AM 129 HW 5 Deliverable

Sarosh Sopariwalla
November 20, 2020

Abstract

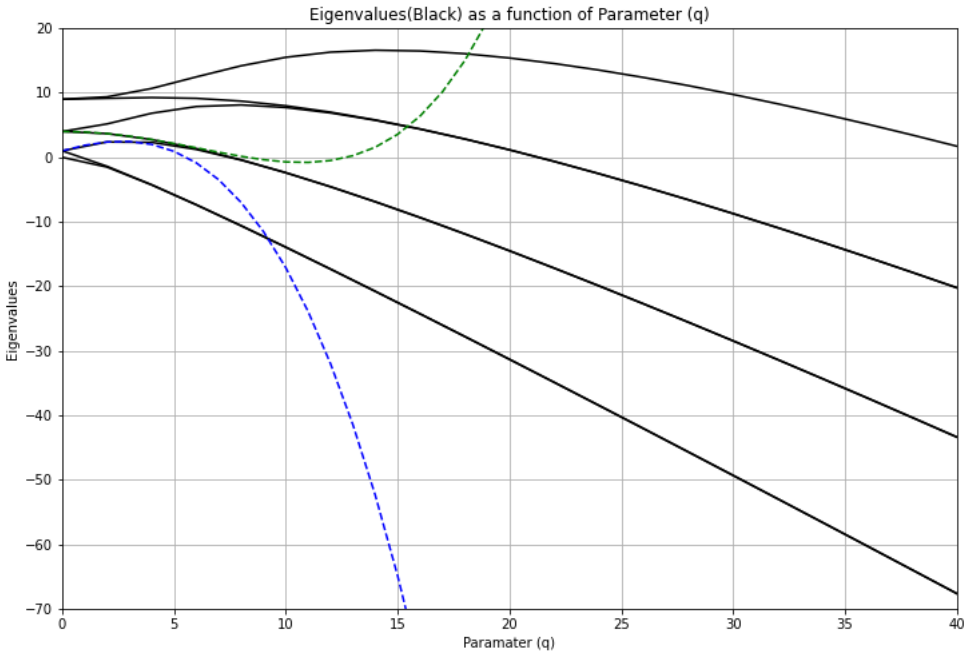
The family of solutions $y(x)$ to the ODE $\frac{d^2y}{dx^2} + (a - 2q\cos 2x)y = 0$ are called Mathieu functions. If we impose the restriction that $y(x)$ must be 2π periodic, we can rewrite our ODE as the following system:

$$\begin{cases} (2q\cos(2x) - \frac{d^2}{dx^2})y(x) = ay(x) \\ y(0) = y(2\pi) \end{cases} \tag{1}$$

While this may not look much simpler, there are now a series of steps that we can follow to solve this problem numerically. The solution to this process was implemented originally in FORTRAN, so our goal now is to write a python script that calls the Mathieu FORTRAN Functions

Numerical Solution

For the purposes of this assignment, we download all the necessary files and FORTRAN programs from the class website. These files are all compiled into a folder called mathieu that we saved into our main "code" directory. Steps one through four of this assignment simply focus on accessing the files in the mathieu sub-directory and and has us create an init file automatically through our python script. To use so many of our system processes, we take advantage of the os library that allows us to do things such as move directories, rename files, and even edit the content of our files. Finally, we put it all together and create the plot of our Eigenvalues as a function of the q parameter. We then overlayed the plot of two lines $a(q)$ and $b(q)$ whose behavior seems to mimic the eigenvalue plot for small q. The final plot can be seen below. Aside from the obviously helpful os package, we also made use of formatted printing which allowed us to use the values of certain variables as input for text in our mathieu.init file. Of course, we also made use of numpy and matplotlib but we have seen the utility of these in the last homework so it is not as interesting to see here.



q as a Hard-Coded Parameter

Since we now understand Python fairly well, it doesn't seem to be an issue that q is hard-coded into our program. After all, if we wanted to change the range of possible q values we could just edit our definitions of qRange. However, in the real world it is entirely possible that someone who doesn't understand Python may want to use this program. Thus, we should have paresweep_mathieu ask for and inputted q and then make the q-range from zero to the inputted value. Then, in the main section of the program, we just have to define $q = 42$ the same way we defined $N = 101$ and call parsweep_mathieu(N,q)

Running the Program in an Interactive Environment

I definitely think it's useful to build and run this program in an interactive Python environment. Based on my own preferences, I generally build my code in Jupyter notebooks and then download it as a .py file for the purposes of this class. When programming on Jupyter, it is easy to debug and change lines of code and see the immediate output without having to recompile the whole file. Further, in a Jupyter session we don't really need to worry about the fact that qRange is hard-coded. We can change its value and then just run two cells and see the graph changed in our notebook itself. (Jupyter is also nice in that we can partition our code so its easy to see the purpose of each section. I have left the .ipynb file in the code directory to show this.)