Supplemental Material – Memory and Computation Time

Advected Tangent Curves: A General Scheme for Characteristic Curves of Flow Fields

T. Weinkauf [1], H.-C. Hege [2], and H. Theisel [3]

[1] MPI Informatik, Saarbrücken, Germany — weinkauf@mpi-inf.mpg.de
[2] Zuse Institute Berlin, Berlin, Germany — hege@zib.de
[3] University of Magdeburg, Magdeburg, Germany — theisel@isg.cs.uni-magdeburg.de

In the following, we provide more details regarding the memory requirements and computation times of our approach. Blue references such as 7d refer to figures or sections in the paper.

Memory Requirements for Time Lines and Advected Stream Lines

Assuming the time-dependent vector field $\mathbf{v}(\mathbf{x}, t)$ has n_T time steps, each of them requiring the amount of memory M, then the vector fields for time and advected stream lines require:

- M, the equivalent of a single time step, if we explore the curves for a given (t, τ). Examples of this are shown using LIC images throughout the paper, for example Figures 7d and 7e.

  Furthermore, this case applies to advected stream surfaces for a given (t, τ) such as the one shown in Figure 9. It shall be noted that the pre-computed $\bar{\bar{\mathbf{q}}}$ can be used to compute *all* advected stream surfaces for the given (t, τ). In other words, the user can interactively change the seeding line (the gray line in Figure 9) and a new surface is computed almost instantly, since these surfaces are stream surfaces of $\bar{\bar{\mathbf{q}}}$. Such an instant response is not possible with a classic approach to computing such surfaces.

  To compute time surfaces for a given (t, τ), on the other hand, we require two time line vector fields. This is explained in Section 4 and examples of time surfaces are shown in Figure 8. Nevertheless, an interactive exploration is just as possible as for advected stream surfaces, since the only added overhead is the computation of a time line before the actual computation of the time surface. Again, such an instant response is not possible with the classic, geometric approach for computing time surfaces.

- $n_T \cdot M$, the equivalent of the original flow, if we stay within a given time step t_0 or a given τ. This represents an important case in practical applications, since it is often the case that users have a good understanding of how long the integration interval τ should be. The memory requirement of $\bar{\bar{\mathbf{q}}}$ for a given τ is equal to the memory requirement of the original flow. In such a scenario, we can interactively browse through all time lines or advected stream lines in all time steps.

- $n_T^2 \cdot M$ to cover the whole space. It is rarely the case that the whole space has to be covered, since there is usually a good understanding about reasonable values for τ.

  It shall also be noted that other flow map methods such as FTLE require the same amount of memory if the whole space is to be covered.

For all the data sets used in this paper, we found it sufficient to use the same resolution for $\bar{\bar{\mathbf{q}}}$ that the original data set is given in, i.e., it yields reliable results even for long integration intervals. We sample $\bar{\bar{\mathbf{q}}}$ on a uniform grid in our current implementation. However, as discussed in Section 6.1, the advected tangent curve vector field is not defined for high τ-values in some parts of the domain, since the path lines left them earlier during the computation of $\bar{\bar{\mathbf{q}}}$. Hence, it might be beneficial to use adaptive meshes such as an octree to represent $\bar{\bar{\mathbf{q}}}$. This would reduce the memory costs without sacrificing the accuracy. Table 1 shows the memory required to store $\bar{\bar{\mathbf{q}}}$ for the 2D and 3D cylinder flows.

Table 1: Time and memory needed to compute and store the vector fields for time lines or advected stream lines.

Data set (spatial resolution)	Temporal Resolution $\tau \times t$	Computation Time in minutes	Memory in MB
Cylinder 2D	1×1 , $\tau = -2$	0.21	0.258
(338×100)	250×1	0.87	64.5
Square Cylinder 3D	1×1 , $\tau = -5$	4	6.8
$(192 \times 64 \times 48)$	102×1	160	688.5

Memory Requirements for Streak Lines

As discussed in Section 3.1, streak lines can also be described using our scheme. In fact, the theory regarding streak lines from [WT10] is a special case of the more general scheme introduced in this paper.

The memory requirements for the streak line vector field are identical to the requirements for the time line vector field (see above) – with one important exception: streak lines cannot be explored with the equivalent of a single time step, since streak lines are defined in the $D \times \Upsilon$ domain. In other words, to compute streak lines using $\bar{\bar{\mathbf{q}}}$, one needs at least $n_T \cdot M$ amount of memory (the equivalent of the original flow).

Computation Times for Time Lines and Advected Stream Lines

Computing $\bar{\bar{\mathbf{q}}}$ for a time step of the 2D cylinder data set took 52 seconds single-threaded on a laptop with an Intel Core 2 Duo T9550 (2.66GHz). Computing a spatial slice of the time line or advected stream line vector fields took 4 minutes for an integration interval $\tau = -5$ in the 3D time-dependent flow around a square cylinder. More measurements are listed in Table 1.

Some of these numbers may appear rather large, maybe even almost prohibitive. For example, we need 160 minutes to compute $\bar{\bar{\mathbf{q}}}$ for all time steps in the 3D flow behind the square cylinder. However, it shall be noted that the result can be used for much more than a single picture. As mentioned earlier, this can be used to browse interactively through all advected stream surfaces of all time steps (for a given τ). If such an application is required by the user, then it seems very beneficial to use our differential description of characteristic curves instead of the classic geometric approach: the memory overhead is not too large (in the order of the original flow) and the time for computing $\bar{\bar{\mathbf{q}}}$ is easily amortized once a certain number of curves/surfaces has been computed using $\bar{\bar{\mathbf{q}}}$.

This shall be explained in more detail using time lines: Integrating a time line in $\bar{\bar{\mathbf{q}}}$ amounts to a simple tangent curve integration, whereas the classic approach amounts to a path surface integration. Comparing the two approaches, the time spent for computing $\bar{\bar{\mathbf{q}}}$ will be amortized for a certain number of time lines. In our implementation, this happens for the 2D cylinder flow at around 300 time lines, see Figure 1. After that, our new approach is faster. Similar statements apply to time surfaces and advected stream lines/surfaces.

As Figure 1 shows, our approach scales very well with the number of integral curves to be computed. As such, we see it as an orthogonal approach to the classic geometric approach:

- If *many* integral curves/surfaces (streak lines, time lines, advected stream lines, ...) are to be computed, the new differential description is to be preferred due to its overall faster computation time in this scenario (computing $\bar{\bar{\mathbf{q}}}$ plus computing the individual lines/surfaces).
- If *some* integral curves/surfaces are to be computed and the focus is on an *interactive* presentation, the new differential description is to be preferred, since it allows for the fastest computation of individual lines/surfaces.

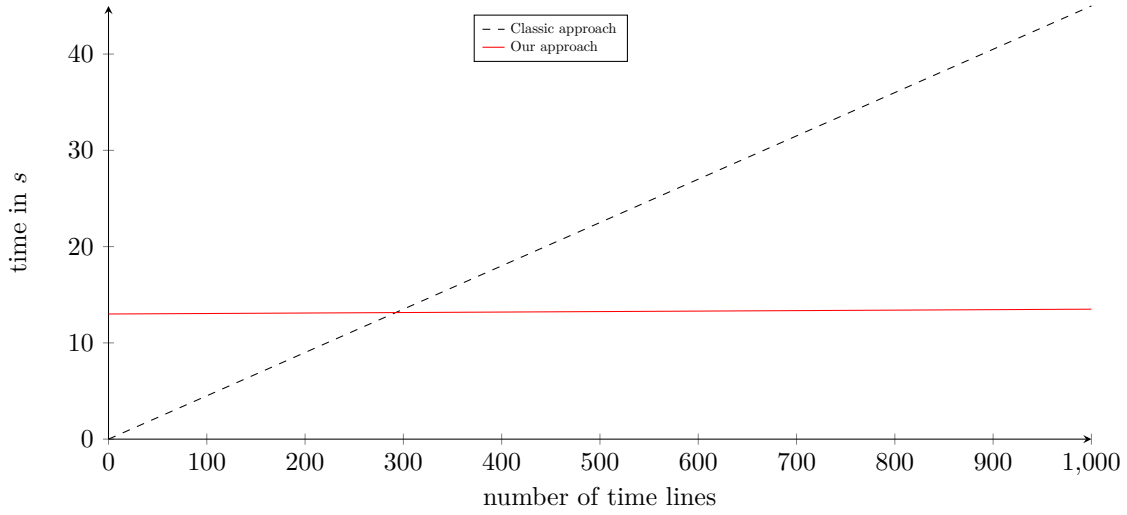

Figure 1: Computation times for integrating time lines.

- If *some* integral curves/surfaces are to be computed for a *non-interactive* application, the classic geometric approach is to be preferred, since it requires less memory and – depending on the number of lines/surfaces – may have an overall faster computation time in this scenario.
- If a *single* integral curve/surface is to be computed, the classic geometric approach is to be preferred, since it will be faster in this scenario.

Computation Times for Streak Lines

In contrast to the time line vector field and the advected stream line vector field, the computation of the streak line vector field requires temporal derivatives of the flow map. Hence, two additional path lines have to be integrated for each grid point of $\bar{\bar{\mathbf{q}}}$, which makes the computation of the streak line vector field slightly more expensive.

Parallel Computation

As discussed in Section 6.1, $\bar{\bar{\mathbf{q}}}$ is computed using a large number of path line integrations. These integrations are independent from each other. Therefore, the algorithm can easily be implemented in a parallel fashion. We did this and tested it on a quad-core i7-2720 CPU. The results for the 2D cylinder flow are given in Table 2. Note that we do not achieve the theoretical speed-up of 4.0 when using all four cores, since only the path line integration has been parallelized. Other parts of the algorithm (such as the derivation of the flow map) are still carried out in a serial fashion.

References

[WT10] T. Weinkauf and H. Theisel. Streak lines as tangent curves of a derived vector field. *IEEE TVCG*, 16(6):1225–1234, 2010.

Table 2: Time needed to compute the vector fields for time lines or advected stream lines using a parallel implementation of our algorithm. The test case is the 2D flow around a cylinder and corresponds to the second line of Table 1. The results have been obtained on a quad-core i7-2720 CPU.

Number of cores	Computation Time in seconds	Factor to the single-threaded computation
1	52.1	1.0
2	27.0	1.93
3	20.0	2.60
4	16.6	3.13