Parameter choice for the semi-global propagator

Ido Schaefer

February 7, 2025

1 General background

Most ODE solvers have only one free parameter, usually the time step size, Δt. The current version of the semi-global propagator (version 1) has 3 free parameters to be provided by the user. This complicates its application compared to most ODE solvers. This short guide provides some practical guidance for the parameter choice.

Ideally, one should be familiar with the mathematical background of the propagator for a successful choice of the parameters (for a review, see [1]). Nevertheless, the approach of this guide is to give some practical guidance even with no familiarity with the mathematical background.

Three free parameters have to be inserted into the function that implements the propagation, `SemiGlobal1`:

1. `Nts`, the number of time-steps (N_t in [1]); the corresponding time-step size (Δt in [1]) is `Tts = T/Nts`, where `T` is the length of the entire propagation time-interval.
2. `Nt_ts`, the number of internal time-points in each time-step (M in [1]);
3. `Nfm`, the number of expansion terms for the function of matrix computation in each time-step (K in [1]).

Another parameter, the number of iterations (`niter` in the code) is determined adaptively for each time-step. If desired, it can be determined manually by the user by the application of the manual mode (see the description of the tolerance input parameter `tol` in the comments).

The parameters determine the error of the computation. The algorithm has three main error sources, all of them depend on the parameter choice:

1. *Time-expansion error* (or *time discretization error*);
2. *Function of matrix computation error*;
3. *Convergence error.*

A detailed explanation on the different error sources is available in [1, Appendix D]. Nevertheless, the present text does not assume familiarity with the meaning of the different error sources.

The adaptive determination of the number of iterations in the current version provides control of the *convergence error* only. The requested tolerance, represented by the input parameter `tol`, provides an *upper limit* for the convergence error. The resulting convergence error can be much smaller, since the input parameter set may not enable larger convergence error, even for a single iteration. The other error sources might become larger than `tol`. The error is estimated by the algorithm. In case that the estimated error exceeds `tol`, a warning message will be displayed (as a default, at the end of the propagation process; there is an option to display the error warnings during the propagation right when this is detected. This is done by the application of the display mode, see comments).

The numerical stability of the algorithm also depends on the parameter choice. We can distinguish between two types of numerical instability:

1. *Convergence failure*: The iterative process does not converge, but leads to divergence. When this behaviour is detected, the following warning message is displayed:
   `Warning: Convergence failure.`

2. *Divergence with propagation*: The error accumulates explosively with the propagation process. This behaviour can be predicted by the code, based on the *instability factor*. Typically, the algorithm becomes unstable when the instability factor is of order of ∼0.1-1 or more. If the instability factor exceeds 0.1, the following warning message is displayed:
   `Warning: Instability in the propagation process may occur.`

More details on the different types of instability are available in [1, Appendix D].

The parameter choice *must* fulfill the following requirements:

1. The overall error is smaller than the required tolerance, specified by the input variable `tol`.

2. The computational process is stable.

In addition, the parameter choice *should* fulfill the following requirements, related to the *efficiency* of the algorithm:

1. The average number of iterations is close to 1. The origin of this requirement is as follows. Although the algorithm controls the convergence error by an adaptive choice of the iteration number in each time step, typically this is not the ideal way to control the convergence error; it can be controlled more efficiently by the variation of the input parameters. Thus, usually a single iteration will be the most efficient, where the other parameters are adjusted accordingly. For very high precision requirements (`tol` $\lesssim 10^{-10}$) this recommendation may not be valid. Anyway, if the average number of iterations is more than 3, this usually indicates that something is wrong (numerical

instability, too large extrapolation error of the guess solution or bad convergence properties). The average number of iterations is stored in `history['mniter']` in the Python code and in the variable `mniter` in the MATLAB code.

2. The resulting error is not much smaller than the tolerance requirements. Moreover, if the error resulting from one of the error sources is much smaller than the overall error, this might indicate that unnecessary numerical effort is invested in reducing it to a lower level than required, with a negligible effect on the final precision of the result. The ideal parameter choice avoids this. The error can be considered too small when it is at least two orders of magnitude below the required tolerance.

2 Parameter choice

Before giving general recommendations on the parameter choice, first we shall mention the legal range of the different parameters in the code:

- `Nts`: Any positive integer;
- `Nt_ts`: Any integer above 1 (the algorithm as defined in [1, Section 3.1] is undefined for $M = 1$, see Eq. (76) therein. However, in principle, the algorithm can be defined for $M = 1$ for a different set of Chebyshev points, see Appendix A.2 therein).
- `Nfm`: Any integer above 1 (in principle, there is no theoretical limitation to define the algorithm also for $K = 1$; however, the current version of the code does not handle this uncommon case); for the Arnoldi algorithm, `Nfm` cannot exceed the dimension of the Hilbert space; this is relevant to problems of a very small dimension.

In the current version of the code, the default maximal value of `Nt_ts` is 13, and of `Nfm` is 15. These default values can be changed by the user (see comments in the beginning of the program).

The discussion of the parameter choice will be divided into two parts:

1. *Initial choice*: The parameter choice for solving a new problem at the first time, without any previous knowledge;
2. *Manual adaptation of parameters*: After solving the problem, it may be found that one or more of the mentioned requirements (regarding the magnitude of the error, stability and efficiency) is not matched by the parameter choice. Then it is possible to correct the parameter choice for a new run of the same problem if required (in case of too large error or suspicion of instability). Moreover, if a class of similar problems with slight variations has to be solved, a successful parameter choice for one problem will probably fit all the others. In this case, the correction of the chosen parameter set to a more efficient one can be applied to all other problems.

2.1 Initial choice

Usually, a good choice for `Nt_ts` is in the range 7-9. Using larger `Nt_ts` can be of advantage; however, the risk of numerical instability and other roundoff error effects increases with `Nt_ts`. Thus, to be on the safe side, `Nt_ts` should not exceed 9. Typically, the algorithm becomes inefficient or completely fails around `Nt_ts` $\sim 13-14$.

As a rule of thumb, a safe choice of `Nfm` satisfies

$$\texttt{Nt\_ts} + \texttt{Nfm} = 14 \tag{1}$$

Often, smaller `Nfm` suffices. A more precise choice of `Nfm` can be easily adjusted based on the estimated function of matrix error, as will be described in Sec. 2.2.

For the sake of numerical stability, it is recommended to choose `Nts` such that the time-step Δt satisfies

$$|\lambda_{max}|\,|\Delta t| \lesssim 10 \tag{2}$$

where λ_{max} is the largest eigenvalue of $\widetilde{G}$. Typically, around the value of 10 in the RHS the phenomenon of divergence with propagation appears.

2.2 Manual adaptation of parameters

In what follows, we will describe the typical effects of the variation of the different parameters on the error sources and the stability. In addition, we will discuss the recommended actions when each of the mentioned requirements is not satisfied by the parameter choice.

2.2.1 Error control

1. *Time-expansion error*:

    - *Effect of the free parameter variations*: The error decreases with the increase of `Nts` (i. e. decrease of the time-step size) or `Nt_ts`. The variation of `Nfm` has no significant effect.
    - *Recommended action when is too large*: Usually, the increase of `Nt_ts` is the most efficient action. If `Nt_ts` reaches the "unsafe" region of above 9 (see Sec. 2.1), the increase of `Nts` should be preferred.
    - *Recommended action when is too small*: No action is required. The decrease of `Nt_ts` has a prominent effect on the other error sources, and thus it should not be varied unless the other error sources are too small either. Typically, when the parameters are chosen correctly, the time-expansion error has the smallest magnitude of the error sources.

2. *Function of matrix computation error*:

    - *Effect of the free parameter variations*: The error decreases with the increase of all free parameters, `Nts`, `Nt_ts` and `Nfm`.

- *Recommended action when is too large*: The increase of `Nfm` is the direct action, and usually the most recommended one. Most frequently, the increase of `Nt_ts` leads to similar decrease in the magnitude of the error; however, the numerical cost of the increment of `Nt_ts` can be larger (when the input variable `Gdiff_matvecs` > 0).
- *Recommended action when is too small*: Decrease of `Nfm`.

3. *Convergence error*:

   - *Effect of the free parameter variations*: For a fixed number of iterations, the convergence error decreases with the increase of `Nts` or `Nt_ts`. Seldom, the increase of `Nfm` also decreases the error (the effect of `Nfm` is more likely to be observed for `Nt_ts` values smaller than recommended in Sec. 2.1).
   - *Recommended action when the average iteration number is too large*: Usually, the increase of `Nt_ts` is the most efficient action. If `Nt_ts` reaches the "unsafe" region of above 9 (see Sec. 2.1), the increase of `Nts` should be preferred.
     In the manual mode, recommended action when the convergence error is too large: Same as above.
   - *Recommended action when the convergence error is too small*: If the average number of iterations does not exceed 1, the decrease of `Nts` should be considered. The same applies for the manual mode for any choice of a fixed iteration number. Note that the decrement of `Nts` will increase also the magnitude of the other error sources, and thus the increase of `Nt_ts` or `Nfm` might be required accordingly. In the following situations the decrease of `Nts` should be avoided:
     - The value of `Nt_ts` is already 9 and the time-expansion error is within an order of magnitude below the tolerance (uncommon);
     - The instability factor is close to the "unsafe" region of 0.1 (see Sec. 1).

2.2.2 Stability control

1. *Convergence failure*: The only parameter that has a significant effect on the convergence properties is `Nts`; thus, the recommended action is the increment of `Nts`.

2. *Divergence with propagation*:

   - *Effect of the free parameter variations*: The instability factor decreases with the increase of `Nts` and `Nfm`. Typically, it also decreases with the increase of `Nt_ts`, however with a slow rate. An increase of the instability factor with `Nt_ts` can be also observed occasionally. As was mentioned in Sec. 1, the stability increases as the instability factor decreases.
   - *Recommended action*: The increase of `Nfm` is usually the most recommended action. When the function of matrix computation error is so small that it approaches the roundoff error regime, `Nts` should be increased instead. Typically, in a correct choice of parameters `Nfm` does not exceed 9.

	`Nts`	`Nt_ts`	`Nfm`
Time-expansion error	✓	✓	X
Function of matrix error	✓	✓	✓
Convergence error for a fixed iteration number; average iteration number	✓	✓	X
Instability factor	✓	✓	✓

Table 1: Table of typical effects of the free parameter variations. Each box in the table indicates whether the increase of the corresponding parameter typically reduces the corresponding property, where the other parameters remain fixed. Uncommon exceptions are possible (see main text).

Table 2.2.2 summarizes the typical effects of the free parameter variations on the errors, average iteration number and the instability factor. Table 2.2.2 summarizes the typically recommended actions for troubleshooting the precision, efficiency and stability issues mentioned above.

References

[1] I. Schaefer, H. Tal-Ezer, and R. Kosloff, "Semi-global approach for propagation of the time-dependent Schrödinger equation for time-dependent and nonlinear problems," *Journal of Computational Physics*, vol. 343, pp. 368 – 413, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0021999117302887

Event	Recommended action
Time-expansion error is too large	If `Nt_ts`<9 increase `Nt_ts`; otherwise increase `Nts`
Time-expansion error is too small	No action required.
Function of matrix error is too large	Increase `Nfm`.
Function of matrix error is too small	Decrease `Nfm`.
Average iteration number is too large; in the manual mode, convergence error is too large	If `Nt_ts`<9 increase `Nt_ts`; otherwise increase `Nts` .
Convergence error is too small	If the average number of iterations is 1 decrease `Nts` and adjust other parameters accordingly.
Convergence failure	Increase `Nts`.
Divergence with propagation	Increase `Nfm`.

Table 2: Summary of typically recommended actions for troubleshooting precision, efficiency and stability issues (see main text for more details and exceptional cases).