The Journal of Open Source Software

`riccati`: an adaptive, spectral solver for oscillatory ODEs

Fruzsina J. Agocs[1] **and Alex H. Barnett**[1]

1 Center for Computational Mathematics, Flatiron Institute, 162 Fifth Avenue, New York, 10010 NY, USA

DOI: 10.21105/joss.05430

Software

- Review
- Repository
- Archive

Editor: Patrick Diehl

Reviewers:

- @mseri
- @victorapm

Submitted: 04 April 2023
Published: 13 June 2023

Summary

Highly oscillatory ordinary differential equations (ODEs) pose a computational challenge for standard solvers available in scientific computing libraries. These conventional methods are typically based on a polynomial approximation, resulting in there being several timesteps per oscillation period, which leads to runtimes scaling as $\mathcal{O}(\omega)$, with ω being the oscillation frequency. This can become prohibitively slow.

The `riccati` (Python) package implements the efficient numerical method described in Agocs & Barnett (2022) (dubbed ARDC for adaptive Riccati defect correction) for solving ODEs of the form

$$u''(t) + 2\gamma(t)u'(t) + \omega^2(t)u(t) = 0, \quad t \in [t_0, t_1], \tag{1}$$

subject to the initial conditions $u(t_0) = u_0$, $u'(t_0) = u'_0$. The frequency $\omega(t)$ and friction $\gamma(t)$ terms are given smooth real-valued functions (passed in as callables). The solution $u(t)$ may vary between highly oscillatory and slowly-changing over the integration range, in which case `riccati` will switch between using nonoscillatory (spectral Chebyshev) and a specialised oscillatory solver (Riccati defect correction) to achieve an $\mathcal{O}(1)$ (frequency-independent) runtime. It automatically adapts its stepsize to attempt to reach a user-requested relative error tolerance. The solver is capable of producing *dense output*, i.e., it can return a numerical solution estimate at a user-selected set of t-values, at the cost of a few arithmetic operations per t-point.

Statement of need

Specialised numerical methods exist to solve Equation 1 in the high-frequency ($\omega \gg 1$) regime, but of those that have software implementations, none are both (1) able to deal with both oscillatory and nonoscillatory behaviors occuring in the solution; and (2) high-order accurate, so that the user may request many digits of accuracy without loss of efficiency. `riccati` fills this gap as a spectral adaptive solver. By spectral, we mean that an arbitrarily high order p may be chosen (e.g. $p = 16$), allowing a high convergence rate that is limited only by the smoothness of the coefficients, and (in the nonoscillatory case) that of the solution.

Being a spectral solver means that its convergence rate is as quick as the smoothness of the coefficients $\omega(t)$, $\gamma(t)$ (in the oscillatory regime), and that of the solution $u(t)$ (in the nonoscillatory regime) allows. `oscode` (Agocs, 2020; Agocs et al., 2020) and the WKB-marching method[1] (Arnold et al., 2011; Körner et al., 2022) are examples of low-order adaptive oscillatory solvers, efficient when no more than about 6 digits of accuracy are required or $\omega(t)$ is near-constant. A high-order alternative is the Kummer's phase function-based method (Bremer, 2018, 2022), whose current implementation supports solving Equation 1 in the highly

[1] Available from https://github.com/JannisKoerner/adaptive-WKB-marching-method.

oscillatory regime when $\gamma \equiv 0$. Other existing numerical methods have been reviewed, e.g., in Petzold et al. (1997). Figure 1 compares the performance of the above specialised solvers and one of SciPy's (Virtanen et al., 2020) built-in methods (Dormand & Prince, 1980) by plotting their runtime against the frequency parameter λ while solving

$$u'' + \omega^2(t)u = 0, \quad \text{where} \quad \omega^2(t) = \lambda^2(1 - t^2\cos 3t), \tag{2}$$

on the interval $t \in [-1, 1]$, subject to the initial conditions $u(-1) = 0$, $u'(-1) = \lambda$. The runtimes were measured at two settings of the required relative tolerance ε, 10^{-6} and 10^{-12}. The figure demonstrates the advantage `riccati`'s adaptivity provides at low tolerances. `riccati` avoids the runtime increase `oscode` and the WKB marching method exhibit at low-to-intermediate frequencies, and its runtime is virtually independent of the oscillation frequency.

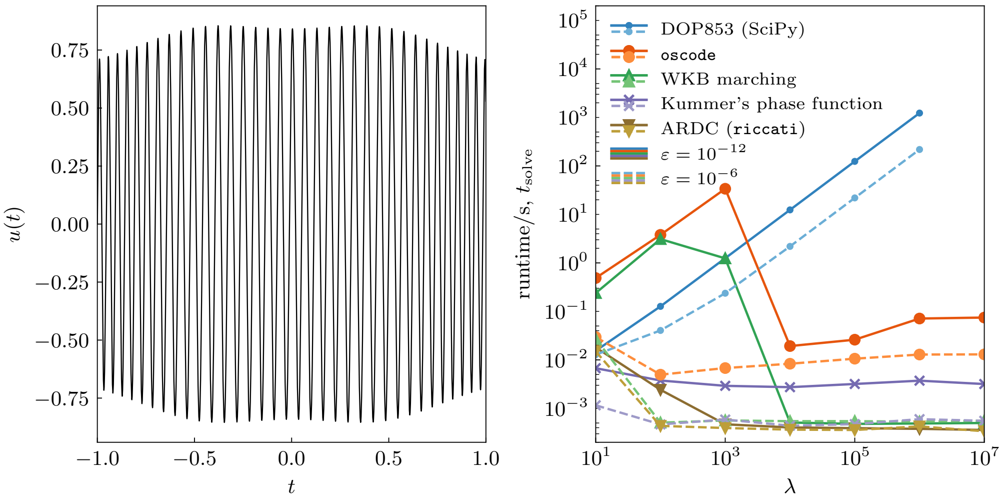

Figure 1: Left: Numerical solution of Equation 2 with $\lambda = 10^2$. Right: performance comparison of `riccati` (labelled ARDC) against state-of-the-art oscillatory solvers. `oscode`, the WKB marching method, Kummer's phase function method, and a high-order Runge–Kutta method (RK78) (Dormand & Prince, 1980) on Equation 2 with a varying frequency parameter λ. Solid and dashed lines denote runs with a relative tolerance settings of $\varepsilon = 10^{-12}$ and 10^{-6}, respectively.

Acknowledgements

We thank Jim Bremer, Charlie Epstein, Manas Rachh, and Leslie Greengard for useful discussions. The Flatiron Institute is a division of the Simons Foundation.

References

Agocs, F. J. (2020). (py)Oscode: Fast solutions of oscillatory ODEs. *Journal of Open Source Software*, *5*(56), 2830. https://doi.org/10.21105/joss.02830

Agocs, F. J., & Barnett, A. H. (2022). *An adaptive spectral method for oscillatory second-order linear ODEs with frequency-independent cost*. arXiv. https://doi.org/10.48550/arxiv.2212.06924

Agocs, F. J., Handley, W. J., Lasenby, A. N., & Hobson, M. P. (2020). Efficient method for solving highly oscillatory ordinary differential equations with applications to physical systems. *Physical Review Research*, *2*(1), 013030. https://doi.org/10.1103/PhysRevResearch.2.013030

Arnold, A., Abdallah, N. B., & Negulescu, C. (2011). WKB-based schemes for the oscillatory 1D Schrödinger equation in the semiclassical limit. *SIAM Journal on Numerical Analysis*, *49*(4), 1436–1460. https://doi.org/10.1137/100800373

Bremer, J. (2018). On the numerical solution of second order ordinary differential equations in the high-frequency regime. *Applied and Computational Harmonic Analysis*, *44*(2), 312–349. https://doi.org/10.1016/j.acha.2016.05.002

Bremer, J. (2022). Phase function methods for second order linear ordinary differential equations with turning points. *arXiv Preprint arXiv:2209.14561*. https://doi.org/10.48550/arxiv.2209.14561

Dormand, J. R., & Prince, P. J. (1980). A family of embedded Runge-Kutta formulae. *Journal of Computational and Applied Mathematics*, *6*(1), 19–26. https://doi.org/10.1016/0771-050X(80)90013-3

Körner, J., Arnold, A., & Döpfner, K. (2022). WKB-based scheme with adaptive step size control for the Schrödinger equation in the highly oscillatory regime. *Journal of Computational and Applied Mathematics*, *404*, 113905. https://doi.org/10.1016/j.cam.2021.113905

Petzold, L. R., Jay, L. O., & Yen, J. (1997). Numerical solution of highly oscillatory ordinary differential equations. *Acta Numerica*, *6*, 437–483. https://doi.org/10.1017/S0962492900002750

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., … SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in python. *Nature Methods*, *17*, 261–272. https://doi.org/10.1038/s41592-019-0686-2