455.86 ± 5.06	92.55 ± 0.21	40.45 ± 0.08	0	7002	6.39	0.953
	Newton-Raphson	1551.81 ± 3.48	95.67 ± 0.38	124.05 ± 0.24	0	6973	—	—
SWIRLING JET	Bisection	8016.07 ± 18.11	523.40 ± 0.87	1446.89 ± 5.81	13345	11765	68.90	0.500
	Clipping (C)	7028.38 ± 26.62	452.12 ± 0.83	499.11 ± 2.11	2585	11765	31.21	0.639
	Clipping (P)	6618.40 ± 7.68	423.61 ± 0.74	78.78 ± 0.14	88	11765	7.27	0.957
	Newton-Raphson	6897.20 ± 21.22	434.04 ± 1.44	326.08 ± 0.22	0	11708	—	—
BOEING	Bisection	3860.64 ± 24.13	294.56 ± 0.98	1977.25 ± 5.22	23740	29511	49.25	0.500
	Clipping (C)	2780.34 ± 71.16	188.00 ± 0.82	973.74 ± 4.37	2973	29511	25.41	0.635
	Clipping (P)	2246.04 ± 23.74	150.55 ± 0.71	330.83 ± 0.49	0	29511	10.6	0.752
	Newton-Raphson	2874.58 ± 4.56	176.61 ± 0.35	1134.34 ± 1.47	0	24932	—	—
NINE CL	Bisection	2867.64 ± 21.73	190.55 ± 0.35	35.08 ± 0.19	0	1625	46.85	0.500
	Clipping (C)	2840.56 ± 1.85	187.36 ± 0.24	8.20 ± 0.08	0	1625	5.23	0.996
	Clipping (P)	2822.20 ± 11.60	185.98 ± 0.46	6.73 ± 0.01	0	1625	5.23	0.995
	Newton-Raphson	2898.47 ± 8.16	188.16 ± 0.59	3.18 ± 0.02	0	1625	—	—

Table 1: *Metrics recorded for bicubic interpolation on different datasets using the bisection method, our component-wise clipping (C), our projection-based clipping (P), and the Newton-Raphson method. All runtimes are measured and averaged for 10 runs, listing mean and standard deviation.*

Dataset	Algorithm	Runtime (Serial)	Runtime (Parallel)	Solve Runtime	# Duplicates	# Solutions	⌀ Sol. Depth	⌀ Clip Size
DELTA WING	Bisection	1020.94 ± 2.19	68.44 ± 0.17	324.44 ± 1.69	2194	3693	40.05	0.500
	Clipping (C)	883.40 ± 3.57	56.34 ± 0.22	183.33 ± 0.25	188	3693	19.36	0.652
	Clipping (P)	778.63 ± 0.26	49.59 ± 0.18	64.65 ± 0.14	0	3693	9.14	0.785
	Newton-Raphson	805.31 ± 0.23	50.46 ± 0.13	145.37 ± 0.07	0	2928	—	—
BORROMEAN	Bisection	6862.38 ± 7.41	463.37 ± 1.07	596.15 ± 0.96	5685	7281	36.15	0.500
	Clipping (C)	6593.85 ± 3.08	442.83 ± 1.10	310.25 ± 1.09	599	7281	16.52	0.686
	Clipping (P)	6357.05 ± 4.69	431.21 ± 1.16	86.73 ± 0.12	0	7281	6.44	0.926
	Newton-Raphson	6484.48 ± 3.39	438.23 ± 1.34	248.00 ± 0.35	0	7090	—	—
SWIRLING JET	Bisection	31801.73 ± 34.19	2166.73 ± 5.21	2313.39 ± 14.15	12674	12640	55.32	0.500
	Clipping (C)	30396.87 ± 10.00	2068.10 ± 5.66	926.76 ± 2.47	1942	12640	26.71	0.635
	Clipping (P)	29689.55 ± 21.24	2034.15 ± 5.85	195.51 ± 0.27	438	12640	7.53	0.883
	Newton-Raphson	29953.97 ± 6.88	2052.56 ± 6.12	656.98 ± 0.88	0	12134	—	—
BOEING	Bisection	12089.95 ± 53.28	831.96 ± 1.91	10158.63 ± 55.16	48344	55133	65.12	0.500
	Clipping (C)	6367.85 ± 16.18	391.01 ± 2.15	4490.11 ± 8.13	6818	55133	32.22	0.640
	Clipping (P)	4839.47 ± 7.64	682.23 ± 6.17	2911.22 ± 9.76	8579	55133	13.19	0.649
	Newton-Raphson	4252.16 ± 22.44	253.75 ± 0.88	2494.52 ± 11.98	0	36025	—	—
NINE CL	Bisection	14747.24 ± 33.03	1016.16 ± 3.65	55.95 ± 0.39	0	1625	34.62	0.500
	Clipping (C)	14675.09 ± 2.36	1010.99 ± 3.32	14.79 ± 0.09	0	1625	4.31	0.994
	Clipping (P)	14664.73 ± 11.16	1014.50 ± 3.35	12.23 ± 0.02	0	1625	4.62	0.993
	Newton-Raphson	14639.45 ± 7.57	1014.59 ± 3.19	3.83 ± 0.03	0	1625	—	—

Table 2: *Metrics recorded for bicubic interpolation with derived acceleration to compare the performance on different datasets using the bisection method, our component-wise clipping (C), our projection-based clipping (P), and the Newton-Raphson method. All runtimes are measured and averaged for 10 runs, listing mean and standard deviation.*

ING dataset. Figure 1 shows the extracted feature curves for this experiment using the Newton-Raphson method on the left and our projection-based Bézier clipping solver on the right. Both methods used the same bicubic interpolant on cell faces. The missing solutions are directly reflected in the rendered results. This suggests that a Newton-Raphson solver is not sufficient for this dataset and interpolant combination. In contrast, the Newton-Raphson solver was better suited for the SWIRLING JET dataset, when using bicubic interpolation and derived acceleration. Table 2 shows that only about 0.49% of the solutions are missing, resulting in minor visual differences in the extracted feature curves, which can be seen in Figure 2. However, for both datasets the clipping solver is still preferred due to its faster solve runtime, achieving a speedup factor of about 3.4× on the BOEING dataset and a speedup factor of about 3.3× on the SWIRLING JET.

3. Cross Product Components of Two Bilinear Fields

In the special case of $\mathbf{v}(\mathbf{x})$ and $\mathbf{w}(\mathbf{x})$ being both bilinearly interpolated across a cell face, the biquadratic cross product can be computed symbolically in Bézier form. We list the control points of the

z-component of the cross product surface $\mathbf{c}(u,v)$ explicitly:

$$\mathbf{c}_{00}^{[z]} = \mathbf{w}_{00}^{[x]} \cdot \mathbf{v}_{00}^{[y]} - \mathbf{w}_{00}^{[y]} \cdot \mathbf{v}_{00}^{[x]} \tag{1}$$

$$\mathbf{c}_{01}^{[z]} = \frac{1}{2}\left(\mathbf{w}_{00}^{[x]} \cdot \mathbf{v}_{01}^{[y]} + \mathbf{w}_{01}^{[x]} \cdot \mathbf{v}_{00}^{[y]} - \mathbf{w}_{00}^{[y]} \cdot \mathbf{v}_{01}^{[x]} - \mathbf{w}_{01}^{[y]} \cdot \mathbf{v}_{00}^{[x]}\right) \tag{2}$$

$$\mathbf{c}_{02}^{[z]} = \mathbf{w}_{01}^{[x]} \cdot \mathbf{v}_{01}^{[y]} - \mathbf{w}_{01}^{[y]} \cdot \mathbf{v}_{01}^{[x]} \tag{3}$$

$$\mathbf{c}_{10}^{[z]} = \frac{1}{2}\left(\mathbf{w}_{00}^{[x]} \cdot \mathbf{v}_{10}^{[y]} + \mathbf{w}_{10}^{[x]} \cdot \mathbf{v}_{00}^{[y]} - \mathbf{w}_{00}^{[y]} \cdot \mathbf{v}_{10}^{[x]} - \mathbf{w}_{10}^{[y]} \cdot \mathbf{v}_{00}^{[x]}\right) \tag{4}$$

$$\mathbf{c}_{11}^{[z]} = \frac{1}{4}\left(\mathbf{w}_{00}^{[x]} \cdot \mathbf{v}_{11}^{[y]} + \mathbf{w}_{01}^{[x]} \cdot \mathbf{v}_{10}^{[y]} + \mathbf{w}_{10}^{[x]} \cdot \mathbf{v}_{01}^{[y]} + \mathbf{w}_{11}^{[x]} \cdot \mathbf{v}_{00}^{[y]} \right. \tag{5}$$

$$\left. - \mathbf{w}_{00}^{[y]} \cdot \mathbf{v}_{11}^{[x]} - \mathbf{w}_{01}^{[y]} \cdot \mathbf{v}_{10}^{[x]} - \mathbf{w}_{10}^{[y]} \cdot \mathbf{v}_{01}^{[x]} - \mathbf{w}_{11}^{[y]} \cdot \mathbf{v}_{00}^{[x]}\right) \tag{6}$$

$$\mathbf{c}_{12}^{[z]} = \frac{1}{2}\left(\mathbf{w}_{01}^{[x]} \cdot \mathbf{v}_{11}^{[y]} + \mathbf{w}_{11}^{[x]} \cdot \mathbf{v}_{01}^{[y]} - \mathbf{w}_{01}^{[y]} \cdot \mathbf{v}_{11}^{[x]} - \mathbf{w}_{11}^{[y]} \cdot \mathbf{v}_{01}^{[x]}\right) \tag{7}$$

$$\mathbf{c}_{20}^{[z]} = \mathbf{w}_{10}^{[x]} \cdot \mathbf{v}_{10}^{[y]} - \mathbf{w}_{10}^{[y]} \cdot \mathbf{v}_{10}^{[x]} \tag{8}$$

$$\mathbf{c}_{21}^{[z]} = \frac{1}{2}\left(\mathbf{w}_{10}^{[x]} \cdot \mathbf{v}_{11}^{[y]} + \mathbf{w}_{11}^{[x]} \cdot \mathbf{v}_{10}^{[y]} - \mathbf{w}_{10}^{[y]} \cdot \mathbf{v}_{11}^{[x]} - \mathbf{w}_{11}^{[y]} \cdot \mathbf{v}_{10}^{[x]}\right) \tag{9}$$

$$\mathbf{c}_{22}^{[z]} = \mathbf{w}_{11}^{[x]} \cdot \mathbf{v}_{11}^{[y]} - \mathbf{w}_{11}^{[y]} \cdot \mathbf{v}_{11}^{[x]} \tag{10}$$

Analogously, the x-components of the cross product $\mathbf{c}_{ij}^{[x]}$ are found by selecting the vector components $[z]$ and $[x]$ of $\mathbf{v}$ and $\mathbf{w}$, and the y-components $\mathbf{c}_{ij}^{[y]}$ of the cross product use the components $[y]$ and $[z]$ of $\mathbf{v}$ and $\mathbf{w}$. In the main paper, we derived the cross product generally for input fields of arbitrary Bézier degree, which includes the biquadratic cross product as a special case.

4. Interpolant Comparison

Our Bézier clipping based parallel vectors solver is applicable to higher-order polynomial interpolants. To examine the differences between a piecewise linear [PR99], a bilinear [BRG21], a bicubic interpolant (Section 3.2 in main paper), and the derived acceleration (Section 3.3 in main paper), we introduce an analytic vector field with known ground truth corelines. The NINECL data set, is a 3D steady vector field $\mathbf{v}(x,y,z)$ defined in the domain $[-2,2]^3$:

$$\mathbf{v}(x,y,z) = \begin{pmatrix} -y(1-y)(1+y) \\ x(1-x)(1+x) \\ 1 \end{pmatrix}, \tag{11}$$

The flow contains vertical corelines for $\mathbf{v} \parallel \mathbf{a}$, including five vortices at $(0,0)$ and $(\pm 1,\pm 1)$, as well as four bifurcation lines at $(0,\pm 1)$ and $(\pm 1,0)$. Further, there are four degenerated lines at $(\pm\frac{1}{\sqrt{3}},\pm\frac{1}{\sqrt{3}})$ at which one eigenvalue vanishes to zero. We compare the accuracy of the different interpolants by measuring the residual error quantitatively and by comparing the resulting corelines visually. For this, we densely sample the analytic field onto a 128^3 grid and then progressively downsample to a resolution of 16^3. With this, we can analyze how many sample points the different interpolants require in order to accurately capture the features of the underlying analytic field. Figure 3 compares the residual error for all interpolants. It shows that the derived acceleration interpolant outperforms the others significantly, suggesting that this interpolation method is better at representing the data of the underlying vector field. With the derived acceleration, a resolution of 64^3 is sufficient to achieve a very high degree of accuracy. However, this accuracy comes with a large drop in performance, which is visible in the table below the plot. Derived acceleration is approximately 4.9× slower than the bicubic interpolant and 38.0× slower than the bilinear interpolant. It is also approximately 2.2× slower than the piecewise linear interpolator. Fig. 4 shows the extracted corelines for resolution 16^3 alongside the ground truth.

Resolution	Piecewise Linear	Bilinear	Bicubic	Derived Accel.
16^3	7.92 ± 0.01	0.60 ± 0.00	4.63 ± 0.01	19.43 ± 0.01
32^3	90.76 ± 0.22	4.91 ± 0.01	37.97 ± 0.06	188.01 ± 0.12
64^3	849.59 ± 1.30	42.29 ± 0.05	333.79 ± 0.21	1712.61 ± 0.72
128^3	6324.08 ± 4.63	358.30 ± 0.23	2898.47 ± 8.16	14639.45 ± 7.57

Figure 3: *Top: Comparison of cross product residuals (lower is better) for: piecewise linear (●), bilinear clipping (●), bicubic clipping (●), and bicubic clipping with derived acceleration (●). Bottom: Comparison of the total runtime (serial) in milliseconds.*

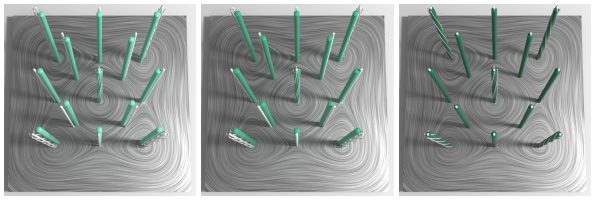


Figure 4: *Comparison of interpolants on the* NINE CL *dataset downsampled to a* 16^3 *grid. From left to right: bilinear (●), bicubic (●) and derived acceleration (●) are shown. Ground truth lines are shown in white, and for vortex cores swirling streamlines are visualized. The interpolant with derived acceleration reproduces the expected lines better than the other two approaches.*

5. Pseudocodes

The Bézier-based formulation includes tensor contractions, which we in part accelerate by precomputing weight tables that store the binomial coefficient fractions. For reproducibility of those algorithms, we list here the pseudocodes of three important steps. First, Alg. 1 lists the computation of control points for the cross product of two given vector fields in Bézier form, where the degree of both vector fields can be arbitrary.

Second, Alg. 2 describes how a $4\times4\times4$ block of values $\mathbf{p}_{i,j,k}$ is tricubicly interpolated, such that the inner $2\times2\times2$ block is interpolated with C^1 continuity. For this, the 4×4 matrix $\mathbf{M}$ is applied successively in each dimension, which converts from the values to the Bézier form of the cubic Hermite interpolant.

Lastly, Alg. 3 lists how the Bézier control points of the acceleration field are computed from the Bézier form of both the velocity field $\mathbf{v}_{i,j,k}$ and the Jacobian field $\mathbf{J}_{i,j,k}$. Again, the binomial coefficient weights are precomputed to avoid redundant computations.

Algorithm 1 Computation of Cross Product Control Points $\mathbf{c}_{i,j}$

Precompute weights:
for $i' = 0$ to n_v **do**
  for $j' = 0$ to m_v **do**
    for $i'' = 0$ to n_w **do**
      for $j'' = 0$ to m_w **do**
        $\omega_{i',j',i'',j''} \leftarrow \frac{\binom{n_v}{i'}\binom{n_w}{i''}}{\binom{n_c}{i}} \cdot \frac{\binom{m_v}{j'}\binom{m_w}{j''}}{\binom{m_c}{j}}$, $i \leftarrow i'+i''$, $j \leftarrow j'+j''$
Compute control points:
for $i = 0$ to n_c **do**
  for $j = 0$ to m_c **do**
    $\mathbf{c}_{i,j} \leftarrow \mathbf{0}$
    for all i', i'' with $i'+i''=i$ **do**
      for all j', j'' with $j'+j''=j$ **do**
        $\mathbf{c}_{i,j} \leftarrow \mathbf{c}_{i,j} + \omega_{i',j',i'',j''} \cdot \left(\mathbf{v}_{i',j'} \times \mathbf{w}_{i'',j''}\right)$
return $\{\mathbf{c}_{i,j}\}_{0\le i\le n_c,\, 0\le j\le m_c}$

Algorithm 2 Computation of Bézier Control Points $\mathbf{b}_{i,j,k}$

Contraction along x–axis:
for $i = 0$ to 3 **do**
  for $j = 0$ to 3 **do**
    for $k = 0$ to 3 **do**
      $\mathbf{t}^{x}_{i,j,k} \leftarrow \sum_{a=0}^{3} \mathbf{M}_{i,a} \cdot \mathbf{p}_{a,j,k}$
Contraction along y–axis:
for $i = 0$ to 3 **do**
  for $j = 0$ to 3 **do**
    for $k = 0$ to 3 **do**
      $\mathbf{t}^{xy}_{i,j,k} \leftarrow \sum_{b=0}^{3} \mathbf{M}_{j,b} \cdot \mathbf{t}^{x}_{i,b,k}$
Contraction along z–axis:
for $i = 0$ to 3 **do**
  for $j = 0$ to 3 **do**
    for $k = 0$ to 3 **do**
      $\mathbf{b}_{i,j,k} \leftarrow \sum_{c=0}^{3} \mathbf{M}_{k,c} \cdot \mathbf{t}^{xy}_{i,j,c}$
return $\{\mathbf{b}_{i,j,k}\}_{0\le i,j,k\le 3}$

Algorithm 3 Computation of Acceleration Control Points $\mathbf{a}_{i,j,k}$

Precompute weights:
for $i' = 0$ to n_1 **do**
  for $i'' = 0$ to n_1 **do**
    $\omega^{u}_{i',i'',i} \leftarrow \frac{\binom{n_1}{i'}\binom{n_1}{i''}}{\binom{2n_1}{i}}, \quad i = i'+i''$
for $j' = 0$ to n_2 **do**
  for $j'' = 0$ to n_2 **do**
    $\omega^{v}_{j',j'',j} \leftarrow \frac{\binom{n_2}{j'}\binom{n_2}{j''}}{\binom{2n_2}{j}}, \quad j = j'+j''$
for $k' = 0$ to n_3 **do**
  for $k'' = 0$ to n_3 **do**
    $\omega^{w}_{k',k'',k} \leftarrow \frac{\binom{n_3}{k'}\binom{n_3}{k''}}{\binom{2n_3}{k}}, \quad k = k'+k''$
Compute acceleration control points:
for $i = 0$ to $2n_1$ **do**
  for $j = 0$ to $2n_2$ **do**
    for $k = 0$ to $2n_3$ **do**
      $\mathbf{a}_{i,j,k} \leftarrow \mathbf{0}$
      for all i', i'' with $i'+i''=i$ **do**
        for all j', j'' with $j'+j''=j$ **do**
          for all k', k'' with $k'+k''=k$ **do**
            $w \leftarrow \omega^{u}_{i',i'',i} \cdot \omega^{v}_{j',j'',j} \cdot \omega^{w}_{k',k'',k}$
            $\mathbf{a}_{i,j,k} \leftarrow \mathbf{a}_{i,j,k} + w \cdot \left(\mathbf{J}'_{i',j',k'} \cdot \mathbf{v}_{i'',j'',k''}\right)$
return $\{\mathbf{a}_{i,j,k}\}_{0\le i\le 2n_1,\, 0\le j\le 2n_2,\, 0\le k\le 2n_3}$

A C++ reference implementation is available at [DG26].

References

[BRG21] BAEZA ROJO I., GÜNTHER T.: Coreline criteria for inertial particle motion. In *Topological Methods in Data Analysis and Visualization IV*. Springer, 2021, pp. 133–157. doi:10.1007/978-3-030-83500-2_8. 3

[DG26] DASSLER N., GÜNTHER T.: Reference implementation for parallel vectors extraction using Bézier clipping. https://github.com/fau-vc/pvBezier, 2026. Accessed: 2026-03-04. 4

[PR99] PEIKERT R., ROTH M.: The "Parallel Vectors" operator – a vector field visualization primitive. In *Proc. IEEE Visualization* (San Francisco, CA, USA, 1999), IEEE, pp. 263–270. doi:10.1109/VISUAL.1999.809896. 1, 3