Overview

If we consider a vector field in space and we have a rectilinear volume whose vertices are sampled from this vector field, one method of visualizing the field is to place particles in the field and animate their motion through the field. This time animation is the basis for a number of visualization methods – streamlines, stream surfaces, and stream ribbons. In this paper we review the numerical techniques necessary for tracing the path of a particle through the field. We will study these techniques in two-dimensional vector fields, but they are directly extensible to three-dimensional fields.

Interpolating the Vectors

Consider a point p inside one cell of our rectilinear volume. This cell has four corner points p₀₀, p₁₀, p₀₁, and p₁₁, each of which has an associated vector. n₀₀, n₁₀, n₀₁, and n₁₁, respectively. We will assume that our cells have a unit width and height.¹

¹This is really no restriction, as we can scale our rectilinear volume such that the cells have this property.
We can calculate the vector at \( p \) by using the values \( u \) and \( v \) and bilinearly interpolating the vectors at the corners. Here

\[
\vec{v} = (1 - v) \vec{v}_0 + v \vec{v}_1
\]

where

\[
\vec{v}_0 = (1 - u) \vec{v}_{0,0} + u \vec{v}_{1,0}
\]
\[
\vec{v}_1 = (1 - u) \vec{v}_{0,1} + u \vec{v}_{1,1}
\]

or

\[
\vec{v} = (1 - v)(1 - u) \vec{v}_{0,0} + (1 - v) u \vec{v}_{1,0} + v(1 - u) \vec{v}_{0,1} + v u \vec{v}_{1,1}
\]

**Euler’s Method**

Given a point \( p \) in a cell of our rectilinear grid, an obvious way to trace a particle through a cell is to step along the vector \( \vec{v} \) associated with \( p \) using a very short step size \( \Delta t \). The new point \( p_1 \) can be written as

\[
p_1 = p + \Delta t \vec{v}
\]

We now calculate a point \( p_2 \) by stepping along the vector associated with \( p_1 \) (calculated by the bilinear interpolation algorithm of the section above), and continue in this fashion. This is illustrated in the following figure.
This figure suggests the following strategy: (1) Select a "step size" $\Delta t$; (2) Starting with the point $p_i$, where $p_0 = p$, we perform the following iteration:

- Calculate $\vec{v}_i$ by bilinearly interpolating the vectors at the corners of the cell (Note: You first have to determine the cell in which $p_i$ lies.), and

- Calculate $p_{i+1} = p_i + \Delta t \vec{v}_i$

This algorithm is called Euler’s Algorithm.\(^2\) It has an error estimate which is bounded by $O(\Delta t^2)$, which is not very good for the estimates that we require in visualization. The algorithm is mostly used as an initial approximation method for the better algorithms discussed below.

**The Improved Euler’s Method**

The improved Euler method utilizes Euler’s method to predict the location of the new point, but then utilizes more information from this point to correct the initial guess. This method is illustrated in the following figure.

\(^2\)Leonhard Euler was a Swiss mathematician of the 18th century.

\(^3\)This estimate comes from Taylor’s theorem which can be found in any elementary numerical analysis book.
In this figure, we begin at the point $p_i$. We use Euler’s method to predict the location of the next point in the iteration $p^P_{i+1}$. We then use the calculated vector $\vec{v}^P_{i+1}$ at $p^P_{i+1}$ as additional information that can be used to better predict $p_{i+1}$. To obtain $p_{i+1}$, we begin at $p_i$ and move half the step size ($\frac{1}{2} \Delta t$) along $\vec{v}^i$, then half the step size along the vector $\vec{v}^P_{i+1}$. Mathematically, this can be written as

$$p_{i+1} = p_i + \frac{1}{2} \Delta t \vec{v}^i + \frac{1}{2} \Delta t \vec{v}^P_{i+1}$$

This numerical method is called a predictor-corrector method (for obvious reasons). It has an error estimate which is bounded by $O(\Delta t^2)$, which is substantially better than Euler’s algorithm. The following illustration shows the difference between Euler’s method and the improved Euler method on our example cell.

The white dots are the approximation due to Euler’s method. The black dots are the approximation due to the improved Euler method. This “improved” version is substantially used in the visualization community, but a better version, generated by two German mathematicians, Runge and Kutta, is most frequently used.

**The Runge-Kutta Algorithm**

The fourth-order Runge-Kutta method is similar to the improved Euler method, but utilizes several predictor and corrector steps. The predictor values are illustrated in this following figure (We note that the picture was drawn with $\Delta t = 1$).
In this case, three vector predictors are calculated:

- \( \vec{v}_{i+1}^1 \) – the vector corresponding to the point \( p_i + \frac{1}{2} \Delta t \vec{v}_i \);
- \( \vec{v}_{i+1}^2 \) – the vector corresponding to the point \( p_i + \frac{1}{2} \Delta t \vec{v}_{i+1}^1 \); and
- \( \vec{v}_{i+1}^3 \) – the vector corresponding to the point \( p_i + \Delta t \vec{v}_{i+1}^2 \).

We note that each predictor is used to obtain the subsequent predictor values. These vectors are blended into the final result as follows:

\[
p_{i+1} = p_i + \frac{1}{6} \Delta t \vec{v}_i + \frac{1}{3} \Delta t \vec{v}_{i+1}^1 + \frac{1}{3} \Delta t \vec{v}_{i+1}^2 + \frac{1}{6} \Delta t \vec{v}_{i+1}^3
\]

which is shown in the following figure.
The formula can be simplified to

\[ \mathbf{p}_{i+1} = \mathbf{p}_i + \frac{1}{6} \Delta t \left( \mathbf{v}_i + 2 \mathbf{v}_{i+1} + 2 \mathbf{v}_{i+1}^2 + \mathbf{v}_{i+1}^3 \right) \]

which is the way it is usually written.

This method seems complex, but can be easily implemented. The steps are the following:

- calculate the vector \( k_1 = \Delta t \mathbf{v}_i \);
- determine the vector \( \mathbf{v}_{i+1}^1 \), which is the vector corresponding to the point \( \mathbf{p}_i + \frac{1}{2} k_1 \);
- calculate the vector \( k_2 = \Delta t \mathbf{v}_{i+1}^1 \);
- determine the vector \( \mathbf{v}_{i+1}^2 \), which is the vector corresponding to the point \( \mathbf{p}_i + \frac{1}{2} k_2 \);
- calculate the vector \( k_3 = \Delta t \mathbf{v}_{i+1}^2 \);
- determine the vector \( \mathbf{v}_{i+1}^3 \), which is the vector corresponding to the point \( \mathbf{p}_i + k_3 \);
- calculate the vector \( k_4 = \Delta t \mathbf{v}_{i+1}^3 \); and then
- calculate \( \mathbf{p}_{i+1} = \mathbf{p}_i + \frac{1}{6} (k_1 + 2k_2 + 3k_3 + k_4) \)

The differences between Euler’s method, the improved Euler method, and the Runge-Kutta method for our sample cell are shown in the following illustration.

The white dots are the approximations due to Euler’s method, the gray dots are the approximations due to the improved Euler method, and the black dots are the approximations due to the fourth-order Runge-Kutta method.
The Runge-Kutta method presented here has an error bounded by $O(\Delta t^4)$, which is significantly better than the other two algorithms.

Summary

We have presented three numerical algorithms that can be used for particle tracing in vector fields. These algorithms, the Euler method, the improved Euler method and the Runge-Kutta method, have errors bounded by $O(\Delta t^2)$, $O(\Delta t^3)$, and $O(\Delta t^4)$, respectively. They are each easy to implement, and the user can pick the algorithm necessary to achieve the errors desired by the application.

These algorithms lift easily to three-dimensional scalar fields. The only difference is that we must utilize trilinear interpolation to calculate the vectors from the eight corner points of a cell.