Numerical solution of the quantum harmonic oscillator

Wet exercise in Quantum Mechanics 046241

Ory Schnitzer

January 27, 2013

Abstract

The stationary states of the one-dimensional quantum harmonic oscillator are determined by numerically solving the time-independent Shrodinger equation. Two methods are employed: a shooting method and a direct matrix method. The shooting method constitutes of integrating the governing equation for a series of given energies, the eigenenergies are then found by a root-finding algorithm based on the normalization constraint. In integrating the differential equation, initial conditions are obtained from the asymptotic behavior of the wave-functions in the classically forbidden regime. The second method consists of direct diagonalization of a discretized version of the Hamiltonian, thus obtaining a series of eigenvalues and eigenfunctions collectively. Both methods give excellent agreement with the exact solution. The code is attached and can be easily adapted to the solution of generic one dimensional potentials.

1 Problem formulation

The time-independent Shrodinger equation for a particle in a harmonic oscillator potential reads

$$-\frac{\hbar}{2m}\frac{d^2}{dx^2}\psi(x)+\frac{1}{2}m\omega_0^2x^2\psi(x)=E\psi(x), \tag{1}$$

where m is the particle's mass, $\psi(x)$ is the position representation wave-function, ω_0 is the angular frequency of the harmonic oscillator, and E are the energy eigenvalues. For bound states, the wave-functions $\psi(x)$ must decay as $x\pm\infty$. We seek the eigenvalues E for which such wave functions exist. We shall choose the eigenfunctions to be normalized such that the total probability is 1

$$\int_{-\infty}^{\infty}|\psi|^2\,dx=1. \tag{2}$$

1.1 Dimensionless form

It is convenient to work in dimensionless form. We thus define the scaled variables

$$x = x_{zp} s, \quad E = \hbar\omega_0 \epsilon, \tag{3}$$

where x_{zp} is the positional fluctuation in the ground level

$$x_{zp} = \sqrt{<(\Delta x)^2>|_0} = \sqrt{\frac{\hbar}{2m\omega_0}}. \tag{4}$$

While not suggested, is is convenient to define a scaled wave-function $\varphi(s)$ as

$$\varphi(s) = \sqrt{x_{zp}}\psi(x) = \left(\frac{\hbar}{2m\omega_0}\right)^{1/4} \psi(x). \tag{5}$$

With this rescaling, the normalization condition remains clean of dimensional quantities

$$\int_{-\infty}^{\infty} |\varphi|^2 \, ds = 1. \tag{6}$$

The governing equation now reads

$$\frac{d^2}{ds^2}\varphi(s) + (\epsilon - \frac{1}{4}s^2)\varphi(s) = 0. \tag{7}$$

The decay conditions are of course still valid. Note that without the rescaling there would be a nasty dimensional constant in the right-hand-side of (6). For later use we define the normalized potential

$$v(s) = \frac{s^2}{4}. \tag{8}$$

1.2 Exact solution in dimensionless form

For later comparison, we give here the exact solution in dimensionless form. The scaled eiganvalues are,

$$\epsilon_n = \frac{1}{2} + n, \quad n = 1, 2, \ldots, \tag{9}$$

while the scaled wave-functions read

$$\varphi(s) = \frac{1}{(2\pi)^{1/4}\sqrt{2^n n!}} \, e^{-\frac{s^2}{4}} \, \mathrm{H}_n\!\left(\frac{s}{\sqrt{2}}\right). \tag{10}$$

Here, H_n is the Hermit polynomial of order n.[1]

[1] When plotting the exact solution (for comparison), a ready-to-use Matlab script for calculating Hermit polynomials was downloaded from the web and used. The m-file is attached to the submission (so that the files may be executed) but the script itself is not added to this document along with the rest of the scripts since it is not mine.

2 Discretization

Following the suggestion in the exercise formulation, we seek one grid appropriate for calculating all eigenvalues up to ϵ_m.[2] We shall employ a uniform grid of N points with spacing d: $s_i, s_i + d, \ldots, s_f - d, s_f$. Since the solution is expected to rapidly decay in the classically forbidden regime ($|s| > 2\sqrt{\epsilon}$), we choose s_i, s_f beyond those points corresponding to the largest energy we seek. In the project document it is suggested to take just 2 excess points. While this is indeed sufficient in most cases, we shall more generally denote k as the number of excess points. Thus

$$s_i = -s_m - kd, \quad s_f = s_m + kd, \tag{11}$$

where $s_m = 2\sqrt{\epsilon_m}$. Note that $N = 1 + 2k + 2\frac{s_m}{d}$. The grid can be written explicitly as

$$s_j = s_i + d(j-1), \quad \varphi_j = \varphi(s = s_j); \quad j = 1 \ldots N. \tag{12}$$

We are left with choosing an appropriate value for the step size d. Of course, trial and error would work just fine. The problem with that approach is that we would have to redo the manual selection process for every different ϵ_m. Instead, we want a formula for d depending on the energy. In our one-grid approach, we require a step size that would be appropriate for the maximal energy ϵ_m (clearly, such step-size would be also appropriate for lower energies).

For this purpose, we define a 'local' De-Broglie wave length

$$\lambda = \sqrt{\frac{h^2}{2mE_m}} \tag{13}$$

where $E_m = \hbar\omega_0\epsilon_m$. A normalized version is

$$\bar{\lambda} = \frac{\lambda}{x_{zp}} = 2\pi\sqrt{\frac{\hbar\omega_0}{E_m}}. \tag{14}$$

Note that if E_m is the ground state energy, this wave length is on the order of the zero-point fluctuation, which is reasonable as the De-Broglie is the "particle wave-length". For higher energies, position fluctations should be on the order of the De-Broglie wave-length. Thus to capture changes in the wave function on that scale, we may choose

$$d = \sqrt{\frac{\hbar\omega_0}{E_m}} = \frac{1}{\sqrt{\epsilon_m}}. \tag{15}$$

[2] At least for the shooting method scheme, it would probably be better to employ different grids for different energies. That way a uniformly accurate solution would be obtained for all eigenfunctions, without employing an excessively resolved grid for the lower energy levels.

Since this qualitative approach is rather obscure, it is best to keep the code general and allow a multiplicity

$$d = \frac{\gamma}{\sqrt{\epsilon_m}} \tag{16}$$

and check what value of γ is best. Note that the dependence on ϵ_m remains.

3 Solution by the shooting method

In this approach, we integrate the differential equation for a given energy value, which is not necessarily an eigen-energy. To start off the solution, we employ the (bounded) asymptotic behavior as $s \to -\infty$. For energies which are not the eigenenergies the function should diverge at large s values. Thus an eigen-energy is found when the solution coincides with the (bounded) asymptotic behavior at $s \to \infty$.

To start of the numerical integration (using Numerov's method), we shall needs two starting values. Note that one option is to use the two values $0, a$ where a is in arbitrary constant. This would (roughly) work since the solution should approach 0, and since we are anyway normalizing the solution after integration, the value of a should not matter. The evident disadvantage of this approach is that it requires starting the grid relatively deep into the forbidden regime so that the error from picking zero for the first value is not too latge.

A far superior approach, that we will employ, is to obtain the asymptotic behavior of the wave functions at large $|s|$. Since the behavior given in the exercise documentation seems to be erroneous, we will not use it.

3.1 Asymptotic behavior of the wave function

We begin by finding the behavior of φ as $s \to \infty$. By rewriting the equation for the variable $t = 1/s$, it is readily seen that $s = \infty$ is an irregular-singular point of the differential equation. We follow the common paradigm in seeking the asymptotic behavior of differential equations around such points.[3] Writing

$$\varphi(s) = e^{T(s)} \tag{17}$$

and substituting into the governing equation, we find

$$T'' + T'^2 + \epsilon - \frac{1}{4}s^2 = 0. \tag{18}$$

Here primes represent differentiation with respect to s. Note that this equation is still exact. Now, as $s \to \infty$, clearly the fourth term dominates the third. The question is does

[3] see e.g. "*Advanced Mathematical Methods for Scientists and Engineers*" by Bender & Orszag. In the context of the Shrodinger equation, the result can be generalized for arbitrary potentials. An essentially equivalent approach is to directly employ the WKB method (not for small $\hbar$ but large $|s|$).

the first, or second term provide a consistent balance (or, perhaps both if we are unlucky). Trying to balance the fourth with the first leads to an immediate contradiction. Thus, we assume $T'' \ll T'^2$ so that

$$T'^2 \sim \frac{1}{4}s^2 \tag{19}$$

where $\sim$ denotes 'asymptotic to'. (We check for consistency later on.) Integration gives

$$T' \sim -\frac{s}{2} \tag{20}$$

where the sign appropriate for decay was chosen and the integration constant was disregarded since this is an asymptotic equality. It is evident that this result is consistent with our assumption $T'' \ll T'^2$. Integrating once more,

$$T \sim -s^2/4. \tag{21}$$

Of course, T has corrections. It is useful to look for corrections at least up to terms of T that are logarithmically large in s for a good practical approximation in these problems. Thus we write

$$T = -\frac{1}{4}s^2 + T_1 + \cdots, \quad T_1 \ll s^2. \tag{22}$$

Substituting back to the exact equation, we have

$$-sT_1' + T_1'^2 + \cdots - \frac{1}{2} + T_1'' + \cdots + \epsilon = 0 \tag{23}$$

It is easily verified that the only consistent balance here is

$$-sT_1' \sim \frac{1}{2} - \epsilon, \tag{24}$$

or

$$T_1 \sim (\epsilon - 1/2)\ln s. \tag{25}$$

We shall be satisfied with this.[4] Thus

$$\varphi \sim s^{\epsilon-\frac{1}{2}}e^{-s^2/4} \quad as \quad s \to \infty. \tag{26}$$

Since the potential is even, and since the solution has multiplicity freedom until normalization, for the integration purposes, which we begin at the other end, we can choose the behavior at the other end to be

$$\varphi \sim (-s)^{\epsilon-\frac{1}{2}}e^{-s^2/4} \quad as \quad s \to -\infty. \tag{27}$$

The constant of proportionality is set after normalizing. For an eigen-energy, and only for an eigen-energy the integrated solution at the other end should follow the behavior of the decaying solution (up to a constant) at large s.

[4]In this problem, going systematically to higher orders eventually yields the exact solution.

3.2 Integration of the differential equation with Numerov's method

The integration formula, termed Numerov's method, is given by

$$\varphi_{n+1} = \frac{\left(2 - \frac{5}{6}d^2 f_n\right)\varphi_n - \left(1 + \frac{1}{12}d^2 f_{n-1}\right)\varphi_{n-1}}{1 + \frac{d^2}{12}f_{n+1}} \tag{28}$$

where $f_n = \epsilon - s_n^2/4$. (note the different notation than in the exercise documentation.) Our starting values are

$$\varphi_1 = (-s_1)^{\epsilon - 1/2} e^{-s_1^2/4} \quad \varphi_2 = (-s_2)^{\epsilon - 1/2} e^{-s_2^2/4}. \tag{29}$$

As asked, we first integrate the equation at the specific energy values: $\epsilon = 0.5, 1.5, 2.5$ and 2. The (normalized) results are plotted in Fig.(1) (black circles). For $\epsilon = 0.5, 1.5, 2.5$, which are of course eigenenergies, the results are compared with the exact solution (dashed blue line) and the asymptotic solution (dashed red lines). For grid parameters $k = 2, \gamma = 0.5$ the results are indistinguishable. For $\epsilon = 2$, which is not an eigen-energy, the integrated function diverges. The numerical scheme can also handle very large energies. This is demonstrated in Fig. (2), where the result for $\epsilon = 65.5$ is shown; in this case more stringent grid parameters are required ($k = 10, \gamma = 0.01$).

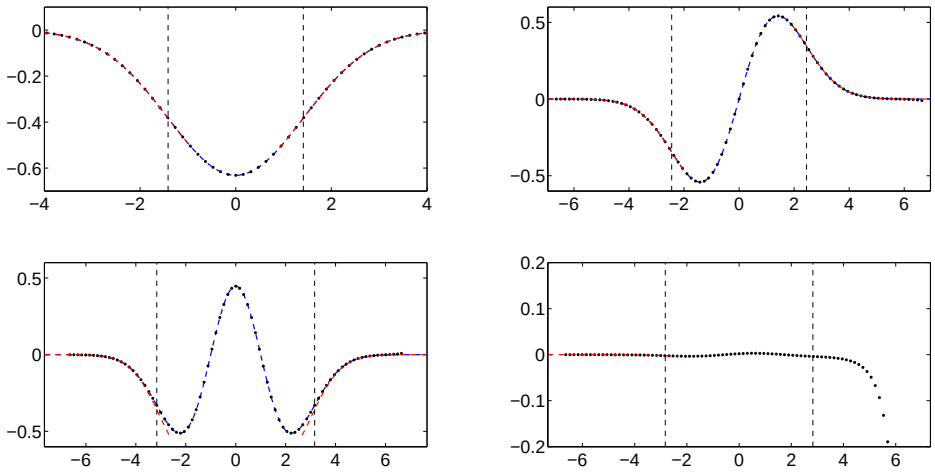

Figure 1: Numerical solution of φ as a function of s (black circles) for $\epsilon = 0.5$ (upper left), 1.5 (upper right), 2.5 (lower left), 2 (lower right). For comparison, the exact solution (dashed blue line) and the asymptotic solution (dashed red lines) are also shown. The vertical dashed lines show the limits of the classically forbidden s-region. The grid parameters are $k = 2$ and $\gamma = 0.5$.

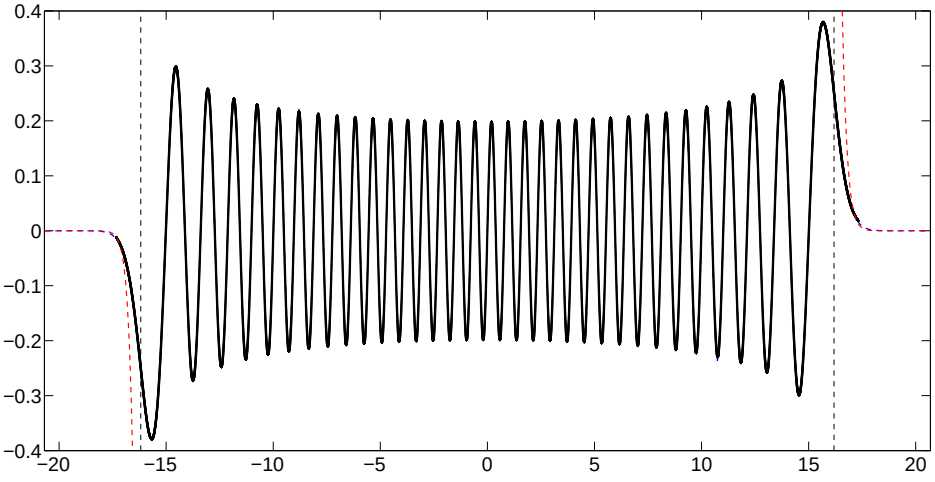

Figure 2: Numerical solution of φ as a function of s (black circles) for $\epsilon = 65.5$. For comparison, the exact solution (dashed blue line) and the asymptotic solution (dashed red lines) are also shown. The vertical dashed lines show the limits of the classically forbidden s-region. The grid parameters are $k = 10$ and $\gamma = 0.01$

3.3 Iteratively determining the eigenenergies

For a general potential, we wouldn't know in advance what energies are the eigenenergies for which bounded solutions exist. We can determine these values iteratively by integrating the equation for a series of energies up to some maximum energy and checking when the end values are consistent with the desired decaying behavior. In principle, the correct way to do this is to check when the end points are consistent with the bounded asymptotic behavior. However, for simplicity, we just seek at what energies the end points are closest to zero. The 'root-finding' method we use is extremely elementary. The codes integrates the equation for a series of energies and saves the end points. Figure 3 shows the end points for energy steps $\delta = 0.05$ for two different sets of grid parameters. Then, the code iterates through the end points looking for a sign change. A sign change means that somewhere in between these two energies an eigen-energy exists. The approximate eigen-energy is then determined by linearly interpolating between these two energy values. The results are then automatically tabulated in 'latex' format. For energy steps of $\delta(\epsilon) = 0.05$, the first nine eigenenergies, for two different sets of grid parameters, are given in table 1.

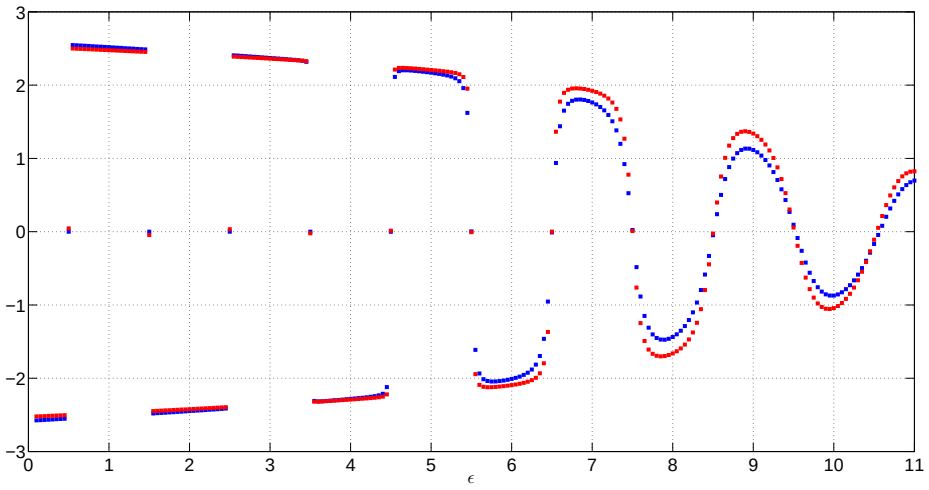

Figure 3: The endpoints of the numerically integrated $\varphi(s)$ as a function of the energy ϵ. The results for two sets of grid parameters are shown Grid A (blue): $\gamma = 0.1, k = 2$, Grid B (red): $\gamma = 0.5, k = 2$.

Exact	Shooting method	
-	Grid A	Grid B
0.500000	0.499999	0.499123
1.500000	1.500000	1.499095
2.500000	2.500002	2.499305
3.500000	3.500009	3.499528
4.500000	4.500032	4.499695
5.500000	5.500121	5.499817
6.500000	6.500528	6.499977
7.500000	7.502281	7.500582
8.500000	8.508470	8.503077

Table 1: The eigenenergies determined from the shooting method. The results for two sets of grid parameters are compared with the exact values. Grid A: $\gamma = 0.1, k = 2$, Grid B: $\gamma = 0.5, k = 2$.

4 Solution by direct diagonlization of the Hamiltionian

The advantage of this method is that an entire set of eigenenergies and eigen-functions can be found directly. We present two variants of the method which differ by the method of discretization.

4.1 Simple Matrix method

With a typical second-order discretization of the second derivative, and for a uniform grid of the type already defined,

$$-\frac{\varphi_{i+1}-2\varphi_i+\varphi_{i-1}}{d^2}+\frac{s_i^2}{4}\varphi_i=\epsilon\varphi_i. \tag{30}$$

Assuming that the beginning and end of the grid is deep enough inside the classically forbidden zone so that the boundary conditions can be taken to be zero, in algebraic vector/matrix form this can be written as

$$\left[\underline{\underline{V}}\,d^2-\left(\underline{\underline{I}}_{N,1}-2\underline{\underline{I}}_{N,0}+\underline{\underline{I}}_{N,-1}\right)\right]\cdot\underline{\varphi}=d^2\,\epsilon\,\underline{\varphi} \tag{31}$$

where $\underline{\underline{I}}_{N,k}$ is a diagonal matrix with ones on the k'th diagonal, $\underline{\varphi}$ is a vector of length N and $\underline{\underline{V}}$ is a diagonal $N\times N$ matrix $\{\underline{\underline{V}}\}_{jj}=v(s_j)$. This equation can be solved for ϵ with Matlab's 'eig' function. A more exact method would be to again use the asymptotic behavior instead of 'zero' boundary conditions. This would add some terms in the right hand side of the Matrix equation. We follow the coarser suggested solution method.

The prepared Matlab function accepts a maximum energy, generates the grid according to that energy and returns the e.values up to the maximum energy. Note that since the grid must be large enough so that the bc are met, the accuracy is not very good unless you take a grid which extends far enough (with respect to the maximal energy). The function also returns the discretized eigen-functions normalized just as in the shooting method.

4.2 Numerov Matrix method

We can instead use the Numerov formula, which is of fourth order:

$$-(\varphi_{n+1}-2\varphi_n+\varphi_{n-1})+\frac{d^2}{12}(\varphi_{n+1}v_{n+1}+10\varphi_n v_n+\varphi_{n-1}v_{n-1})=\frac{d^2}{12}\epsilon(\varphi_{n+1}+10\varphi_n+\varphi_{n-1}) \tag{32}$$

With the same approximate 'zero' boundary conditions, this can be written in Matrix form

$$-\underline{\underline{A}}\cdot\underline{\varphi}+\frac{d^2}{12}\underline{\underline{B}}\cdot\underline{\underline{V}}\cdot\underline{\varphi}=\frac{d^2}{12}\epsilon\underline{\underline{B}}\cdot\underline{\varphi}, \tag{33}$$

where

$$\underline{\underline{A}}=\underline{\underline{I}}_{N,1}-2\underline{\underline{I}}_{N,0}+\underline{\underline{I}}_{N,-1},\quad \underline{\underline{B}}=\underline{\underline{I}}_{N,1}+10\underline{\underline{I}}_{N,0}+\underline{\underline{I}}_{N,-1}. \tag{34}$$

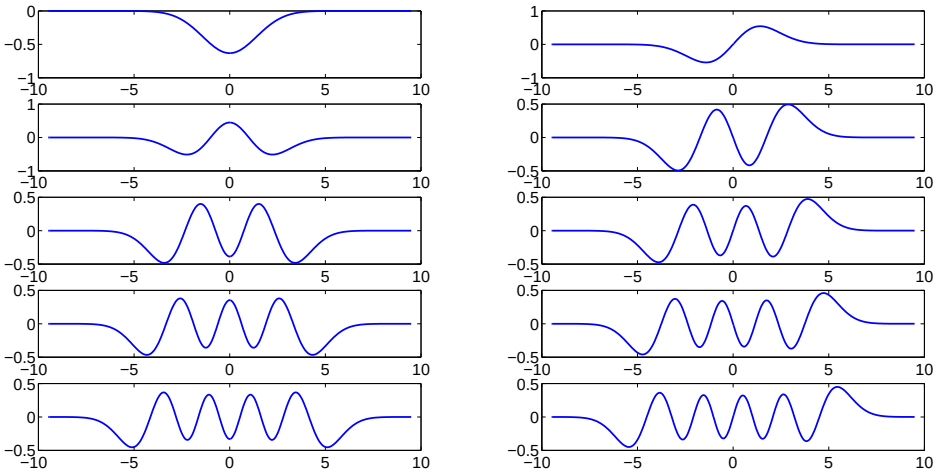

Figure 4: The first ten modes ($\varphi(s)$ against s) as obtained from the Matrix Numerov method. The modes are ordered from up to down, left to right. In this specific run the grid parameters were set to $k = 2, \gamma = 0.5$.

Multiplying by $\underline{\underline{B}}^{-1}$ from the left, we have

$$\left(-\underline{\underline{B}}^{-1} \cdot \underline{\underline{A}} + \frac{d^2}{12}\underline{\underline{V}}\right) \cdot \underline{\varphi} = \frac{d^2}{12}\epsilon\underline{\varphi} \tag{35}$$

which can be solved just as in the simple case again resulting in a set of eigenvalues and eigen-vectors.

The first ten modes, as calculated with the Matrix Numerov method, are plotted in Fig. 4; the modes obtained from the simple Matrix method cannot be visually distinguished and so not shown. The superiority of the higher-order Numerov method over the simple matrix method is best seen when comparing the eigen-values directly, see table 2. For a better comparison, the methods are compared with respect to two sets of grid parameters.

Exact	Simple Matrix method		Numerov Matrix method	
-	Grid A	Grid B	Grid A	Grid B
0.500000	0.499984	0.499609	0.500000	0.499999
1.500000	1.499922	1.498044	1.500000	1.499996
2.500000	2.499797	2.494912	2.500000	2.499985
3.500000	3.499609	3.490212	3.500000	3.499967
4.500000	4.499359	4.483982	4.500000	4.499977
5.500000	5.499047	5.476449	5.500000	5.500251
6.500000	6.498672	6.468663	6.500000	6.501906
7.500000	7.498234	7.464200	7.499999	7.508805
8.500000	8.497734	8.472112	8.499999	8.530838
9.500000	9.497171	9.509161	9.499999	9.586390

Table 2: The first ten eigenenergies calculated by directly diagonlizing the discretized Hamiltonian. The exact eigenergies are compared to the 'Simple Matrix method' and the 'Numerov Matrix method'. For each method, the results obtained with two sets of grid parameters are shown: Grid A, $k = 100, \gamma = 0.1$; Grid B, $k = 2, \gamma = 0.5$.

5 Code

5.1 Shooting method

```
function [s,Psi,norm_const] = calcPsiHarmonic(epsilon_m,epsilon,gamma,k)
% Add why returns norm_const - for plots of asymptotics

% grid options
ds=gamma/sqrt(epsilon_m);  % see notes for selection method
% prepare grid
sm=2*sqrt(epsilon_m);
si=-sm-k*ds;
N=floor(1+2*k+2*sm/ds);
s=linspace(si,-si,N);
h=s(3)-s(2); %exact actuall ds, the same as ds if epsilon_m is chosen properly
% potential
f=epsilon-(s.^2)/4;

p=zeros(1,N);
p(1)=(-si)^(epsilon-1/2)*exp(-si^2/4);
p(2)=(-s(2))^(epsilon-1/2)*exp(-(s(2))^2/4);
for jj=2:(N-1)
    p(jj+1)=((2-5*(h^2)/6*f(jj))*p(jj)-(1+h^2/12*f(jj-1))*p(jj-1))/(1+h^2/12*f(jj+1));
end

% normalization and return of Psi
norm_const=-sqrt(trapz((abs(p).^2)*h));
Psi=p/norm_const;
end

function [vec_eps,psistart,psiend] = endPoints(epsilon_i, epsilon_m, delta_epsilon,gamma,k)
vec_eps=epsilon_i:delta_epsilon:epsilon_m;

psiend=zeros(1,length(vec_eps));
psistart=zeros(1,length(vec_eps));
for ii=1:length(vec_eps)
    [s,psi,nc]=calcPsiHarmonic(epsilon_m,vec_eps(ii),gamma,k);
    psistart(ii)=psi(1);
    psiend(ii)=psi(end);
end

end
```

5.2 Matrix methods

```
function [s,eps,EV] = calcEigsHarmonic_SimpleMatrix(epsilon_m,gamma,k)

% grid options
ds=gamma/sqrt(epsilon_m)  % see notes for selection method
% prepare grid
sm=2*sqrt(epsilon_m);
si=-sm-k*ds;
N=floor(1+2*k+2*sm/ds);
s=linspace(si,-si,N);
h=s(3)-s(2) %exact actuall ds, the same as ds if epsilon_m is chosen properly


v = 1/4*diag(s.^2);
Ip = diag(ones(1,N-1),1);
I0 = diag(ones(1,N));
Im = diag(ones(1,N-1),-1);

A = v*h^2-(Ip-2*I0+Im);
[vectors,eps] = eig(A);
eps=diag(eps)/h^2;

in = find(eps<epsilon_m);
eps = eps(in); % return e.values up to epsilon_m

EV=zeros(N,length(eps)); % return e.vectors up to epsilon_m
for ii=1:length(in)
    norm_const=sqrt(trapz((abs(vectors(:,in(ii))).^2)*h));  % normalization
    if vectors(3,in(ii))>0
        norm_const=-norm_const;
```

```
    end
    EV(:,ii) = vectors(:,in(ii))/norm_const;
end

end

function [s,eps,EV] = calcEigsHarmonic_NumerovMatrix(epsilon_m,gamma,k)

% grid options
ds=gamma/sqrt(epsilon_m)  % see notes for selection method
% prepare grid
sm=2*sqrt(epsilon_m);
si=-sm-k*ds;
N=floor(1+2*k+2*sm/ds);
s=linspace(si,-si,N);
h=s(3)-s(2) %exact actuall ds, the same as ds if epsilon_m is chosen properly


v = 1/4*diag(s.^2);
Ip = diag(ones(1,N-1),1);
I0 = diag(ones(1,N));
Im = diag(ones(1,N-1),-1);

A = (Ip-2*I0+Im);
B = (Ip+10*I0+Im);
[vectors,eps] = eig(-inv(B)*A+h^2/12*v);
eps=diag(eps)/(h^2/12);


in = find(eps<epsilon_m);
eps = eps(in); % return e.values up to epsilon_m
EV=zeros(N,length(eps)); % return e.vectors up to epsilon_m
for ii=1:length(in)
    norm_const=sqrt(trapz((abs(vectors(:,in(ii))).^2)*h));  % normalization
    if vectors(3,in(ii))>0
        norm_const=-norm_const;
    end
    EV(:,ii) = vectors(:,in(ii))/norm_const;
end

end
```

5.3 Plotting and tabulation scripts

```
% Create plots by employing the shoothing method solution at a specific energy and compare to exact solution and
% asymptotic behavior

gamma=0.01;  % ratio of step size and suggested step size (suggested = 1)
k=10;        % number of steps into the forbidden regime (suggested = 2)

epsilon_m=75;
n=65;
epsilon=0.5+n;

[s,psi,normconst]=calcPsiHarmonic(epsilon_m,epsilon,gamma,k);


%% Asymptotic behavior for comparison (uses the normconst to determine constant of
% porpotionality)
forb=2*sqrt(epsilon);
s_left=linspace(-forb-4.5,-forb+0.5,300);
s_right=linspace(forb-0.5,forb+4.5,300);
asym_left=1/normconst*(-s_left).^(epsilon-1/2).*exp(-s_left.^2/4);
asym_right=(-1)^n/normconst*(s_right).^(epsilon-1/2).*exp(-s_right.^2/4);

%% Exact solution for comparison
ss = linspace(-forb-4.5,forb+4.5,600);
exact=(-1)^(n-1)*exp(-ss.^2/4).*polyval(HermitePoly(n),ss/sqrt(2))/(2*pi)^(1/4)/sqrt(2^n*factorial(n));

%% Plot
figure(1); hold on
%subplot(2,2,4); hold on
plot(ss,exact,'b--','LineWidth',1.4);
plot(s,psi,'ko','MarkerSize',2,'LineWidth',1.4)
plot(s_left,asym_left,'r--','LineWidth',1.3)
plot(s_right,asym_right,'r--','LineWidth',1.3)

% plot forbidden domain
```

```
line([-forb -forb],[-1 1],'LineStyle','--','Color','k')
line([forb forb],[-1 1],'LineStyle','--','Color','k')

axis([ss(1) ss(end) -0.4 0.4])

% Tables and plots generated from the two direct diagonalization methods
clear all; clc
epsilon_m=10;

% Good grid
gamma1=0.1;  % ratio of step size and suggested step size
k1=100;        % number of steps into the forbidden regime

% Bad grid
gamma2=0.5;
k2=2;

[Ss1,Seps1,SEV1] = calcEigsHarmonic_SimpleMatrix(epsilon_m,gamma1,k1); % simple method - good grid
[Ss2,Seps2,SEV2] = calcEigsHarmonic_SimpleMatrix(epsilon_m,gamma2,k2); % simple method - bad grid
[Ns1,Neps1,NEV1] = calcEigsHarmonic_NumerovMatrix(epsilon_m,gamma1,k1); %Numerov method - good grid
[Ns2,Neps2,NEV2] = calcEigsHarmonic_NumerovMatrix(epsilon_m,gamma2,k2); %Numerov method - bad grid

% Create tables in Latex format
for k=1:(length(Seps1))
    fprintf('%8.6f & %8.6f & %8.6f & %8.6f & %8.6f \\\\ \\hline  \n', 1/2+(k-1), Seps1(k), Seps2(k), Neps1(k),Neps2(k))
end

% Create figure with 9 first modes, plotted using the four options
figure(1); clf
for ii=1:10
subplot(5,2,ii)
%plot(Ss1,SEV1(:,ii),'k.');
hold on
%plot(Ss2,SEV2(:,ii),'r.');
plot(Ns1,NEV1(:,ii),'b');
%plot(Ns2,NEV2(:,ii),'m.');
end

% Plots for the shooting method, iterative root finding
clear all; clc

gamma1=0.1;
k1=2;

gamma2=0.5;
k2=2;

[vec_eps,starts,ends]=endPoints(0.1,12,0.05,gamma1,k1);
figure(1)
plot(vec_eps,ends,'o');

% simple root finding method based on end points only
rtA=[];
for ii=1:(length(vec_eps)-1)
    if (ends(ii+1)*ends(ii)<0)  % if sign change - interpolate to root
        a=(ends(ii+1)-ends(ii))/(vec_eps(ii+1)-vec_eps(ii));
        b=ends(ii)-a*vec_eps(ii);
        rtA=[rtA -b/a];
    end
end

[vec_eps,starts,ends]=endPoints(0.1,12,0.05,gamma2,k2);
hold on;
plot(vec_eps,ends,'or');

% simple root finding method based on end points only
rtB=[];
for ii=1:(length(vec_eps)-1)
    if (ends(ii+1)*ends(ii)<0)  % if sign change - interpolate to root
        a=(ends(ii+1)-ends(ii))/(vec_eps(ii+1)-vec_eps(ii));
        b=ends(ii)-a*vec_eps(ii);
        rtB=[rtB -b/a];
    end
end


% Create tables in Latex format
for k=1:9
```

```
    fprintf('%8.6f & %8.6f & %8.6f \\\\ \\hline  \n', 1/2+(k-1), rtA(k), rtB(k))
end
```