The Journal of Open Source Software

(py)oscode: fast solutions of oscillatory ODEs

Fruzsina Julia Agocs[1, 2]

1 Astrophysics Group, Cavendish Laboratory, J. J. Thomson Avenue, Cambridge, CB3 0HE, UK **2** Kavli Institute for Cosmology, Madingley Road, Cambridge, CB3 0HA, UK

DOI: 10.21105/joss.02830

Software

- Review
- Repository
- Archive

Editor: Melissa Weber Mendonça

Reviewers:

- @jakryd
- @dlfivefifty

Submitted: 07 October 2020
Published: 15 December 2020

Summary

Oscillatory differential equations are ubiquitous in physics, chemistry and beyond. They arise in quantum mechanics, electrical circuitry, suspension systems, molecular dynamics, and in models of gravitational and electromagnetic waves. The numerical solution of such systems however can be a computational bottleneck when tackled with conventional methods available from numerical libraries.

We present `(py)oscode`, a general-purpose numerical routine for solving a class of highly oscillatory ordinary differential equations (ODEs) efficiently. The package has been designed to solve equations which describe a single harmonic oscillator with a time-dependent frequency and damping term, i.e. are of the form

$$y'' + 2\gamma(x)y' + \omega^2(x)y = 0. \tag{1}$$

The frequency $\omega(x)$ and damping $\gamma(x)$ terms do not need to be explicit functions of x (they can instead be e.g. the result of another numerical solution of an ODE), as they are supplied as sequences ω_j, γ_j evaluated at $x_i \leq x_j \leq x_f$, where (x_i, x_f) is the integration range.

`(py)oscode` is written in C++, but comes with a Python wrapper. Its Python interface was designed to be similar to those included in `SciPy`'s (Virtanen et al., 2020) numerical ODE solution modules. This is demonstrated in the example below whose output is shown in Figure 1.

```
import numpy as np
import scipy.special as sp
import pyoscode

# Set up the Airy equation as an example: y'' + xy = 0
xs = np.linspace(0,40.0,5000)
ws = np.sqrt(xs)
gs = np.zeros_like(xs)
# Initial conditions
xi = 1.0
xf = 40.0
yi = sp.airy(-xi)[0]
dyi = -sp.airy(-xi)[1]
# Get dense output at the following points
t_eval = np.linspace(15,35,600)
# Solve the equation
solution = pyoscode.solve(xs, ws, gs, xi, xf, yi, dyi, t_eval=t_eval)
```

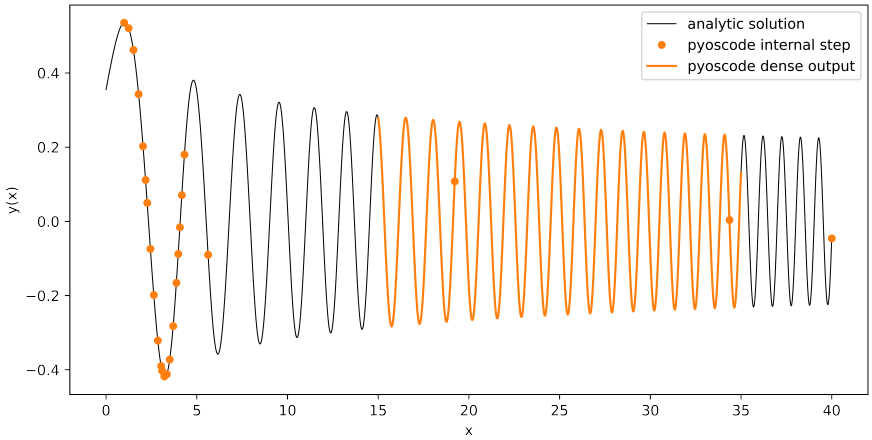

Figure 1: Numerical solution of the Airy equation, $y'' + xy = 0$, with `pyoscode`. The increase in step-size of `pyoscode`'s internal steps (orange dots) is due to the algorithm switching from using the RK method to the WKB approximation in the presence of high-frequency oscillations. The orange segment shows dense output, the solution at these points was computed at no additional evaluations of terms in the differential equation.

Statement of need

Even if the terms in Equation 1 change slowly, if the frequency of oscillations in the solution is high enough, standard numerical methods struggle to solve such equations quickly. Traditional methods have to trace every oscillation in the solution, taking many steps in x at an enormous computational cost. The algorithm underlying `(py)oscode`, published in Agocs, Handley, et al. (2020) and based on W. J. Handley et al. (2016), can detect when the solution is oscillatory and switch to a method based on an analytic approximation (Wentzel–Kramers–Brillouin, WKB) suited for oscillatory functions, otherwise using a Runge–Kutta (RK) method. Using the WKB approximation allows the algorithm to skip over several wavelengths of oscillation in a single step, reducing the number of steps taken drastically. It adaptively updates its step-size to keep the local numerical error within a user-specified tolerance. `(py)oscode` is capable of producing a solution estimate at an arbitrary value of x, not just at its internal steps, therefore it can be used to generate a "continuous" solution, or dense output (Agocs, Hobson, et al., 2020).

Related research and software

`(py)oscode`'s development was motivated by the need for a significantly more efficient solver for the evolution of early-universe quantum fluctuations. These perturbations are thought to have been stretched to macroscopic scales by a phase of accelerated expansion of the universe (cosmic inflation), to later become the large-scale structure we see today. To understand the origins of structure it is therefore essential to model the perturbations and understand the physics involved in inflation. `(py)oscode` has been used to speed up the numerical evolution of quantum fluctuations in the early universe, enabling the exploration of models beyond the standard model of cosmology (W. Handley, 2019). It served as inspiration for other numerical methods aiming to extend the range of oscillatory ODEs to solve (Bamber & Handley, 2020).

The efficient solution of oscillatory ODEs is a long-standing numerical analysis problem with many existing methods to handle certain sub-classes of equations. Examples include successful

methods by Petzold (Petzold, 1981), reviewed in Petzold et al. (1997) with many references therein, Iserles et al. (Condon et al., 2009, 2011; Deaño et al., 2017), and Bremer (Bremer, 2018a), with code available from Bremer (2018b).

Acknowledgements

I thank Lukas Hergt for invaluable discussions during the early development of (py)oscode and his ongoing support. Construction of the algorithm would not have been possible without the help and guidance of Will Handley, Mike Hobson, and Anthony Lasenby. I was supported by the Science and Technology Facilities Council (STFC).

References

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*, 013030. https://doi.org/10.1103/PhysRevResearch.2.013030

Agocs, F. J., Hobson, M. P., Handley, W. J., & Lasenby, A. N. (2020). *Dense output for highly oscillatory numerical solutions*. http://arxiv.org/abs/2007.05013

Bamber, J., & Handley, W. (2020). Beyond the Runge-Kutta-Wentzel-Kramers-Brillouin method. *Physical Review D*, *101*, 043517. https://doi.org/10.1103/PhysRevD.101.043517

Bremer, J. (2018a). 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. (2018b). Phase functions: Fortran 90 code for solving highly oscillatory ordinary differential equations in O(1) time. In *GitHub repository*. GitHub. https://github.com/JamesCBremerJr/Phase-functions

Condon, M., Deaño, A., & Iserles, A. (2011). Asymptotic solvers for oscillatory systems of differential equations. *SeMA Journal*, *53*(1), 79–101. https://doi.org/10.1007/bf03322583

Condon, M., Deaño, A., Iserles, A., & Maczyński, K. (2009). On numerical methods for highly oscillatory problems in circuit simulation. *COMPEL - The International Journal for Computation and Mathematics in Electrical and Electronic Engineering*, *28*(6), 1607–1618. https://doi.org/10.1108/03321640910999897

Deaño, A., Huybrechs, D., & Iserles, A. (2017). *Computing highly oscillatory integrals*. Society for Industrial and Applied Mathematics. https://doi.org/10.1137/1.9781611975123

Handley, W. (2019). Primordial power spectra for curved inflating universes. *Physical Review D*, *100*, 123517. https://doi.org/10.1103/PhysRevD.100.123517

Handley, W. J., Lasenby, A. N., & Hobson, M. P. (2016). *The Runge-Kutta-Wentzel-Kramers-Brillouin Method*. http://arxiv.org/abs/1612.02288

Petzold, L. R. (1981). An efficient numerical method for highly oscillatory ordinary differential equations. *SIAM Journal on Numerical Analysis*, *18*(3), 455–479. https://doi.org/10.1137/0718030

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