riccati: an adaptive, spectral solver for oscillatory ODEs

Fruzsina J. Agocs and Alex H. Barnett

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

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 $O(\omega)$, with $\omega$ 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

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

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 $O(1)$ (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 occurring 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 method1 (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

---


---

https://doi.org/10.21105/joss.05430.
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 $\lambda$ while solving

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

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 $\varepsilon$, $10^{-6}$ and $10^{-12}$. The figure demonstrates the advantage of 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 $\lambda$. Solid and dashed lines denote runs with a relative tolerance settings of $\varepsilon = 10^{-12}$ and $10^{-6}$, respectively.](image)

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


