User information for the BesselXZeros package

The functions of interest in this package are cross-product combinations of the Bessel functions of the first and second kinds, $J_\nu(z)$ and $Y_\nu(z)$, and the corresponding derivatives, $J'_\nu(z)$ and $Y'_\nu(z)$. For complex order ν, complex variable z and positive real parameter λ ($\neq 1$), the Bessel cross-products have the form

$$X_\nu^{ab}(z,\lambda) = J_\nu^{(a)}(z)Y_\nu^{(b)}(\lambda z) - Y_\nu^{(a)}(z)J_\nu^{(b)}(\lambda z) \tag{1}$$

where $a, b \in \{0, 1\}$ are superscripted in brackets on the right side to indicate differentiation with respect to the argument. The cross-products often emerge as eigenfunctions in physical problems that have annular or tubular symmetry. It follows that the roots of the $X_\nu^{ab}(z,\lambda)$ are required as the eigenvalues of the problem.

The set of functions in this package implement an algorithm that reliably locates the zeros of the Bessel cross-products in the complex-z plane for complex ν. The mathematical detail for the algorithm is described in the manuscript: "Computing the zeros of cross-product combinations of the Bessel functions with complex order" by Jeff Kershaw & Takayuki Obata [1],

About the package

All of the code was written and tested in Mathematica version 12.1. The package consists of two files:

1. *BesselXZeros.m* (Mathematica code)
2. *BesselXZeros_ userinfo.pdf* (this file).

The code may be loaded into Mathematica in the usual way by placing *BesselXZeros.m* somewhere in the current search path and then executing `Get["BesselXZeros`"]`. Loading the package gives the user direct access to eight functions (in two groups of four):

- `BesselX00`, `BesselX01`, `BesselX10` & `BesselX11`.
- `BesselX00Zeros`, `BesselX01Zeros`, `BesselX10Zeros` & `BesselX11Zeros`.

Information on how to use these functions is obtained in the usual way by typing `?BesselX*`. The remaining functions in the package are for internal use only, but if necessary thay are accessible via either the context path `BesselXZeros`Private`` or by opening *BesselXZeros.m* in a notebook.

Test code

The following Mathematica code applies `BesselX11Zeros` to find the roots for given ν and λ and then plots them in the complex plane. If $|\nu| < 20$ and $\lambda < 5$ the solutions will be superimposed on a contour plot of $X_\nu^{11}(z,\lambda)$.

```
nu = 14.6 Exp[Pi/3 I]
la = 4.51
Nz = 10
rts = Quiet@BesselX11Zeros[nu, la, Nz]
Print["Number of roots along each seedline: ", Length[#] & /@ rts];
If[Abs[nu] < 20 && la < 5,
 x1 = -1; x2 = 1. + Max@Re@rts;
 y1 = Min@{1.5 Im@rts, -1}; y2 = Max@{1.5 Im@rts, 1};
 epilog = {{AbsolutePointSize[5], Red,
    Point@{Re@#, Im@#} & /@ rts[[1]]}, {AbsolutePointSize[5], Green,
    Point@{Re@#, Im@#} & /@ rts[[2]]}, {AbsolutePointSize[5], Blue,
    Point@{Re@#, Im@#} & /@ rts[[3]]}};
 cplt = ContourPlot[{Re@BesselX11[x + I y, nu, la],
    Im@BesselX11[x + I y, nu, la]}, {x, x1, x2}, {y, y1, y2},
   AspectRatio -> Automatic, PlotPoints -> 20, ContourShading -> None,
    Epilog -> epilog],
 Show[
  Graphics@{AbsolutePointSize[8], Black, Point@{Re@#, Im@#} & /@ Flatten@rts},
  Graphics@{AbsolutePointSize[6], Red, Point@{Re@#, Im@#} & /@ rts[[1]]},
  Graphics@{AbsolutePointSize[6], Blue, Point@{Re@#, Im@#} & /@ rts[[3]]},
  Graphics@{AbsolutePointSize[6], Green, Point@{Re@#, Im@#} & /@ rts[[2]]}]
]
```

Some details on the implementation for $X^{11}_\nu(z,\lambda)$

There are a number of problems that must be solved numerically to construct a reliable algorithm for the roots of the $X^{ab}_\nu(z,\lambda)$. This subsection gives a brief description of the steps and functions involved in the implementation for $X^{11}_\nu(z,\lambda)$. The equations, figures and definitions for the mathematical expressions discussed below may be found in [1]. Implementations for the remaining cross-products are similar with changes made to suit the details included in Section 3 of the same manuscript.

1. A set of functions that quickly and accurately calculate $X^{11}_\nu(z,\lambda)$ and $\mathrm{Xi}^{11}(Z,\nu,\lambda)$ are essential. The Airy and Bessel functions implemented in Mathematica were employed in this work to construct the functions `BesselX11` and `Xi11`. A reliable root-finding algorithm is also required, with Mathematica's `FindRoot` function utilised in the implementation described here.
2. The second step is to implement functions that accurately calculate $\zeta(Z)$ and its inverse. The difficulty for numerical calculations is that the definition of these functions provided by Eq. (A.5) is multivalued. Mathematica always returns the value corresponding to the principal branch when working with complex numbers, which means that the code must be designed to choose the correct branch when the principal value is not appropriate. For $\zeta(Z)$, it is straightforward to calculate the correct value for the intermediate quantity $\rho(Z)$ because it corresponds to the principal value, but a little more care is required to obtain the correct value of $\zeta(Z) = (3\rho(Z)/2)^{2/3}$ due to the cube root. The function `zi` implements $\zeta(Z)$ for Z in the upper half of the complex Z-plane, and uses the relation $\zeta(\bar{Z}) = \overline{\zeta(Z)}$ for the lower half of the plane. Calculating the inverse function is addressed by the function `Zed`. The function first calculates the intermediate value in the ρ-plane and then uses `FindRoot` to solve the transcendental equation $\rho = \sigma - \tanh\sigma$ for σ in the half-strip defined by $\mathrm{Re}\,\sigma > 0$ and $-\pi < \mathrm{Im}\,\sigma \leqslant 0$. The desired result follows immediately because $Z = \mathrm{sech}\,\sigma$. The success of `Zed` clearly hinges on a robust choice of the initial guess for `FindRoot` for all possible input values of ζ. Extensive calculations with the value $0.1 - i\pi/2$ have never found it to fail. Note also that `zi` and `Zed` both contain some additional code to account for the special case when ν is real.
3. Given `Zed`, the solutions of Eqs. (9) & (12) follow after first transforming the right side of each equation into the ζ-plane. The text in Section 2.2 describes how the correct branch is chosen after the transformation. The functions `Xi11Z0m1` & `Xi11Z0m2` perform these operations.
4. Solving Eq. (15) requires a means to calculate the inverse of $\eta(Z,\lambda)$. As seen in Fig. 8, η maps each point in the upper-right quadrant of the Z-plane to a unique point in the lower-right quadrant of the complex η-plane. Based on that knowledge, a practical means to solve the problem was to perform a direct search for the solution in the Z-plane itself. The function `Xi11Z0m3` has been written to calculate the required solutions with `FindRoot` doing most of the work again. The value $1+i$ provides a robust initial guess reflecting the expected location of the solution, but an initial value of 1 is used when ν is real.

5. The next step is to calculate M_1, M_2 and M_3. Key to solving this problem is a routine that can reliably calculate O_α, which is defined in the manuscript as the intersection point of the trajectories $L_\nu^\lambda(\pi)$ and $L_\nu^1(-\pi/3)$. The former trajectory implies that $Z(t \geqslant 0) = \lambda^{-1}\texttt{Zed}[te^{(\pi-2\alpha/3)i}]$, which can be inserted into the condition $\arg\zeta_\nu(Z) = -\pi/3$ specifying the latter trajectory. The resultant equation is solved numerically with `FindRoot` for the value of t that corresponds to O_α along $L_\nu^\lambda(\pi)$. Noting that the solutions for t are $-\zeta(\lambda)$ when $\alpha = 0$ and 0 when $\alpha = -\pi/2$, it was discovered that as α increases from $-\pi/2$ to 0 the solution for t also increases monotonically within the range $[0, -\zeta(\lambda)]$. The numerical search is therefore restricted to this range with initial values of 10^{-5} and $-\zeta(\lambda)/2$ chosen for $\alpha < -0.1$ and $-0.1 \leqslant \alpha < 0$, respectively. O_α for $0 < \alpha < \pi/2$ is the conjugate of the result for $-\pi/2 < \alpha < 0$, while the results for $\alpha = 0$ & $\pm\pi/2$ are special cases corresponding to $O_\alpha = 1$ & λ^{-1}, respectively. The code to perform these calculations is contained in `PtOalfa`. The functions `M11m1`, `M11m2` and `M11m3` use `PtOalpha` to obtain M_1, M_2 and M_3 according to the definitions presented in the text.
6. Once Eqs. (9), (12) & (15) can be solved, the perturbation series solutions of Eqs. (8), (11) & (14) can be calculated. This is done from first principles by the functions `Xi11Zm1`, `Xi11m2` & `Xi11m3`. These functions use the ability of Mathematica to perform analytical operations so that they can calculate the perturbation solutions to any order as long as it is within practical computational limits. Nevertheless, it is found that cutting the perturbation series at the first-order term is sufficient for the algorithm to produce satisfactory results.
7. The core of the implementation is the code that calculates and sorts the roots of $\mathrm{Xi}^{11}(Z, \nu, \lambda)$. Apart from the two special cases discussed in the next section, the code in `Xi11Roots` largely follows the recipe described in the manuscript: the perturbation series solutions are fed to `FindRoot` as initial guesses, after which m_0 is calculated to determine which course of action should be taken for the central root. If $m_0 = 1$, application of Mathematica's `Union` function removes the duplicated root. If $m_0 = -1$ then Eq. (19) is used to approximate the central root. Of course, if $m_0 = 0$ then no action is needed. The roots are also sorted according to which seedline the initial guess belongs to. When $m_0 = -1$ the perturbation solutions for $m_1 = M_1 + 1$, $m_2 = M_2 + 1$ and $m_3 = M_3 - 1$ are also evaluated and the central root is attributed to the seedline corresponding to the nearest of those in the complex plane. `Xi11Roots` also contains some code that checks for possible unknown faults with the algorithm, in which case the routine will print out the information that it has and then abort.
8. `BesselX11Zeros` is the front-end function that the user must call to calculate the roots. The code first uses the symmetry relations in Eq. (6) to transform the input values of α and λ into the ranges $-\pi \leqslant \alpha \leqslant 0$ and $\lambda > 1$ as described in the text. `Xi11Roots` is then called and the results are fed as initial values to `FindRoot` after multiplying by ν. After finding the roots, the code then transforms the solutions back into the correct region of the complex z-plane with the symmetry relations if necessary. Note that the program only outputs those roots that have positive real part; roots with negative real part can be obtained by multiplying the output by -1. `BesselX11Zeros` will return all roots associated with the first and second seedlines, while the number of roots associated with the third seedline is controlled by the third argument to the function.

Other issues

The implementation described above has been run for parameter values in the ranges $0.1 < |\nu| < 100$, $|\arg\nu| \leqslant \pi$ and $1 < \lambda < 10$. Within those ranges the perturbation series solutions and Eq. (19) always provide an accurate understanding of the true root distribution.

Two situations have been found where `FindRoot` fails to converge to the correct root within the `Xi11Roots` function. The following paragraphs briefly describe the checks and workarounds implemented to address those failures:

(i) Due to the complicated structure of $\mathrm{Xi}^{11}(Z, \nu, \lambda)$ around O_α, the search corresponding to the seed Z_{M_1} occasionally fails to converge to the correct root. The function `findRootXi11m1` was introduced to deal with this problem. The function first applies `FindRoot` and then checks to see if the root it has found is acceptable. The check is performed by simply testing whether the distance between the found root and the initial value is less than the distances to the adjacent seeds on either side. If not, the code drops through to a loop that incrementally moves the initial value along the first seedline towards Z_{M_1-1} until `FindRoot` produces a satisfactory result.

(ii) Along the second seedline, although it is clear from contour plots of $\mathrm{Xi}^{11}(Z, \nu, \lambda)$ that the initial guesses are very close to the true roots of the function, `FindRoot` sometimes converges to the wrong root. By recording each step that `FindRoot` takes after being given the initial value, it has been found that the problem occurs in the first few steps. It is also known that $\mathrm{Xi}^{11}(Z, \nu, \lambda)$ can have values of $\pm 10^{20}$ and more in the vicinity of the root when the problem occurs, in which case there must be very large gradients that may lead `FindRoot` astray. A method to circumvent the problem has been improvised in the form of the `findRootXi11m2` function. Similar to the procedure used to fix the problem in (i) above, the function first applies `FindRoot` and then tests whether the result is acceptable. The function takes m_0 and the roots associated with the first and third seedlines as arguments to ensure that the roots that are found are unique to the second seedline. If the result is unsatisfactory the code executes a loop that incrementally moves the initial value along the second seedline towards $Z = 1$ until the root is acceptable.

Other than for the problems just discussed, `FindRoot` always returns a result that is near to the initial guess it is given. However, it often produces a lot of warning messages that suggest the accuracy of the result is questionable. `FindRoot` has been employed using the default settings in the implementation, but it is possible that some code that adaptively adjusts the options of the function might improve the results.

References

[1] J. Kershaw, T. Obata, Computing the zeros of cross-product combinations of the Bessel functions with complex order, *Under review, June 2025.*