Super Space Clothoids: Supplemental material
Piecewise Summation Algorithm on a Simple 2D Case Study

Romain Casati Florence Bertails-Descoubes

INRIA and Laboratoire Jean Kuntzmann (Grenoble University, CNRS), France*

1 Introduction

In this supplemental material we highlight on a simple 2D study case how catastrophic cancellation shows up when integrating an explicit linear ODE with power series expansions. We take as a study case a simple 2D Cauchy problem on $SO(2)$ on which catastrophic cancellation occurs when naively computing the sum of the series, and we explain in detail how to remedy to this loss of precision thanks to our piecewise summation algorithm. Unlike our actual kinematic problem, the 2D problem considered here possesses an explicit, closed-form solution, which allows for a straightforward evaluation of the accuracy of employed summation algorithms. We encourage the reader to test the different summation algorithms that we provide here in pseudo-code.

2 Simple 2D Case Study

Let us consider the following Cauchy-Lipschitz problem on $SO(2)$,

$$\left\{ \begin{array}{rcl} \mathscr{A}' & = & J\mathscr{A} \\ \mathscr{A}(0) & = & I_2 \end{array} \right. \quad \text{where} \quad J = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}. \tag{1}$$

The unique solution to that problem can be explicitly formulated in closed-form as

$$\mathscr{A}(s) = \exp(sJ) = \begin{pmatrix} \cos s & -\sin s \\ \sin s & \cos s \end{pmatrix}. \tag{2}$$

However, suppose we are not aware of this expression and want to evaluate the solution $\mathscr{A}(s)$ of (1) at any point $s > 0$ using the power series expansion $\sum_{n=0}^{\infty} \tilde{\mathscr{A}}_n(s)$. The corresponding recursion is

$$\left\{ \begin{array}{rcl} \tilde{\mathscr{A}}_0 & = & I_2 \\ \tilde{\mathscr{A}}_n(s) & = & \dfrac{s}{n+1} J \tilde{\mathscr{A}}_n(s), \end{array} \right.$$

and the naive algorithm writes

```
Matrix2d function naiveSum(double s, Matrix2d init) {
  // Initialize general term of the series
  Matrix2d term = init;
  // Initialize accumulation matrix
  Matrix2d sum  = term;
  for (int n=0; term.norm()>eps; n++) {
    // Apply induction
    term *= {0., -1., 1., 0.} * s / (n+1);
    // Add
    sum  += term;
  }
  return sum;
}
```

Catastrophic cancellation When testing the algorithm above, one realizes, as illustrated in Figure 1 by comparing the result with the exact solution given by (2), that a dramatic precision loss occurs. Instead of remaining bounded in norm as expected, the result is approximated by a value that is all the more erroneous as s is large, and becomes unacceptable when the error is the same order of the computed quantity. Formulating the recursion for the norm

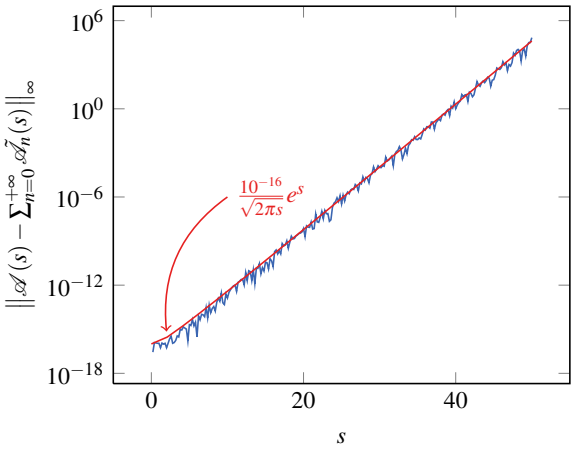

Figure 1: *Exponential increasing with s of the error made when evaluating the solution of problem (1) with power series expansion, compared to the result given by the exact formula (2), in log scale.*

of the general term $\left\|\tilde{\mathscr{A}}_n(s)\right\|_\infty$ gives

$$\left\|\tilde{\mathscr{A}}_{n+1}(s)\right\|_\infty = \frac{s}{n+1} \left\|\tilde{\mathscr{A}}_n(s)\right\|_\infty \qquad (\text{with } s > 0) \tag{3}$$

i.e.,

$$\left\|\tilde{\mathscr{A}}_n(s)\right\|_\infty = \frac{s^n}{n!} \qquad (\text{with } s > 0) \tag{4}$$

where we identify, like in our own kinematic problem, the general term of the power series of the exponential function evaluated in s, $\tilde{e}_n(s) = \frac{s^n}{n!}$. The hillock-like profile of this general term, represented in the blue curves in Figure 2, implies that when computing the sum of the series, one actually adds very small values together with very large ones in norm, the widest range being obtained when getting to the top of the hillock. Note also that the larger s is, the higher the top of the hillock is. Moreover, entries of the matrices to be added are of alternating sign (due to the product with the skew-symmetric matrix J), leading to cancellation when computing the sum. All this combined together, it is then not surprising that we are faced with a catastrophic cancellation issue when s becomes *too large*.

Limiting the range of summands for guaranteeing precision Let us precisely quantify the "top of the hillock" of Figure 2 and see how it evolves with s. The maximum of the general term $\tilde{e}_n(s) = \dfrac{s^n}{n!}$ over $n \in \mathbb{N}$ is reached when $n = \lfloor s \rfloor$, where $\lfloor . \rfloor$ denotes the floor function. At the maximum, the term of the series thus reads

$$\max_n \tilde{e}_n(s) = \tilde{e}_{\lfloor s \rfloor}(s) = \frac{s^{\lfloor s \rfloor}}{\lfloor s \rfloor !}$$

*e-mail:{romain.casati,florence.descoubes}@inria.fr

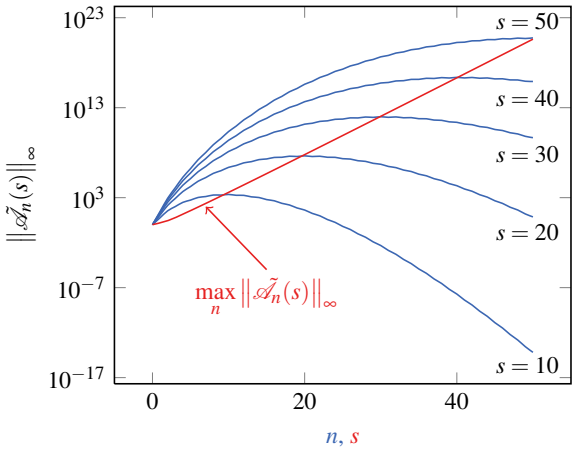

Figure 2: *General term* $\left\|\tilde{\mathscr{A}}_n(s)\right\|_\infty = \frac{s^n}{n!}$. *In blue: Evolution of* $\left\|\tilde{\mathscr{A}}_n(s)\right\|_\infty$ *function of n, in log scale. As expected, the decreasing towards 0 appears to be super-linear. In red: Evolution, function of s, of* $\max_n \left\|\tilde{\mathscr{A}}_n(s)\right\|_\infty$*, in log scale. As predicted by Equation (5), this evolution is asymptotically exponential.*

and at the limit when s tends to infinity, we get

$$\max_n \tilde{e}_n(\lfloor s \rfloor) \underset{s\to+\infty}{\sim} \frac{e^{\lfloor s \rfloor\left(1+\log\frac{s}{\lfloor s \rfloor}\right)}}{\sqrt{2\lfloor s \rfloor \pi}}, \tag{5}$$

meaning that for large s, the top of the hillock grows quasi-exponentially with s (see Figure 2, red curve). To prevent the absorption of significant digits and thus avoid catastrophic cancellation, a natural idea then consists in *upper-bounding* s by a value M depending on the machine precision, so that the top of the hillock (approximated with the above equivalent) remains within the range where additions between two numbers can be safely performed with no absorption of their leading digit. More precisely, if the machine possesses a precision of 10^{-d} ($d = 7$ for a floating number encoded on 32 bits, $d = 16$ on 64 bits), then we aim at upper-bounding the top of the hillock by $10^{\frac{d}{2}}$ so as to be able to safely cover additions on the range $[10^{-\frac{d}{2}}, 10^{\frac{d}{2}}]$. This leads to the following inequality for M,

$$M \leqslant \max\left\{n \in \mathbb{N} \text{ s.t. } (n+1)^n \leqslant 10^{\frac{d}{2}} n!\right\} \tag{6}$$

Using a simple iterative algorithm on integers, one finds $M \leqslant 19$ for $d = 16$.

Piecewise computation algorithm The solution $\mathscr{A}$ of problem (1) can now be evaluated safely at s by splitting the interval $[0, s]$ into subintervals as described in the paper, Section 6.3. Since J is constant w.r.t s and $\lambda(s) = s$ in this case, all subintervals have the same maximum length M. Then $[0, s]$ is split into $p = \lceil \frac{s}{M} \rceil$ subintervals and problem (1) is iteratively solved on each subinterval. Let $\mathscr{A}^k(u) = \mathscr{A}\left(k\frac{s}{p} + u\right)$ be the shifted restriction of $\mathscr{A}$ on $\left[k\frac{s}{p}; (k+1)\frac{s}{p}\right]$. Note that for integrating (1) on the subinterval k, the initial condition $\mathscr{A}(0) = I_2$ has to be modified as $\mathscr{A}^k(0) = \mathscr{A}^{k-1}(\frac{s}{p})$, the rest of the equation remaining unchanged. The new algorithm with guaranteed precision now reads

```
Matrix2d function piecewiseSum(double s, Matrix2d init) {
  // Initial value
  Matrix2d initk = init;
  // Compute the number p of subintervals
  int p = ceiling(s/M);
  // Safely compute the sum on each subinterval
  for (int k=0; k<p; k++) {
    // Compute the series at the end of the segment
    initk = naiveSum(s/p, initk)
  }
  return initk;
}
```

Interestingly, this method actually appears to be the generalization of the third method described in [Moler and Loan 2003] for computing matrix exponential, which is said to be "one of the most effective". Of course, our algorithm could be optimized by considering that problem (1) is an *autonomous* system, i.e., taking into account the fact that the coefficient J is independent of s. However, this algorithm was designed for the larger class of *non-autonomous*, explicit linear ODEs with polynomial coefficients (such as our kinematic problem). It can thus be safely used to compute the solution of more complex ODEs with a high precision.

References

MOLER, C., AND LOAN, C. V. 2003. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM Review 45*, 1, 3–49.