Collocation method for computing Mathieu functions

Stuart Brorson
Northeastern University

July 3, 2025

1 Introduction

This brief report describes an alternate method for computing the angular Mathieu functions. This method is a so-called collocation method which directly computes the function on a grid of points covering the domain. The traditional methods typically expand the Mathieu functions into Fourier series, then derive a recursion relation for the Fourier coefficients. Then the recursions are used to get the coefficients using one of two strategies:

1. A pair of continued fraction expansions are derived from the recursion relation. Then the first continued fraction is evaluated in the "forward" direction, and the other is evaluated in the "backward" direction. The Mathieu eigenvalue is varied to make the meeting point of the forward and backward continued fractions meet, usually using some sort of rootfinder. I find this method is extremely fiddly – the function derived using this method (whose root we want to find) is very featureless except in a very small domain around the root. Therefore, most rootfinding algorithms have a hard time finding the desired root. Instead, they sometimes wander away from the desired root and find a different root somewhere else. This is the basic reason the SciPy implementation of the Mathieu functions is buggy.

2. The recursion is used to create a matrix which may be decomposed into eigenvalues and eigenvectors. The eigenvalues are the Mathieu "characteristic values" and the eigenfunctions are the coefficients of the Fourier decomposition. Several authors contend that this is the right way to find the Fourier coefficients. I agree – based on my experience with attempting rootfinding on the continued fraction methods.

Constructing a Fourier sum giving the Mathieu functions has advantages, but here I present a collocation method which skips the Fourier series and computes the Mathieu functions directly.

2 The Mathieu ODEs

We are studying the angular Mathieu functions $ce(v, q)$ and $se(v, q)$ which arise as solutions of the ODE,

$$\frac{d^2S}{dv^2} + (a - 2q\cos 2v)S = 0 \tag{1}$$

These functions take take two args: v is the angular coordinate and q is the frequency parameter.

3 Collocation

In the collocation method, we use a finite difference approximation to (1). Set up a uniform grid of points over the domain $v \in [-\pi, \pi]$ with spacing h. The Mathieu function S is sampled at each point v_n as shown in Fig. 1. (We assume the solution S remains a function of continuous parameter q, i.e. $S_n = S_n(q)$.)

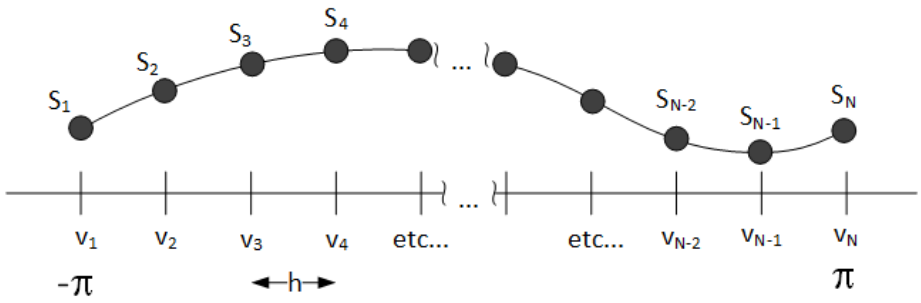

Figure 1: Sampling the function S at discrete points on the domain $v \in [-\pi, \pi]$. The result of sampling is a vector of sample point S_n which are a discretized representation of the original function $S(v)$.

The discretized equation is

$$\frac{S_{n+1} - 2S_n + S_{n-1}}{h^2} + (a - 2q\cos 2v_n)S_n = 0 \tag{2}$$

The discretized equation allows us to immediately write a matrix-vector equation. We ignore boundary conditions on the ODE for the moment. In the middle of the matrix, the discretized equation in matrix-vector form is:

$$\begin{bmatrix} \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ \dots & \frac{1}{h^2} & (\frac{-2}{h^2} + a - 2q\cos 2v_{n-1}) & \frac{1}{h^2} & 0 & 0 & \dots \\ \dots & 0 & \frac{1}{h^2} & (\frac{-2}{h^2} + a - 2q\cos 2v_n) & \frac{1}{h^2} & 0 & \dots \\ \dots & 0 & 0 & \frac{1}{h^2} & (\frac{-2}{h^2} + a - 2q\cos 2v_{n+1}) & \frac{1}{h^2} & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \end{bmatrix} \begin{pmatrix} \vdots \\ S_{n-1} \\ S_n \\ S_{n+1} \\ \vdots \end{pmatrix} = \begin{pmatrix} \vdots \\ 0 \\ 0 \\ 0 \\ \vdots \end{pmatrix}$$

Now move the coefficient a to the RHS to get

$$\begin{bmatrix} \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ \dots & \frac{1}{h^2} & (\frac{-2}{h^2} - 2q\cos 2v_{n-1}) & \frac{1}{h^2} & 0 & 0 & \dots \\ \dots & 0 & \frac{1}{h^2} & (\frac{-2}{h^2} - 2q\cos 2v_n) & \frac{1}{h^2} & 0 & \dots \\ \dots & 0 & 0 & \frac{1}{h^2} & (\frac{-2}{h^2} - 2q\cos 2v_{n+1}) & \frac{1}{h^2} & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \end{bmatrix} \begin{pmatrix} \vdots \\ S_{n-1} \\ S_n \\ S_{n+1} \\ \vdots \end{pmatrix}$$

$$= -a \begin{pmatrix} \vdots \\ S_{n-1} \\ S_n \\ S_{n+1} \\ \vdots \end{pmatrix}$$

This is an eigenvalue equation of form $AS = aS$, and the vector of S coefficients corresponds to the Mathieu function sampled at discrete points v_n. The a coefficient on the RHS corresponds to the Mathieu "characteristic values" (i.e. eigenvalues). Two remarks:

1. Performing an eigenvalue decomposition (EVD) on the LHS matrix computes the Mathieu function directly, without going through a Fourier series. However, using collocation we only get values of the function at the sample points v_n. Using the Fourier series we can get values of the function for arbitrary input v. (Of course, we could interpolate between the v_n points to get values for arbitrary v, but the accuracy of those values depends on the grid spacing h and the order of interpolation.)

2. We have not yet handled the question of boundary conditions on equation (1). We need to impose boundary conditions at the ends of the $[-\pi, \pi]$ domain in order to get the correct even and odd Mathieu functions. We'll do that in the next section.

4 Boundary conditions

There are two different boundary conditions corresponding to the two different Mathieu functions, *ce* and *se*. They are:

1. For *ce*, the function's first derivative at $v = \pm\pi$ is zero. We incorporate this information into the matrix A by using a second-order approximation to the derivative, $dS/dv \approx (S_{n+1} - S_{n-1})/2h$. This situation is depicted in Fig. 2. Assuming the left boundary lies exactly on the point v_1, the derivative condition becomes $(S_2 - S_0)/2h = 0$ or $S_0 = S_2$. We insert this into the second-derivative expression in (2) and get

$$\left(\frac{-2}{h^2} - 2q\cos 2v_1\right) S_1 + \frac{2}{h^2} S_2 = -aS_1$$

   Similarly, at the right boundary we get

$$\frac{2}{h^2} S_{N-1} + \left(\frac{-2}{h^2} - 2q\cos 2v_n\right) S_N = -aS_N$$

Therefore, the top and bottom rows of the matrix used to compute the even functions ce_n must incorporate these elements. The matrix becomes

$$\begin{bmatrix} (\frac{-2}{h^2} - 2q\cos 2v_{n-1}) & \frac{2}{h^2} & 0 & \dots & 0 & 0 & 0 \\ \frac{1}{h^2} & (\frac{-2}{h^2} - 2q\cos 2v_n) & \frac{1}{h^2} & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & \frac{1}{h^2} & (\frac{-2}{h^2} - 2q\cos 2v_{n+1}) & \frac{1}{h^2} \\ 0 & 0 & 0 & \dots & 0 & \frac{2}{h^2} & (\frac{-2}{h^2} - 2q\cos 2v_{n+1}) \end{bmatrix}$$

Note that the elements in positions $(1, 2)$ and $(N, N-1)$ have factors $2/h^2$ instead of the usual $1/h^2$ on the off-diagonals.

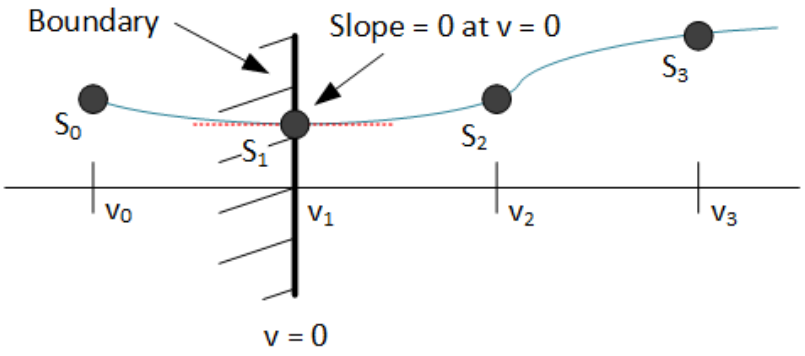

Figure 2: Grid showing position of boundary and last point on boundary.

2. For se, the function itself is zero at $v = \pm\pi$. Therefore, we set $S_1 = 0$ and $S_N = 0$ so the S vector looks like this:

$$\begin{pmatrix} 0 \\ S_2 \\ S_3 \\ \vdots \\ S_{N-2} \\ S_{N-1} \\ 0 \end{pmatrix}$$

Since the ends of the S vector are zeros, we can remove the first and last rows and columns of the $N \times N$ matrix prior to performing the EVD. That is, we perform the EVD on the reduced matrix which multiplies the vector $[S_2, S_3, \dots S_{N-2}, S_{N-1}]^T$ This is shown in Figure 3. After performing the EVD, we may take the reduced eigenvector and "glue" the value 0 back on the top and bottom of the vector. Note that we take care to compute the grid spacing h on the whole N-point domain, but perform the EVD on only the interior points.

These methods to introduce boundary conditions are implemented in the two functions "make_matrix_e" and "make_matrix_o".

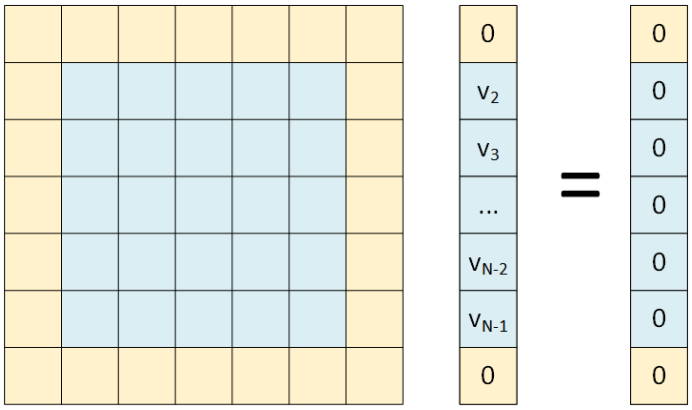

Figure 3: This depicts the eigenvalue problem at hand. Since the end values of S are fixed to zero, we perform the EVD on the interior portion of the matrix, then place zero elements on top and bottom of the S eigenvector after the EVD.

5 Post-decomposition: sorting, sign flipping, and normalization

Matlab's eigenvalue decomposition (EVD) algorithm doesn't guarantee the eigenvalues/eigenvectors are emitted in any sorted order (although in my experience they usually are). Therefore, after computing the EVD I sorted the eigenvalues first, then used that sort order to also sort the eigenvectors. The goal is to create a matrix of eigenvectors sorted by order n.

Also, eigenvectors may "point" in one of two directions (i.e. the overall sign on the eigenvector is undetermined). The Mathieu functions presented in the DLMF seem to have a definite sign convention: *ce* functions are positive at $v = 0$ and *se* functions have positive slope at $v = 0$. Therefore, I check the sign of the functions at zero and flip the sign if needed to match the DLMF.

Finally, Matlab's eigenvalue decomposition provides eigenvectors with a scaling different from that desired for the Mathieu functions. Our Mathieu functions are scaled to satisfy the normalization provided by the DLMF, eq. 28.2.30. Therefore, after computing the eigenvectors I integrated them using Matlab's trapz() (trapezoidal integration) function, found the integral, and used that value to scale the functions.

6 Results

The eigenvalues computed by the collocation method are shown in Figure 4. The famous DLMF plot is also shown for comparison. Values of *ce* with $q = 1$ are shown in Figs 5 – 6 along with the corresponding figures from the DLMF. Values of *se* for $q = 1$ are shown in Figs 7 – 8 along with the corresponding figures from the DLMF.

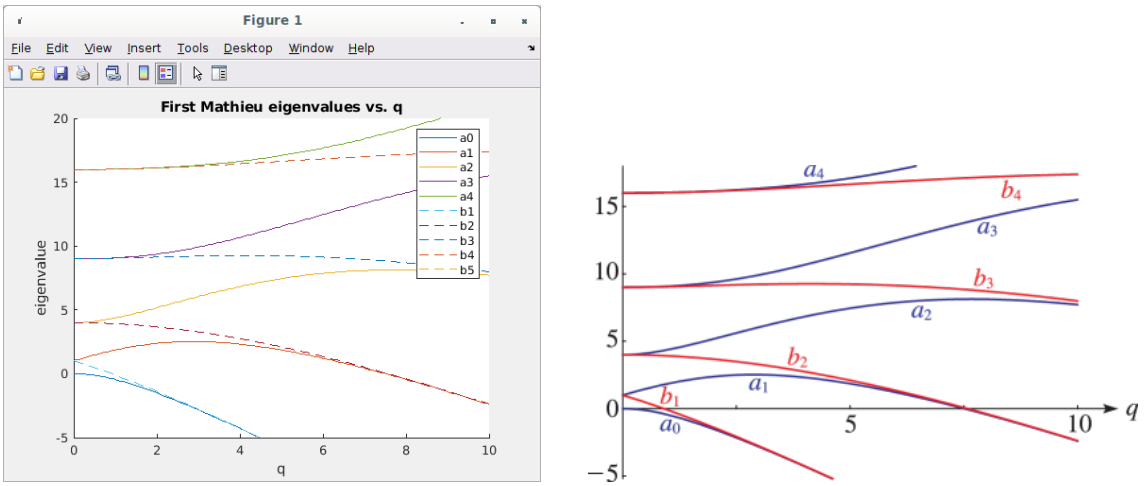

Figure 4: Left: Computed Mathieu eigenvalues a (even) and b (odd) vs. q. Right: Mathieu eigenvalues a (even) and b (odd) vs. q from the DLMF.

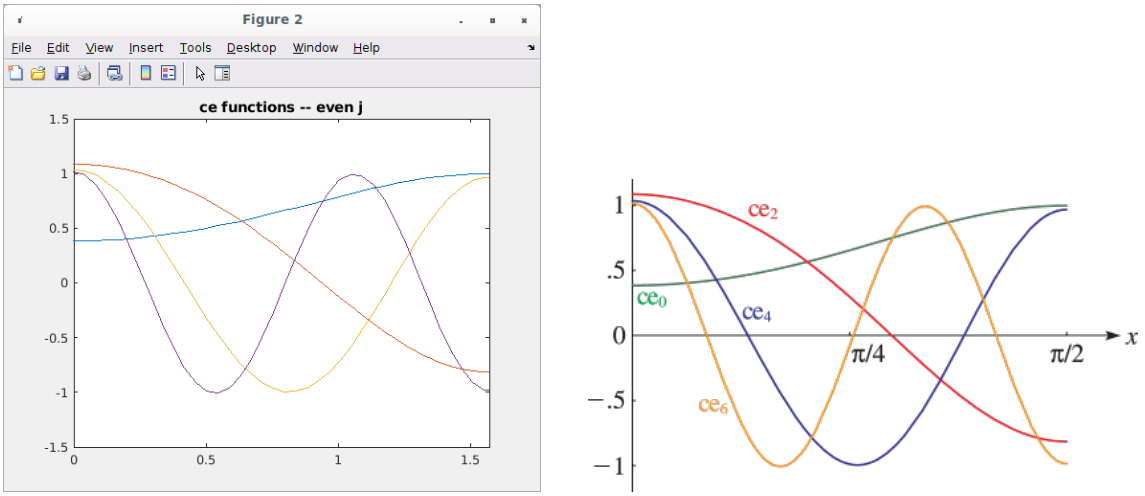

Figure 5: Left: Computed Mathieu *ce* function for $q = 1$ and even orders. Right: DLMF plot.

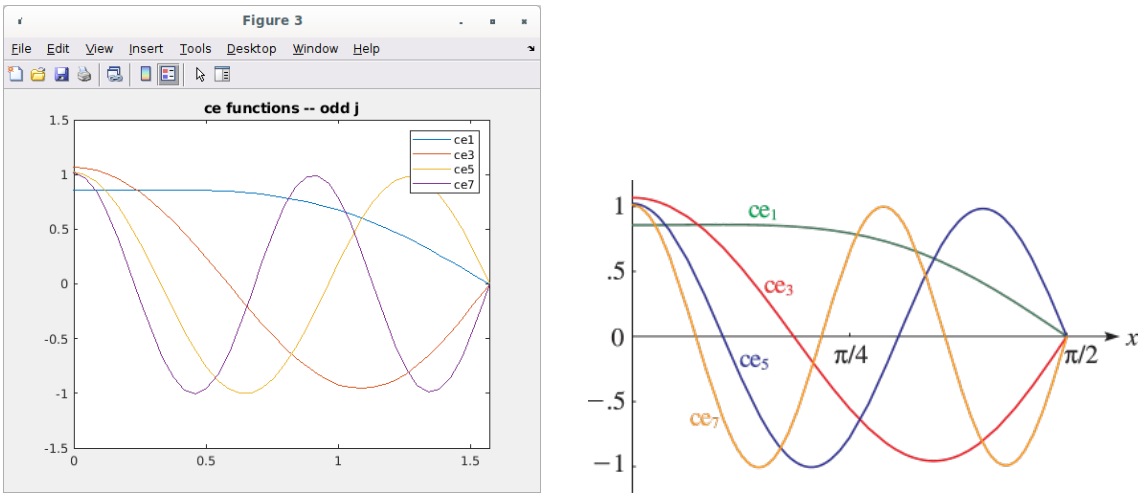

Figure 6: Left: Computed Mathieu *ce* function for $q = 1$ and odd orders. Right: DLMF plot.

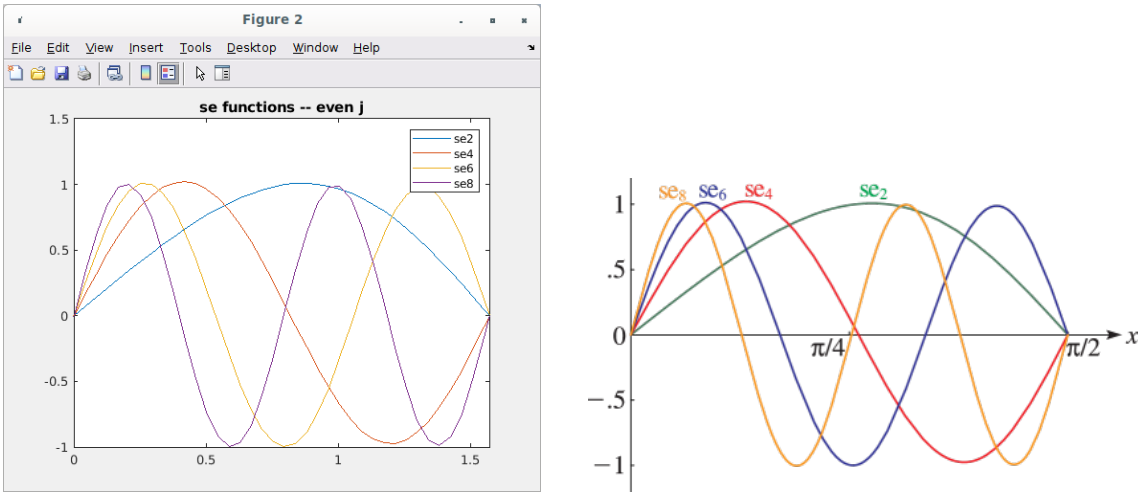

Figure 7: Left: Computed Mathieu *se* function for $q = 1$ and even orders. Right: DLMF plot.

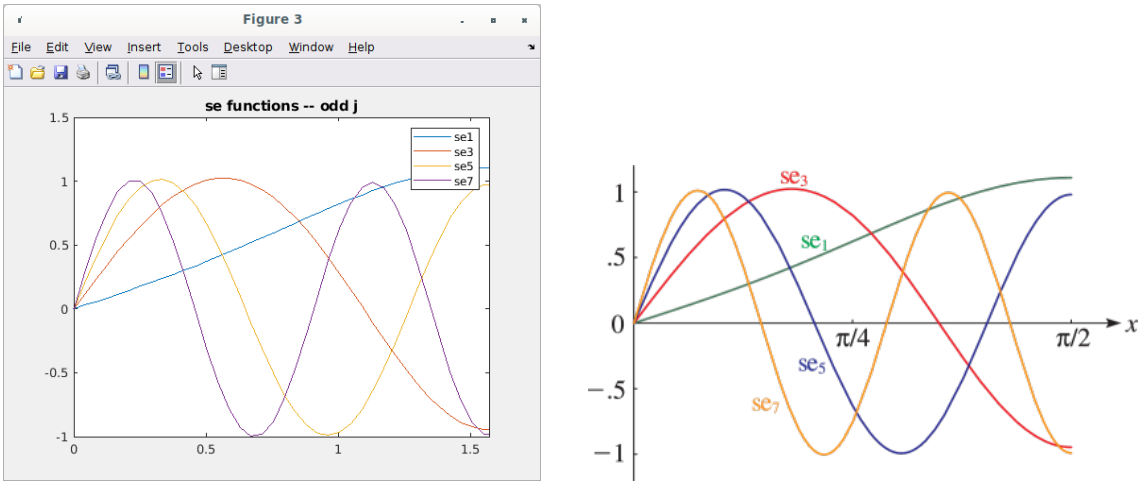

Figure 8: Left: Computed Mathieu *se* function for $q = 1$ and odd orders. Right: DLMF plot.

7 Testing

I first tested my Mathieu eigenvalue implementation by comparing my eigenvalues $a_n(q)$ and $b_n(q)$ vs. q against the series expansions given in section 28.6 of the the DLMF (https://dlmf.nist.gov/28.6#i). My values computed using collocation are presented alongside the power series expansions in Fig. 9. I was originally surprised to see the two values diverge from each other for fairly small values of q. The divergences may reflect poor behavior on the part of the power series. Table 28.6.1 of the DLMF presents the radius of convergence for each of the power series expansions. I reproduce the table in Figure 10.

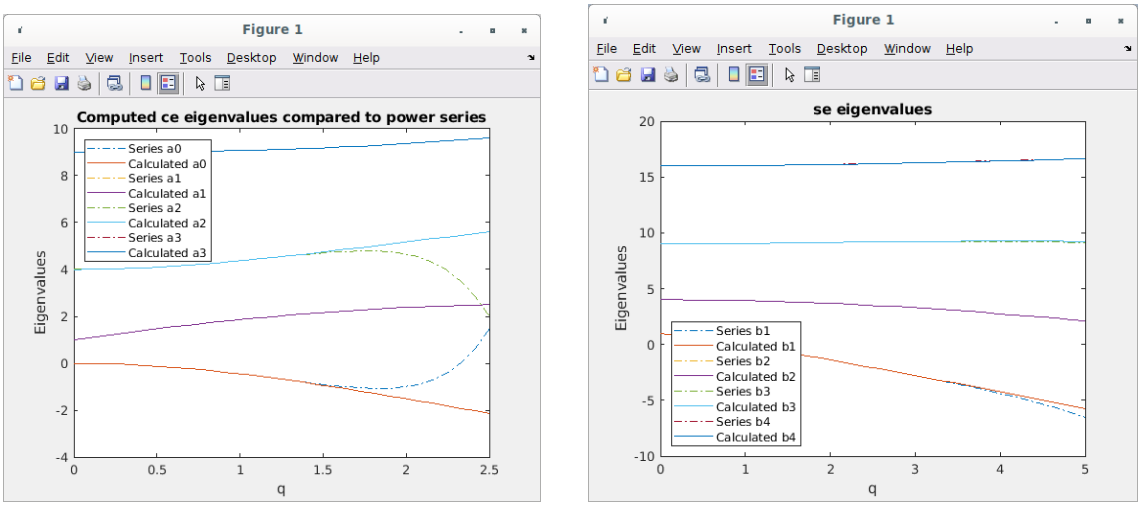

Figure 9: Left: Even (ce) eigenvalues computed using collocation vs. those computed using the DLMF series expansions for order n = 0, 1, 2, 3. Right: Odd (se) eigenvalues computed using collocation vs. those computed using the DLMF series expansions for order n = 1, 2, 3, 4.

Table 28.6.1: Radii of convergence for power-series expansions of eigenvalues of Mathieu's equation.

n	$\rho_n^{(1)}$	$\rho_n^{(2)}$	$\rho_n^{(3)}$
0 or 1	1.46876 86138	6.92895 47588	3.76995 74940
2	7.26814 68935	16.80308 98254	11.27098 52655
3	16.47116 58923	30.09677 28376	22.85524 71216
4	30.42738 20960	48.13638 18593	38.52292 50099
5	47.80596 57026	69.59879 32769	58.27413 84472
6	69.92930 51764	95.80595 67052	82.10894 36067
7	95.47527 27072	125.43541 1314	110.02736 9210
8	125.76627 89677	159.81025 4642	142.02943 1279
9	159.47921 26694	197.60667 8692	178.11513 940

Figure 10: Table 28.6.1 from the DLMF. Observe that the radius of convergence of each power series is very small, and depends upon the eig and its order n.

The table is difficult to understand, so I used it to create the table presented below, which tries to show the radius of convergence for each of the a and b eigenvalues for different orders.

Order n	ρ_a	ρ_b
0	1.46876	
1	3.76995	3.76995
2	1.46876	6.92895
3	3.76995	3.76995
4	7.26814	6.92895

The table suggests that convergence of the power series for a_0 and a_2 is guaranteed only up to around 1.47 – consistent with its behavior in the plot Figure 9. The other two eigenvalues diverge above the plotted range, so they look fine in the plot. The table also shows the power series for b_1 and b_3 converges only up to around 3.78 – also consistent with its behavior in the plot Figure 9. Again, the other two eigenvalues look fine over the range plotted since their radius of convergence is larger than the plotted range.. The limited convergence range for the power series places a strong limit on their utility to check the results of our eigenvalue implementations.

To test the functions *ce* and *se* themselves, I implemented functions which asked for the eigenvalues and the functions themselves, then computed the residual of the original ODE, equation (1). The results of this round-trip computation are shown in Figure 11. The residuals are clearly small, but non-zero.

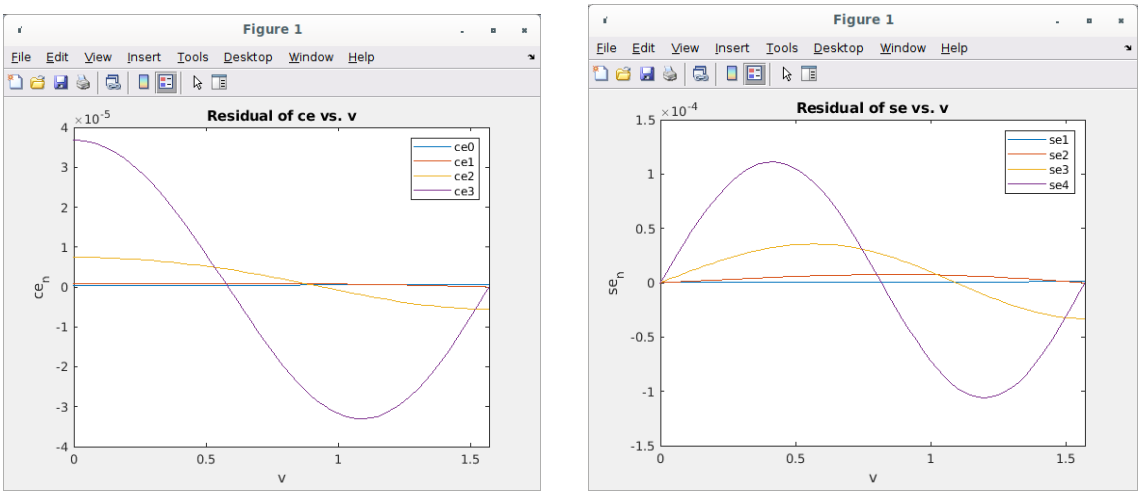

Figure 11: Left: Residual of round-trip for my implementation of $ce_n(v, q)$ for $q = 1$ and order n = 0, 1, 2, 3. Right: Residual of round-trip for my implementation of $se_n(v, q)$ for $q = 1$ and order n = 1, 2, 3, 4.