TA Hints: Quantum Oscillator HW 1

Add Tag Resolve Subscribe

4
vote(s)

Computational Goal: Find first 5 eigen values/functions ε_n for the second order ODE $\frac{d^2\phi_n}{dx^2} - [Kx^2 - \varepsilon_n]\phi_n = 0$ with $K = 1$, using the shooting method.

Hints:

(1) Implementing Boundary Conditions: There have been some great inputs from everyone already regarding this already, but I thought I should just explain the idea again here:

Consider the simplified second-order differential equation:

$$\frac{d^2\phi_n}{dx^2} - \beta_n\phi_n = 0$$

with same boundary conditions $\phi_n(x) \to 0$ as $x \to \pm\infty$. This system has the general solution $\phi_n(x) = c_1\exp(\sqrt{\beta_n}x) + c_2\exp(-\sqrt{\beta_n}x)$, only $\beta \geq 0$ are considered to have decaying solutions $\phi_n(x)$. Furthermore, for decaying solutions we require:

$$x \to \infty : \phi_n = c_2\exp(-\sqrt{\beta_n}x)$$

$$x \to -\infty : \phi_n = c_1\exp(\sqrt{\beta_n}x)$$

These solutions can be also expressed as the solutions to the first order equations:

$$x \to \infty : \frac{d\phi_n}{dx} + \sqrt{\beta_n}\phi_n = 0$$

$$x \to -\infty : \frac{d\phi_n}{dx} - \sqrt{\beta_n}\phi_n = 0$$

Comparing the above equations, with the constant β_n value replaced by the variable $\beta_n(x) = Kx^2 - \varepsilon_n$, a good approximation to implement the required boundary conditions $\phi_n(x) \to 0$ as $x \to \pm L$ at the boundaries of the computational domain $x \in [-L, L]$ is given by equations

$$x = L : \frac{d\phi_n}{dx} + \sqrt{Kx^2 - \varepsilon_n}\phi_n = 0$$

$$x = -L : \frac{d\phi_n}{dx} - \sqrt{Kx^2 - \varepsilon_n}\phi_n = 0$$

To get an idea of the error introduced due to the above approximation, computing the derivative of the boundary condition equations with $K = 1$ gives:

$$x = L : \frac{d^2\phi_n}{dx^2} = \frac{-x}{\sqrt{x^2-\varepsilon_n}}\phi_n + (x^2 - \varepsilon_n)\phi_n$$

$$x = -L : \frac{d^2\phi_n}{dx^2} = \frac{x}{\sqrt{x^2-\varepsilon_n}}\phi_n + (x^2 - \varepsilon_n)\phi_n$$

Hence the error introduced is $\frac{\pm x}{\sqrt{x^2-\varepsilon_n}}\phi_n$, which is small as $\phi_n(x) \to 0$ as $x \to \pm L$.

Meghana Velegar (Staff)
on Tue 16 Oct 2012 1:06:30 AM CEST

Comments

This comment has been deleted.

2

These BCs is not so good if you don't say about them explicitly in the

problem statement, because if somebody smart enough to use wiser boundary conditions then grader don't accept the answer. Evidently, the asymtotic of given equation is:

$$\frac{d^2\phi}{dx^2} - x^2\phi = 0$$

not

$$\frac{d^2\phi}{dx^2} - \varepsilon\phi = 0$$

and if you base your boundary conditions on the right asymptotic behavior $\phi \sim x^e e^{-x^2/2}$ then you get something like that: $\phi' = -x\phi$, but your solutions wouldn't be accepted. Moreover, it easy to find the exact boundary conditions because this problem could be solved analytically. Grader accepted boundary conditions mean that we supposed to find eigenvalues for a finite well with $V(x) = x^2$ if x=[-L L] and L^2 otherwise.

Mikhail Garasyov (Student)

0

Yes. Moreover we can see that boundary conditions in form $\phi' = -x\phi$ give us solutions that are closer to the theoretiacl values than grader approved solutions.

Constantin Fishkin (Student)

0

Mikhail and Constantin - your BC approximations are valid. My goals for detailing the approximation to the BCs in my first post are:

(1) The approximations leading to the BVP the Grader solved were not explicitly stated as noted above, so I wanted to state those so that everyone has the same starting point as the Grader. Since I was not able to do so before yesterday, there has already been confusion as to why "Grader approved" answers were not matching what several of you were getting as solutions. This may be the biggest reason for the discrepancies, so now maybe you can perform a quick check that your algorithm produces the same solution as the Grader when you use this approximation, as a check on the shooting/bisection part of your code. After all: the goal of this project is the implementation of the shooting/bisection method as a solution method to a BVP - not so much trying out different approximations for the BCs.

(2) As pointed out in another forum, it is important to have uniform approximations as a starting point - so that if there are errors in other parts of your algorithm, there would be a way for us to work those out. Hence there is a Grader with the "correct" answers in place as a check for the numerics that your algorithm generates as a solution. I reiterate my 3rd post: The answers recorded by the grader are "correct" for the above approximations made.

(3) As for the BCs being not "good": Not everyone taking this course will have a background in Asymptotic Analysis, in fact I did not have it either when I first took this course. So we were given the above hint as a simpler starting point to come up with approximate BCs, and we were also asked to estimate what the error would be in making the approximation. I went one step further and actually worked out the BCs and the error estimate, so there is no confusion.

We always learn numerical methods with ODEs we know an exact analytical solution to, so that we have something to compare our solutions with. I would always emphasize comparing your numerical results to theoretical values, that is an excellent point.

I/You can start another thread for further discussion of BC implementation, since this is an interesting topic in it's own right and I am sure we can learn lots from it! But please keep this thread strictly as a starting point for everyone to get a working shooting/bisection

algorithm, from which point we can play more with the various aspects of the problem. Again, the only way to check if we have a working algorithm, is comparing our results to that of the Grader.

Meghana Velegar (Staff)

Add New Comment

Time (Oldest to Newest) Time (Newest to Oldest) Votes (Most to Least)

1 vote(s)

(2) The steps for shooting method would thus be:

(i) For every eigenfunction, eigenvlaue pair $[\phi_n, \varepsilon_n]$ to be computed, start with an initial guess for ε_n and $\phi_n\ |_{-L} = y_1(-L)$.

(ii) Solve the eigenvalue problem using the MATLAB function ode45 or similar algorithm, giving values at the grid points $x_n = -L : n * \triangle x : L$ of $y_1(x_n)$.

(iii) If solution at $x = L$ satisfies the required boundary condition within required tolerance, the required eigenvalue ε_n and non-normalized eigen functions have been found. Since the ODE is linear, the normalized eigenfunction can be simply computed after convergence from the non-normalized ones by normalizing with the appropriate (in this case L^2) norm. Hence in a linear ODE the initial guess value of $\phi_n(x)$ can be arbitrary (for convenience it can be chosen to be 1).

(iv) If the solution does not meet the required boundary condition $x = L$, then the value of ε_n is adjusted in progressively smaller step sizes till the solution converges. Here, it would be useful to note that the eigenfunctions for the "odd" eigen values approach 0 at $x = L$ from positive values, and those for the "even" eigen values approach 0 at $x = L$ from negative values. This would help you to decide whether to add/subtract a correction when you overshoot/undershoot.

Hope this helps!

Meghana Velegar (Staff)
on Tue 16 Oct 2012 1:22:17 AM CEST

Add New Comment

1 vote(s)

Please ask me if you have further implementation questions. The answers recorded by the grader are "correct" for the above approximations made.

Meghana Velegar (Staff)
on Tue 16 Oct 2012 1:24:27 AM CEST

Add New Comment

1 vote(s)

Hi,

Thanks for the tips. Although I feel the results I'm getting are relatively good approximations, I am still unsure how to implement the boundary conditions, and I think I am misinterpreting your explanation. I currently perform the following steps:

1. Guess y1(-L) = A, e = 0 (I use A = 1)
2. Compute solution for x = -L:0.1:L
3. Compute err(i) = sqrt(L*L-e)*y1(L) + y2(L), check if err(i) < tol, if so then the eigenvalue is found, otherwise continue
4. If err(i+1) > err(i), de = -de / 2
5. e = e + de

This procedure does require some tweaking, since convergence depends on decent initial values for e and de, and it might not be the most efficient. The first three eigenvalues I find are correct, the last two are close but not accepted: e_4 = 6.9988, e_5 = 8.9980.

Do you have any tips on how to improve my method?

Onno Bartels
on Tue 16 Oct 2012 7:02:55 PM CEST

Add New Comment

1
vote(s)

Great work so far Onno! Here are some suggestions you can try out:

1. Guess y1(-L) = A, e = 0 (I use A = 1)

You can use the theoretical eigen values for the infinite domain as the guess, so $\varepsilon_{guess} = 2 * n - 1$ for the n^{th} eigen value. This is because we are solving a discretized approximation to the BVP on a finite domain, but the eigen values we are trying to find should be close to the theoretical values - if our approximation is any good. The bisection/shooting method will then hopefully take fewer iterations to converge.

1. Compute solution for x = -L:0.1:L

Correct! Specify $\phi(-L)$ and $\frac{d\phi}{dx}(-L)$ and solve using an ode solver of your choice

1. Compute err(i) = sqrt(L*L-e)*y1(L) + y2(L), check if err(i) < tol, if so then the eigenvalue is found, otherwise continue

Just rephrasing the above: if abs(sqrt(L*L-epsilon_n)*y1(L) + y2(L)) < tolerance, we have found the correct eigen value

1. If err(i+1) > err(i), de = -de / 2 e = e + de

Here you can make the simple changes: Starting with an initial de = 0.1, say

i f (sqrt(L*L-epsilon_n)*y1(L) + y2(L)) > 0 % overshooting for odd eigen modes, undershooting for even eigen modes

epsilon_n = epsilon_n (- if you overshoot/ + if you undershoot) de; % if I am overshooting, decrease energy level, if I am undershooting, increase energy level

de = de/2 ;

e l s e i f (sqrt(L*L-epsilon_n)*y1(L) + y2(L)) < 0 % overshooting for even eigen modes, undershooting for odd eigen modes

epsilon_n = epsilon_n + (- if you overshoot/ + if you undershoot) dE_n;

de = de/ 2 ;

end

Hope this helps!

Meghana Velegar (Staff)
on Wed 17 Oct 2012 2:31:21 AM CEST

Comments

0

Yep, adapting my bisection gave me the correct values, thanks for the tips. This method does seem to be a bit more sensitive to the initial value for the eigenvalues.

Onno Bartels

Add New Comment

Reply to Thread *Use $$LaTeX$$ for math support.*

This forum supports the Markdown markup language. Markdown has some quirks with regard to line and paragraph breaks. By default, leaving one new line (pressing enter once) does not do anything. To start a new paragraph, leave two new lines. To force a line break with only one new line, leave two or more spaces after the end of the line.

Make this post anonymous to other students.

Subscribe to this thread at the same time.

Reply to Thread

Instant Preview Render Math