The Journal of Open Source Software

PathFinder: A Matlab/Octave package for oscillatory integration

Andrew Gibbs [1]

1 University College London, United Kingdom

DOI: 10.21105/joss.06902

Software

- Review
- Repository
- Archive

Editor: Olexandr Konovalov

Reviewers:

- @YehorYudinIPP
- @fruzsinaagocs

Submitted: 06 February 2024
Published: 30 October 2025

Summary

Oscillatory integrals arise in models of a wide range of physical applications, from acoustics to quantum mechanics. `PathFinder` is a Matlab/Octave package for efficient evaluation of oscillatory integrals of the form

$$I = \int_a^b f(z) \exp(\mathrm{i}\omega g(z)) \, \mathrm{d}z, \tag{1}$$

where the endpoints a and b can be complex-valued, even infinite; $\omega > 0$ determines the angular frequency; $f(z)$ is the *amplitude function*, a non-oscillatory entire function; $g(z)$ is the *phase function*, which must be a polynomial. The basic syntax is simple:

```
I = PathFinder(a, b, f, gCoeffs, omega, N);
```

Here, `f` is a function handle representing $f(z)$, `gCoeffs` is a vector of coefficients of $g(z)$, `omega` is the frequency parameter ω, and `N` is a parameter that controls the degree of approximation.

`PathFinder` is the first black-box software that can evaluate (1) accurately, robustly, and efficiently for any $\omega > 0$. It will be useful across many scientific disciplines, for problems that were previously too computationally expensive or too mathematically challenging to solve.

Statement of need

Based on the method of Numerical Steepest Descent (Huybrechs & Vandewalle, 2006), `PathFinder` is an implementation of the algorithm described in Gibbs et al. (2024), where an earlier version of the code was used to produce numerical experiments. Since these experiments, much of the code has been rewritten in C, interfacing with Matlab/Octave via MEX (Matlab executable) functions. These are easily compiled using a single script.

Ease of use

Standard quadrature rules (midpoint rule, Gauss quadrature, etc.) are easy to use, and many open-source implementations are available. However, when applied to (1), such methods become prohibitively inefficient for large ω.

On the other hand, several methods exist for the efficient evaluation of oscillatory (large ω) integrals such as (1); a thorough review is given in Deaño et al. (2018). However, applying these methods often requires an expert understanding of the process and a detailed analysis of the integral, making such methods inaccessible to non-mathematicians. Even with the necessary mathematical understanding, models may require hundreds or thousands of oscillatory integrals to be evaluated, making detailed analysis of each integral highly challenging or impossible.

Despite being based on complex mathematics, PathFinder can be easily used by non-mathematicians. The user must simply understand the definitions of the components of (1).

Use in academic research

In many physical models, interesting physical phenomena occur in the presence of *coalescing saddle points* (see, e.g., Gibbs et al. (2024) for a definition). Examples include chemical reactions, rainbows, twinkling starlight, ultrasound pulses, and focusing of sunlight by rippling water (*NIST Digital Library of Mathematical Functions*, 2023, sec. 36.14).

Coalescing saddle points can cause steepest descent methods to break down, even in simple cases where $g(z)$ is a cubic polynomial (Huybrechs et al., 2019). By design, PathFinder is robust for any number of coalescing saddle points. This is demonstrated in Figures 1 and 2, where PathFinder has been used to model well-known optics problems with coalescing saddle points. Specifically these are the *Cusp Catastrophe*

$$\int_{-\infty}^{\infty} \exp(\mathrm{i}(z^4 + x_2 z^2 + x_1 z))\ \mathrm{d}z \tag{2}$$

and the *Swallowtail*

$$\int_{-\infty}^{\infty} \exp(\mathrm{i}(z^5 + x_3 z^3 + x_2 z^2 + x_1 z))\ \mathrm{d}z$$

respectively. More information about the physical significance of these integrals can be found in *NIST Digital Library of Mathematical Functions* (2023), Section 36.14. In these plots, each point (x_1, x_2) requires a separate evaluation of (1) and thus a separate call to PathFinder. For example, for (2), the following code was used for each (x_1, x_2):

```
PathFinder( pi, 0, ... % angles of valleys
            [], ... % no amplitude function
            [1 0 x2 x1 0],... % phase coeffs
            1, ... % frequncy parameter
            10, ... % number of quadrature points per contour
            'infcontour', [true true] % doubly infinite contour
            );
```

Note the optional input specified by 'infcontour' and the vector [true true], which tells PathFinder that *both* endpoints of the integration contour are unbounded. The first two arguments of PathFinder, which describe the endpoints of the contour, are then interpreted as angles describing complex valleys: $\exp(\mathrm{i}\pi)\infty = -\infty$ and $\exp(\mathrm{i}0)\infty = \infty$, our limits of integration. Recall that in the *standard* syntax, the first two inputs are interpreted as finite endpoints of the contour.

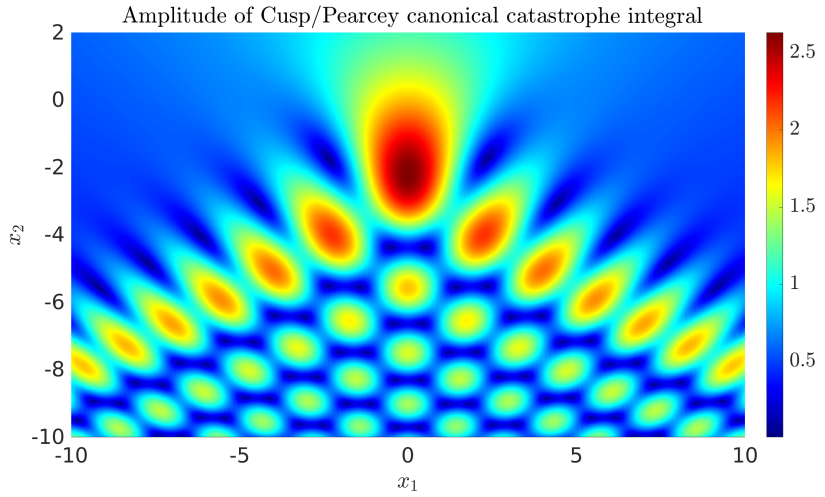

Figure 1: PathFinder approximation to Pearcey/Cusp Catastrophe integrals (Pearcey, 1946), which contain coalescing saddle points.

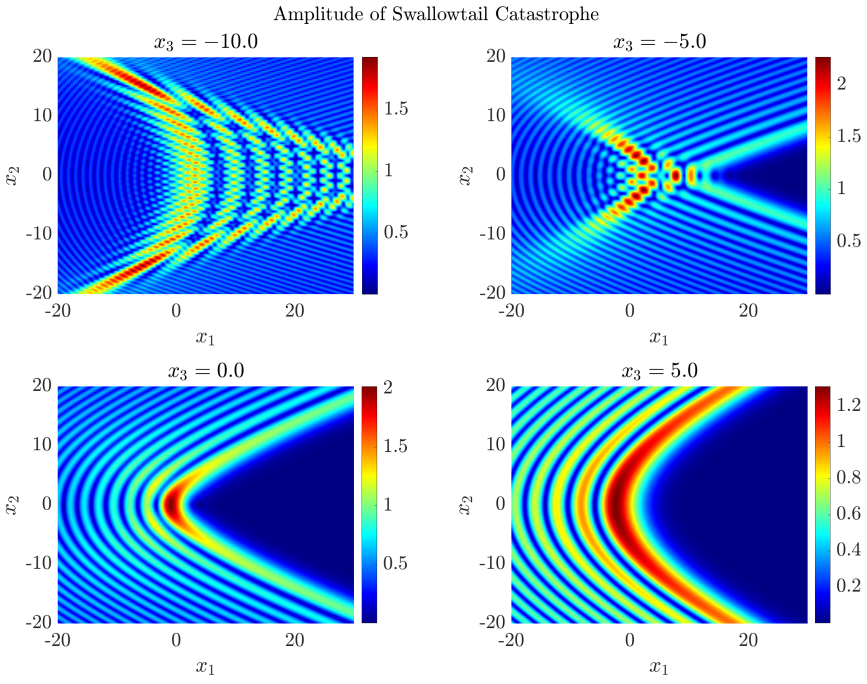

Figure 2: PathFinder approximation to Swallowtail Catastrophe integrals (Arnol'd, 1981), which contain many coalescing saddle points.

In Hewett et al. (2019) a new technique was described for the construction of integral solutions to the *Parabolic Wave Equation*, typically with coalescing saddle points. Plots of some solutions were provided using `cuspint` (described below) in the cases that were "not too difficult", but others were excluded, for example, A_{32} of equation (32) therein. This omission can now be easily produced using `PathFinder`, as shown in Figure 3.

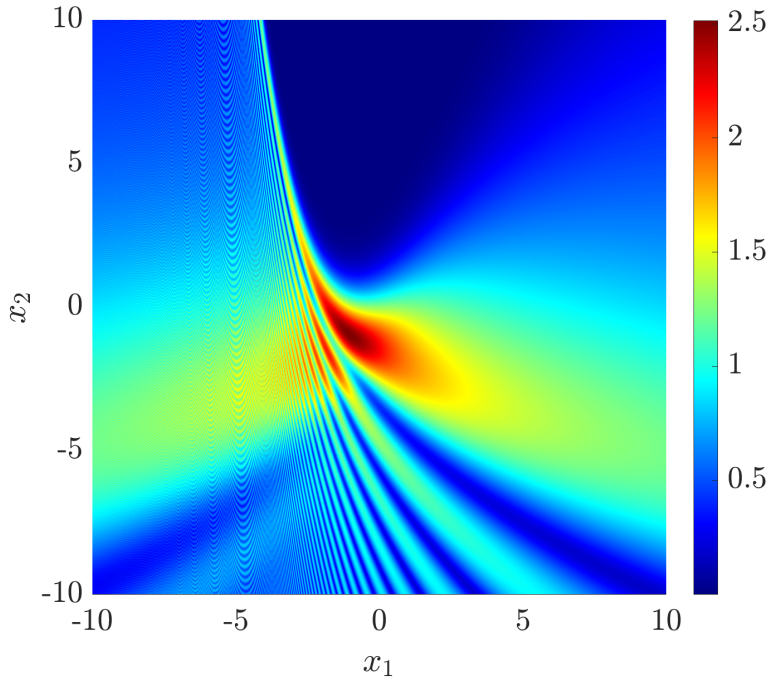

Figure 3: PathFinder approximation to $|A_{32}(x_1, x_2)|$, (32) of Hewett et al. (2019).

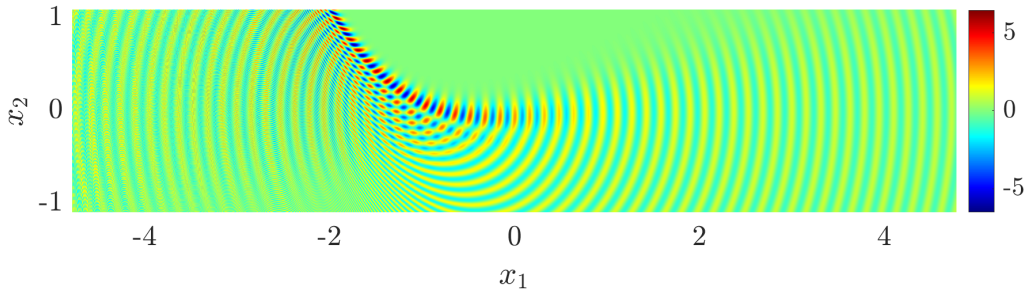

Figure 4: PathFinder approximation of a wavefield with a caustic near an inflection point, wavenumber 40.

The ideas of Hewett et al. (2019) were combined with `PathFinder` in Ockendon et al. (2024) and applied to the famous (unsolved) inflection point problem of Popov (1979). Via a simple change of variables, these solutions to the Parabolic Wave Equation could be transformed into meaningful solutions of the Helmholtz equation. Here `PathFinder` was used to visualise a wavefield with caustic behaviour close to a curve with an inflection point (as in Figure 4) and provided numerical validation of the asymptotic approximations therein. In Figure 3, the integral is

$$\int_{\mathrm{e}^{\mathrm{i}9\pi/10}\infty}^{\mathrm{i}\infty} z \exp(\mathrm{i}(-x_2 z^2 - x_1 z^4/2 + 2z^5/5))\, \mathrm{d}z; \tag{3}$$

this was subsequently transformed to a solution to the Helmholtz equation, in Figure 4. The integrals (3) are evaluated using the following code:

```
PathFinder( 9\pi/10, 1/2, ... % angles of valleys
            @(z) z, ... % amplitude function f(z)=z
            [2/5 -x_1/2 0 -x_2 0 0],... % phase coeffs
            1, ... % frequncy parameter
            10, ... % number of quadrature points per contour
            'infcontour', [true true] % doubly infinite contour
            );
```

Note again the optional input `'infcontour'` which specifies that the integration contour is unbounded.

Comparison with other software

To the best knowledge of the author, there are only a handful of other software packages that can efficiently evaluate oscillatory integrals. We now compare these to PathFinder.

Mathematica's `NIntegrate`

- This is a built-in function of Wolfram Mathematica (Wolfram, 2024), based on the Levin method (see, e.g., (Deaño et al., 2018, sec. 3.3)).
- An advantage of Mathematica's `NIntegrate` is that the oscillatory component does not always need to be factored explicitly, and it can evaluate some oscillatory integrals where g is non-polynomial.
- Based on experiments (Gibbs et al., 2024, sec. 5.3), `NIntegrate` does not appear to have a frequency-independent cost for general polynomial phase functions.
- `NIntegrate` does not work in general for an unbounded contour with complex endpoints.
- `NIntegrate` is not open source; the code cannot be seen or modified, and one must acquire a license to use it.

`cuspint`

- This package is written in Fortran, based on the paper Kirk et al. (2000).
- The `cuspint` package is somewhat similar to `PathFinder` in that it is also based on steepest descent contour deformation.
- The problem class is restricted to (1) when $(a, b) = \mathbb{R}$. Therefore, it may be used to model the catastrophe integrals of Figures 1 and 2, but not those of Figure 3 and 4.
- `cuspint` can experience "violent" exponential growth (Kirk et al., 2000, sec. 2), which can lead to inaccurate results. This is because, unlike `PathFinder`, it does not attempt a highly accurate approximation of the steepest descent contours.

`Picard_Lefschetz_Integrator`

- This C++ package is also based on steepest descent. The key difference is the algorithm gradually deforms the contour, details are given in Feldbrugge et al. (2023).
- The scope of problems to which it is applicable appears broad, the full extent is unclear based on existing documentation. Like PathFinder, it can be applied to catastrophe integrals. There are examples where it is also applied to singular oscillatory integrals.
- To the best understanding of the PathFinder developers, it appears that prior user expertise in the underlying mathematics is required to use `Picard_Lefschetz_Integrator`, various parameters must be tweaked to obtain accurate results, integrals must be manually truncated, etc. This is in contrast to PathFinder, which aims to be fully automated where possible, requiring minimal user input.

`OscillatoryIntegralsODE.jl`

- This package is based on the Levin method (see, e.g., Deaño et al. (2018), Section 3.3).

- This package can evaluate oscillatory integrals of the form

$$I = \int_a^b f(x)S(\omega x)\mathrm{d}x, \tag{4}$$

when S is a Bessel function (*NIST Digital Library of Mathematical Functions*, 2023, sec. 10.2), a Spherical Bessel function (*NIST Digital Library of Mathematical Functions*, 2023, sec. 10.47), or the Fourier oscillator $S(\omega x) = \mathrm{e}^{\mathrm{i}\omega x}$. The latter is clearly equivalent to (1) when g is a monomial, thus `OscillatoryIntegralsODE.jl` excludes the general case of PathFinder, where the high frequency oscillator has a polynomial phase function.

In summary, we believe that `PathFinder` is the only existing software package that can be applied in general to (1), without prior user expertise in the underlying mathematics.

Acknowledgments

I am very grateful for the guidance of Daan Huybrechs and David Hewett throughout the development of this software. I am also grateful for financial support from KU Leuven project C14/15/05 and EPSRC projects EP/S01375X/1, EP/V053868/1.

Some of the code in `PathFinder` is copied from other projects. I acknowledge Aryo (2024), used for the Dijkstra shortest path algorithm, originally proposed in Dijkstra (1959). I also acknowledge Dirk Laurie and Walter Gautschi for writing the code used for the Golub-Welsch algorithm, a full mathematical explanation of this algorithm can be found in Gautschi (2004).

Finally, I must express my gratitude to the referees and editor who gave their time to review the software and this paper. This paper, the code, and its documentation were significantly improved as a result of the review process.

References

Arnol'd, V. I. (1981). Lagrangian manifolds with singularities, asymptotic rays and the unfurled swallowtail. *Funktsional. Anal. I Prilozhen.*, *15*(4), 1–14, 96. https://doi.org/10.1007/BF01106152

Aryo, D. (2024). *Dijkstra algorithm*. https://www.mathworks.com/matlabcentral/fileexchange/36140-dijkstra-algorithm

Deaño, A., Huybrechs, D., & Iserles, A. (2018). *Computing highly oscillatory integrals*. SIAM. https://doi.org/10.1137/1.9781611975123

Dijkstra, E. W. (1959). *A note on two problems in connexion with graphs*. *1*, 269–271. https://doi.org/10.1007/BF01386390

Feldbrugge, J., Pen, U.-L., & Turok, N. (2023). Oscillatory path integrals for radio astronomy. *Annals of Physics*, *451*, 169255. https://doi.org/10.1016/j.aop.2023.169255

Gautschi, W. (2004). *Orthogonal polynomials: Computation and approximation*. OUP Oxford. https://doi.org/10.1093/oso/9780198506720.001.0001

Gibbs, A., Hewett, D. P., & Huybrechs, D. (2024). Numerical evaluation of oscillatory integrals via automated steepest descent contour deformation. *Journal of Computational Physics*, 112787. https://doi.org/10.1016/j.jcp.2024.112787

Hewett, D. P., Ockendon, J. R., & Smyshlyaev, V. P. (2019). Contour integral solutions of the parabolic wave equation. *Wave Motion*, *84*, 90–109. https://doi.org/10.1016/j.wavemoti.2018.09.015

Huybrechs, D., K., A., & Lejon, N. (2019). A numerical method for oscillatory integrals with coalescing saddle points. *SIAM J. Numer. Anal.*, *57*(6), 2707–2729. https://doi.org/10.1137/18M1221138

Huybrechs, D., & Vandewalle, S. (2006). On the evaluation of highly oscillatory integrals by analytic continuation. *SIAM J. Numer. Anal.*, *44*(3), 1026–1048. https://doi.org/10.1137/050636814

Kirk, N. P., Connor, J. N. L., & Hobbs, C. A. (2000). An adaptive contour code for the numerical evaluation of the oscillatory cuspoid canonical integrals and their derivatives. *Comp. Phys. Comm.*, *132*(1), 142–165. https://doi.org/10.1016/S0010-4655(00)00126-0

NIST Digital Library of Mathematical Functions. (2023). http://dlmf.nist.gov

Ockendon, J. R., Ockendon, H., Tew, R. H., Hewett, D. P., & Gibbs, A. (2024). A caustic terminating at an inflection point. *Wave Motion*, *125*, Paper No. 103257. https://doi.org/10.1016/j.wavemoti.2023.103257

Pearcey, T. (1946). The structure of an electromagnetic field in the neighbourhood of a cusp of a caustic. *Philos. Mag. (7)*, *37*, 311–317. https://doi.org/10.1080/14786444608561335

Popov, M. M. (1979). The problem of whispering gallery waves in a neighbourhood of a simple zero of the effective curvature of the boundary. *J. Sov. Math. (Now J. Math. Sci.)*, *11*, 791–797. https://doi.org/10.1007/BF01455058

Wolfram. (2024). *Mathematica NIntegrate integration rules - LevinRule*. https://reference.wolfram.com/language/tutorial/NIntegrateIntegrationRules.html#32844337