1 \textbf{DERIVATION SECTIONAL NEWTON DESCENT}

The third component of the cross product \( f \) vanishes to zero, if the first two components are already zero. To see this, assume that the first two components of \( f(x) \) are 0 and \( w_3 \neq 0 \):

\[
\begin{align*}
f(x) = v(x) \times w(x) = & \begin{bmatrix} v_2w_3 - v_3w_2 \\ v_3w_1 - v_1w_3 \\ v_1w_2 - v_2w_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}
\end{align*}
\]

The only exception is the special case of \( v(x) \neq 0 \), \( w(x) \neq 0 \) and \( v_3 = w_3 = 0 \), which is caught by the solver not converging [2]. Solving for \( v_1 = \frac{v_2}{w_3} \) and \( v_2 = \frac{v_1}{w_2} \) and inserting into Eq. (1) yields:

\[
\begin{align*}
\begin{bmatrix}
0 \\
0 \\
\frac{v_1w_2 - v_2w_1}{w_3} - \frac{v_3w_2}{w_3}
\end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}
\end{align*}
\]

We can therefore consider only two components of \( f \) and follow Schindler et al. [2] in choosing the components \( f_k, f_l \) that maximize

\[
k, l = \arg \max_{k, l \in \{0, 1, 2\}} \| \nabla f_k(x_i) \times \nabla f_l(x_i) \|_2, \text{ s.t. } k \neq l
\]

We can find the roots of \( f \) by considering only two components of

\[
\nabla f(x_i) \cdot d = -f(x_i)
\]

which gives rise to the linear system

\[
(\nabla f_k(x_i), \nabla f_l(x_i))^T \cdot d = -\begin{bmatrix} f_k(x_i) \\ f_l(x_i) \end{bmatrix}
\]

where \( d \) is a linear combination of the two selected gradients:

\[
d = \alpha \nabla f_k(x_i) + \beta \nabla f_l(x_i) = (\nabla f_k(x_i), \nabla f_l(x_i)) \begin{bmatrix} \alpha \\ \beta \end{bmatrix}
\]

Inserting Eq. (6) into Eq. (5) and rearranging for the weights \( \alpha, \beta \):

\[
\begin{bmatrix} \alpha \\ \beta \end{bmatrix} = -A^{-1} \begin{bmatrix} f_k(x_i) \\ f_l(x_i) \end{bmatrix}
\]

with \( A = (\nabla f_k(x_i), \nabla f_l(x_i))^T (\nabla f_k(x_i), \nabla f_l(x_i)) \). Inserting the weights \( \alpha, \beta \) from Eq. (7) into Eq. (6) leads to the expression

\[
d = -(\nabla f_k(x_i), \nabla f_l(x_i)) \cdot A^{-1} \begin{bmatrix} f_k(x_i) \\ f_l(x_i) \end{bmatrix}
\]

Implementation Details. During the sectional Newton descent, we use trilinear hardware interpolation to sample the vector fields \( v \) and \( w \), when evaluating the current residual. The filter field \( s(x) \) can be sampled with nearest-interpolation. Our Newton descent employs a trust-region, which means that each step has an upper bound of one voxel. Further, if the residual increases, the step is revoked and the step size is decreased for the next attempt. Since the filter field \( s(x) \) takes us close to a solution, we use in practice at most 10 iterations for the sectional Newton descent. Later in Fig. 2, we vary the number of iterations, showing that 10 is an adequate choice. Note that for 0 iterations, the filter field \( s(x) \) is visualized.

2 \textbf{PSEUDO CODE}

2.1 \textbf{Bézier-based Test if PV Solution Exists}

The search for a PV solution is a multi-variate root finding problem:

\[
f(x, y) = v(x, y) \times w(x, y) = \begin{bmatrix} f_1(x, y) \\ f_2(x, y) \\ f_3(x, y) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}
\]

By converting the cross product components into Bernstein basis \( B_i^k(t) = \binom{k}{i} t^i (1 - t)^{k-i} \):

\[
f_k(x, y) = \sum_{i=0}^{2} \sum_{j=0}^{2} B_i^k(x) \cdot B_j^k(y) \cdot b_{i,j}
\]

the potential existence of a PV solution on the face of a cell can be tested using the convex hull property, as listed in Alg. 1.

2.2 \textbf{Ray Marching Loop}

Alg. 2 gives a high-level view of the inner steps of the ray marching loop. At each sample location, we first lookup the filter field to do empty space skipping. If we are in or nearby a voxel with potential PV solution, we perform the sectional Newton descent. If the descent does not converge to a solution, we return to advance on the ray with the next step. If a solution is found, a ray intersection test with the tube proxy is performed. On success, the normal is estimated and Lambertian shading is performed. Transfer functions are stored in look-up tables.
in Alg. 3, we provide the pseudo-code for the sampling of the cross-product Jacobian of two trilinearly interpolated vector fields. Note that the gradients of the cross-product components \( \nabla f_1, \nabla f_2 \) and \( \nabla f_3 \) are found in the rows of the Jacobian of the cross-product \( \nabla \mathbf{v} \):

\[
\nabla \mathbf{v} = \begin{pmatrix} f_x & f_y & f_z \end{pmatrix} = \begin{pmatrix} -\frac{\partial f_2}{\partial T} & \frac{\partial f_3}{\partial T} & 0 \\ -\frac{\partial f_2}{\partial U} & -\frac{\partial f_3}{\partial U} & 0 \\ \frac{\partial f_2}{\partial V} & \frac{\partial f_3}{\partial V} & 0 \end{pmatrix}.
\]

2.3 Sampling the Cross-Product Jacobian

Solving for the direction \( \mathbf{d} \) of the next step in Eq. (8) requires the gradients of the cross-product components, i.e., \( \nabla f_1, \nabla f_2, \) and \( \nabla f_3 \). In Alg. 3, we provide the pseudo-code for the sampling of the cross-product Jacobian of two trilinearly interpolated vector fields. Note that the gradients of the cross-product components \( \nabla f_1, \nabla f_2 \) and \( \nabla f_3 \) are found in the rows of the Jacobian of the cross-product \( \nabla \mathbf{v} \):

\[
\nabla \mathbf{v} = \begin{pmatrix} f_x & f_y & f_z \end{pmatrix} = \begin{pmatrix} -\frac{\partial f_2}{\partial T} & \frac{\partial f_3}{\partial T} & 0 \\ -\frac{\partial f_2}{\partial U} & -\frac{\partial f_3}{\partial U} & 0 \\ \frac{\partial f_2}{\partial V} & \frac{\partial f_3}{\partial V} & 0 \end{pmatrix}.
\]

### References


Figure 1: Comparison of varying ray marching step sizes, shown here for a close-up of the Cylinder flow.

Figure 2: Results for varying number of sectional Newton descent iterations in the Swirling Jet using the $v \parallel b$ criterion.

Figure 3: Comparison of spheres and cylinders used to span the line geometry during the implicit ray casting with varying radii: 0.1 (left), 0.3 (middle) and 1.0 (right). Shown for the Swirling Jet data set. Differences can be seen for small radii in the left-most column.
Figure 4: Performance scaling plot of all datasets using our NVIDIA IndeX implementation (lower is better).

Figure 5: Explicit extraction of point geometry by descending a uniformly distributed set of particles onto the nearest line structure. Here, a CPU implementation of the sectional Newton descent was used.

Figure 6: Here, the camera perspectives are shown that were used for the Nvidia IndeX performance benchmark, showcasing the rendering coverage of the two most challenging data sets under consideration: The Borromean Rings (a), (b) and the Swirling Jet (c), (d). The rendering and volume resolution are the same as given in Table 1 in the main paper.