Semiclassical Methods: Basic Training in Condensed Matter Physics

Erich Mueller

Lecture Notes: Spring 2011

Preface

About Basic Training

Basic Training in Condensed Matter physics is a modular team taught course offered by the theorists in the Cornell Physics department. It is designed to expose our graduate students to a broad range of topics. Each module runs 2-4 weeks, and require a range of preparations. This module, "Semiclassical methods," is designed for students who have completed a standard one semester graduate quantum mechanics course.

Prior Topics

2006 Random Matrix Theory (Piet Brouwer)
Quantized Hall Effect (Chris Henley)
Disordered Systems, Computational Complexity, and Information Theory (James Sethna)
Asymptotic Methods (Veit Elser)

2007 Superfluidity in Bose and Fermi Systems (Erich Mueller)
Applications of Many-Body Theory (Tomas Arias)
Rigidity (James Sethna)
Asymptotic Analysis for Differential Equations (Veit Elser)

2008 Constrained Problems (Veit Elser)
Quantum Optics (Erich Mueller)
Quantum Antiferromagnets (Chris Henley)
Luttinger Liquids (Piet Brouwer)

2009 Continuum Theories of Crystal Defects (James Sethna)
Probes of Cold Atoms (Erich Mueller)
Competing Ferroic Orders: the Magnetoelectric Effect (Craig Fennie)
Quantum Criticality (Eun-Ah Kim)

2010 Equation of Motion Approach to Many-Body Physics (Erich Mueller)
Dynamics of Infectious Diseases (Chris Myers)
The Theory of Density Functional Theory: Electronic, Liquid, and Joint (Tomas Arias)
Nonlinear Fits to Data: Sloppiness, Differential Geometry and Algorithms (James Sethna)

2011 Practical Density Functional Theory (Tomas Arias)
Semiclassical Methods (Erich Mueller)
Ginzburg-Landau Theory of Superconductivity (Chris Henley)
Continuum Quantum Monte Carlo Methods in Chemistry and Physics (Cyrus Umrigar)

About these Notes

These are lecture notes. This is not a review article. This is not a textbook. I have stolen ideas, and borrowed arguments. I've used rigor where it appropriate, and flapped my arms where it isn't. The notes are designed to be self-contained, and I've tried to keep the number of citations down. The style is casual. I've included "group activities" which are done in class. I do this because most of my students are smarter than I am, and its good to hear what they say. I've also included "problems". Do them. They are fun.

Contents

Lecture 1

What is Classical and What is Quantum?

A. Introduction

This is a module on ”Semiclassical Physics.” What is semiclassical physics? It is studying quantum systems from a classical perspective. The example you are all familliar with is “Bohr quantization” where one produces “quantized energy levels” from classical trajectories. This module will explore a number of semiclassical methods. Here are some of the reasons you might want to learn about these tools:

1. Practical: the semiclassical tools are powerful, allowing you to calculate things which are impossible by other methods

2. Philosophical: these tools illucidate how the classical world emerges as a particular limit of the quantum world

3. Organizational: semiclassics helps one organize structures

Additionally, sometimes you have a classical understanding of a system and you want to produce a quantum description. If you know how semiclassical methods work, you have a big clue about how to quantize your theory. [In fact, in some disciplines this is the standard approach – you start with a classical Lagrangian, then you quantize it.]

Those of you who took David Rubin’s graduate quantum mechanics class saw a bit of this. Much of that was pretty formal though. Here we will really push the practical aspects of semiclassical approaches.

Group Activity: Classify the following as Classical or Quantum. Are there degrees of "quantumness"?

- photons – the particle nature of light
- atoms – the particle nature of matter
- radio waves
- Bose Einstein condensate
- Superconductor
- Fermi Liquid
- Landau Levels
- deHaas van Alphen Oscillations
- Quantum Hall Effects
- Mott Insulators
- Fock States (states of fixed particle number)
- Coherent States
- tunelling
- above barrier reflection
- Schrodinger's equation

What is classical and what is quantum depends on your point of reference. Thus there is no unique "semiclassical approach". To some people, semiclassics might mean adding a bit of particle-like character to light. To others it might mean adding a bit of wave-light character to atoms. To others it is simply a mathematical technique, which is divorced from quantum mechanics.

Typically semiclassical techniques are those which deal with how a quantum system behaves when something varies slowly in space or time.

B. The Double Well

B.1. Defining the prolem

To be concrete, lets think about the physics of a particle in a 1D double well. The double well is nice because it is simple, yet it shows some definitively quantum behaviors: quantum tunelling, and over-barrier reflection. How does one get at these from a classical perspective?

A convenient thing to look at is the "density of states" $\rho(E)$, which tells how many states exist at energy E. In quantum mechanics, the allowed E's are found by solving the time independent Schrodinger equation:

$$\left[-\frac{\hbar^2}{2m}\partial_x^2 + V_0 x^2(x^2-a^2)\right]\psi_j = E_j\psi_j, \tag{1.1}$$

and $\rho(E) = \sum_j \delta(E-E_j)$. The double well has minima at $x = \pm a/2$ and the barrier separating them has height $V_0 a^4/4$. The top of the barrier is defined to be the zero of the potential. In classical mechanics the density of states is

$$\rho_c(E) = \int \frac{dx\,dp}{2\pi\hbar}\delta\left(E - \frac{p^2}{2m} - V(x)\right), \tag{1.2}$$

where $V(x) = V_0 x^2(x^2-a^2)$. What we want to know is how are the quantum and classical densities of states related? What are the generic features of each?

Problem 1.1. Show that

$$\rho_c(E) = \sqrt{\frac{m}{\hbar^2 V_0^2 a^2}} f(E/V_0 a^4). \tag{1.3}$$

Plot the function $f(\mathcal{E})$ as a function of $\mathcal{E}$ for $-1/4 < \mathcal{E} < 1/4$.
Hint: There are several ways to do this. For those who admire special functions, I believe f can be related to elliptic integrals. For the rest of you, I would suggest just doing it numerically. [Moreover, the numerical approach is generic, while any other approach will be special.] One can numberically do the integral. More physically, a nice approach is to simply choose a grid of x and p. Find the energy on each grid point. Bin the data points and plot a histogram. Note, by symmetry it suffices to only consider one quadrant of phase space.
You should find a singularity at $E = 0$. This is known as a Van Hove singularity. It is associated with a change in the topology of the isoenergy surfaces. For $E < 0$ there are two disconnected surfaces, while for $E > 0$ there is just one.

The first step to understanding the connection is to identify a small parameter. Crudely speaking, the system behaves more classically when the derivatives in Eq. (1.1) do not contribute much.

Group Activity: Identify a dimensionless parameter which controls how "classical" the system is.

Rescaling x by a and E by V_0a^4 yields a dimensionless Schrodinger equation

$$\left[-\epsilon\partial_x^2 + x^2(x^2-1)\right]\psi = \mathcal{E}\psi. \tag{1.4}$$

The semiclassical limit is $\epsilon = \hbar^2/(2mV_0a^6) \to 0$. When ϵ is small the ground state is localized near the bottom of each well, while when ϵ is large, it is extended. Numerically finding the spectrum of this Hamiltonian is straightforward. One generic approach is to discretize space, replacing Eq. (1.4) with a matrix eigenvalue problem. One keeps the matrix finite by putting hard walls at some "large" x.

Taking a step back, the sort of problem in Eq. (1.4) is mathematically very interesting. Note that the small parameter multiplies the largest derivative. If you set $\epsilon = 0$ you do not even have a differential equation! Perturbation theory cannot work under such circumstances. How then does one do the asymptotics? The approaches to solving equations like Eq. (1.4) as $\epsilon \to 0$ are refered to as semiclassical methods.

B.2. Numerically solving the Schrodinger equation

Basics

The following Mathematica command generates the matrix we want. Its eigenvalues are approximants to the $\mathcal{E}$ in Eq. (1.4)

```
hamiltonianmatrix[maxx_?NumericQ, xstep_?NumericQ, eps_,interpolationorder_Integer: 4] :=
 Module[{grid = Table[x, {x, -maxx, maxx, xstep}], laplacian, potential, potentialfun},
  potentialfun[x_] = -x^2 (1 - x^2);
  potential = SparseArray[Band[{1, 1}] -> potentialfun /@ grid];
  laplacian = NDSolve`FiniteDifferenceDerivative[Derivative[2], grid,
    "DifferenceOrder" -> interpolationorder]["DifferentiationMatrix"];
  potential - eps laplacian]
```

I will assume that all of you have some experience with Mathematica, and know basic things like hitting "shift-enter" to run things. Let me walk through the syntax of this particular example to make sure everyone is on the same page. I'll use more Mathematica examples throughout the module, and it is good to get everyone on board. I apologize for starting with such a gritty example, but sometimes it is best to just jump in the deep end. I will also only be giving the briefest overview. If you want more, some good intros to Mathematica can be found at `http://www.chem.cornell.edu/gse1/chem3890.html`.

Mathematica can define a function in a number of ways. The most transparent is through pattern matching. One might write `f[x_]=x^2` to define an object (a function) which will square anything you feed it. The argument `x_` represents a named pattern. When you type `f[2]`, the symbol `x` gets replaced by `2` – and the computer will output `4` .

A function can be made to recognize more than one pattern. For example you could also type `f[x_,y_]=x*y`. Now the computer will recognize both `f[2]`, and `f[3,5]`, giving `4` in the first case and `15`, in the second.

Sometimes you want Mathematica to evaluate the right hand side of a function when it is defined. Other times you want to hold off on evaluating it until you actually call the function. In the first case use `=`, in the second case use `:=`.

It is good programming practice to restrict your pattern matching to only valid input. For example, here is a function which adds all the integers up to some number `g[n_Integer]:=Sum[j,{j,1,n}]`. The pattern `n_Integer` is only matched by an object whose "head" is `Integer`. [Note this is an example where you need to use the `:=`.] Thus if you type `g[9.3]` it does not get evaluated – but `g[9]` does. Caution, while `9` is classified as an integer, `9.` is not. This is a good thing, but might be confusing if you have not done much programming.

One can also restrict to patterns that pass some test. For example in `f[x_?NumericQ]=x^2`, the pattern is only matched by a number. For now you can just learn these as syntactic tricks – but there is in fact a very beautiful logic behind them.

Finally one often wants to give "default" values to a parameter. As an example `h[x_,y_:7]=x-y` defines a function which you can call with one or two arguments. If you call it with only one argument, it assigns that value to `x`, and assigns `7` to `y`. Thus one says that `y` is an "optional" argument.

With this background we now understand the first line:

```
hamiltonianmatrix[maxx_?NumericQ, xstep_?NumericQ, eps_,interpolationorder_Integer: 4] :=
```

that defines a function called `hamiltonianmatrix` which takes 4 arguments, the last of which is optional. The first two arguments must be numbers, and the last much be an integer. Physically `maxx` is going to be our cutoff: we will restrict $x \in [$`maxx`$, -$`maxx`$]$. We will discretize this compact domain, taking `xstep` to be the spacing between gridpoints. The parameter `eps` is simply the small parameter in Eq. (1.4). Finally, `interpolationorder` is a parameter which controls how the differential equation is discretized. Most of the time I would leave it at 4: if you want to use a more primitive "nearest neighbor" finite differences, then set it to 1.

Often in programming it is necessary to introduce "scratch" variables – ie. things which will be used for intermediate calculations, but which are not the final result. A good programming practice is to use "local" variables for these objects. This means that the variables are isolated from the rest of the program. In Mathematica, the way to generate local variables is with the `Module` construct. To encapsulate some code that uses four local variables, `grid`, `laplacian`, `potential`, and `potentialfun`, one would write

```
 Module[{grid = Table[x, {x, -maxx, maxx, xstep}], laplacian, potential, potentialfun},
 ..... Code that does stuff ...
]
```

Actually this code does one extra thing, it also populates the variable `grid`, with a list of numbers corresponding to our spatial grid.

The rest of the routine is straightforward. We define the potential function with

```
  potentialfun[x_] = -x^2 (1 - x^2);
```

We then create a diagonal matrix whose entries are the potential evaluated at each point with

```
  potential = SparseArray[Band[{1, 1}] -> potentialfun /@ grid];
```

The syntax `/@` is kind of cool. To see what it does, type the following: `f[x_]=x^2` then `f/@{1,2,3,4,5}`.

Note that the matrix formed by this command is stored as a "sparse array". This means that only the non-zero elements are stored in memory. There are efficient algorithms for doing linear algebra with such sparse matrices.

To create a matrix which is a discrete approximation of the Lapplacian, we use the expression

```
  laplacian = NDSolve`FiniteDifferenceDerivative[Derivative[2], grid,
    "DifferenceOrder" -> interpolationorder]["DifferentiationMatrix"];
```

Rather than going into detailed descriptions of what this command does, I will refer you to Mathematica's documentation at: `http://reference.wolfram.com/mathematica/tutorial/NDSolvePDE.html`.

Finding the Eigenvalues

We can now make a function which generates the matrix for a fixed grid. For example

```
h1[eps_] = hamiltonianmatrix[3, 0.001, eps]
```

You should get as output

```
SparseArray[<30009>,{6001,6001}]
```

which means that you have generated a 6001×6001 matrix, with 30009 non-zero entries. These are banded near the diagonal. This is a bit of overkill. We don't really need such a fine mesh.

We can find the 50 lowest eigenvalues for $\epsilon = 0.0001$ by writing

```
low0001 = Eigenvalues[h1[0.0001], 50, Method -> {"Arnoldi", "Shift" -> -1/4}]
```

The $-1/4$ is there because the minimum of the potential is at $-1/4$. If you instead wanted the 10 eigenvalues closest to the top of the barrier you would write

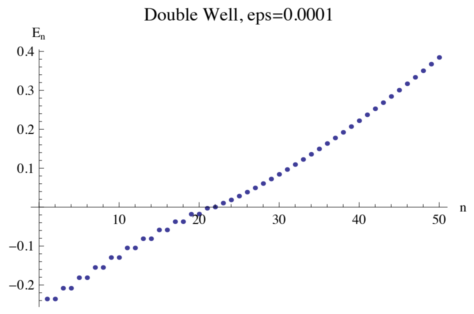
```
top0001 = Eigenvalues[h1[0.0001], 10, Method -> {"Arnoldi", "Shift" -> 0}]
```

The `"Shift" -> 0` means to find the eigenvalues closest to 0.

Structure of the Eigenvalues

Below is a typical picture of the energy eigenvalues – created by writing

```
p1 = ListPlot[Sort@e2, AxesLabel -> {"n", Subscript["E", "n"]},
  PlotLabel -> "Double Well, eps=0.0001"]
```



Group Activity: What features do you see in this graph?

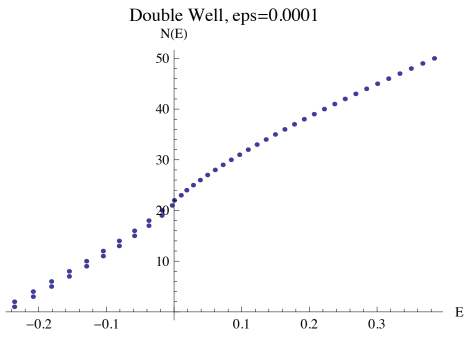
The group activity should have revealed a few things. First, all states below the barrier top come in pairs. This we all know is due to having two minima: those nearly degenerate states are split by the tunneling between the wells. Second, if you squint your eyes you should see that a smooth curve could be fit through this data. What is that curve?

A good way to think about things is to introduce a quantity $N(E)$, which is the number of states with energy less than or equal to E,

$$N(E) = \int_{-\infty}^{E} \rho(E) dE. \tag{1.5}$$

The plot we have made is nothing but E as a function of $N(E)$. In other words, if we turned the graph sideways it would be a plot of $N(E)$.

```
p2 = ListPlot[Transpose[{Sort@e2, Range[Length[e2]]}],
  AxesLabel -> {"E", "N(E)"}, PlotLabel -> "Double Well, eps=0.0001"]
```



Well, what does the density of states look like? To find that out, we need to take some sort of derivative of our discrete data. The crudest thing to plot is the set of points:

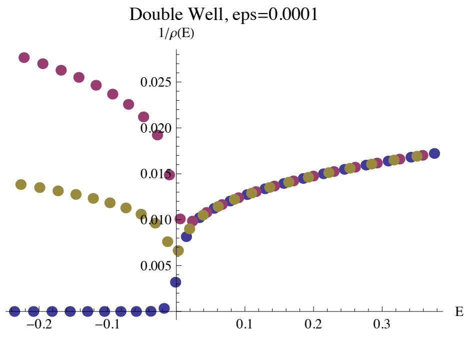
$$\left(\bar{E}_j, 1/\bar{\rho}\right) \equiv \left((E_{j+1}+E_j)/2, E_{j+1}-E_j\right), \tag{1.6}$$

the logic being that $E_{j+1}-E_j \approx \partial E/\partial N \equiv 1/\rho$. Given that the data comes in pairs, one might also want to just take every other data point, taking

$$\left(\tilde{E}_j, 1/\tilde{\rho}\right) \equiv \left((E_{j+1}+E_{j-1})/2, E_{j+1}-E_{j-1}\right), \tag{1.7}$$

and only including even j's. The resulting graph look like

```
p3 = ListPlot[
  {evenave2 = {(#1 + #2)/2, (#2 - #1)} & @@@ Partition[Sort@e2, 2],
   oddave2 = {(#1 + #2)/2, (#2 - #1)} & @@@ Partition[Rest[Sort@e2], 2],
   ave4pt2 = {Mean[#], (Max[#] - Min[#])/2} & /@ Partition[Sort@e2, 3, 2]},
   PlotStyle -> PointSize[Large],
  AxesLabel -> {"E", "1/\[Rho](E)"},
  PlotLabel -> "Double Well, eps=0.0001"]
```



The Blue and Red points represent the 2-point averages (even or odd). The gold points are the 3-point averages (only even). The Gold points $\left(\tilde{E}_j, 1/\tilde{\rho}\right)$ essentially follow the classical prediction for the density of states. Thus there are only two things missing in the classical model: the discreteness of the points, and the structures whereby the points appear in pairs.

C. Outline

We will develop the tools of semiclassics. These will include

- WKB (review)
  - formal approach to directly attacking singular ODE's.
- Path Integrals
  - cleanest way to connect the classical and the quantum
  - relates Hamilton-Jacobi equations to Schrodinger equation
  - tunneling in this language $\rightarrow$ instantons
  - forced to analytically continue into complex plane
- Semiclassics in time (adiabatic theorem)
  - Landau-Zener
  - Berry Phase
- Applications – ex. de Haas van Alphen effect, Thomas-Fermi, random matrix theory...
- Semiclassics in many-body physics
- Other modern topics

D. Final Thoughts

What is so special about the double well? Do we really care so much about it? Clearly the answer is a qualified no. The double well is the best example for illustrating some of the features of semiclassics. I always find it easier to generalize a specific example than to make concrete an abstract story.

E. Final Exercise

Problem 1.2. Numerically calculate the splitting between the lowest two levels, $\Delta = \mathcal{E}_2 - \mathcal{E}_1$, in the double oscillator as a function of ϵ. Plot $\log(\Delta)$ as a function of $\epsilon^{1/2}$.