AM205: Assignment 5 solutions*

Problem 1 – Function minimization

The minimization algorithms are tested on the Rosenbrock function,

$$f(x,y) = 100(y-x^2)^2 + (1-x)^2. \tag{1}$$

Part (a) – Steepest descent

The gradient of the function $f(x,y)$ defined in Eq. 1 is

$$\nabla f = \begin{pmatrix} -400(y-x^2)x - 2(1-x) \\ 200(y-x^2) \end{pmatrix}, \tag{2}$$

and the direction of steepest descent is

$$s_k = -\nabla f. \tag{3}$$

We then perform a line search to compute the magnitude of the local descent. It is equivalent of solving a constrained optimization problem. Instead of searching over a two-dimensional domain, we only need to search for the minimum on a line, which is a much simpler computational task. The linear constraint can be written as

$$A_1 x + A_2 y = b, \tag{4}$$

and by considering Eq. 3 we know that

$$A_1 = \frac{\partial f}{\partial y}, \qquad A_2 = \frac{\partial f}{\partial x}. \tag{5}$$

We now solve for b. We have

$$b = A_1 x_k + A_2 y_k = \frac{\partial f}{\partial y} x_k + \frac{\partial f}{\partial x} y_k. \tag{6}$$

We then use the Matlab function `fmincon` to solve the one-dimensional linear search. The Matlab code is as follows.

```
{[}x\_min,f\_min,exitflag,output{]}=...

fmincon(@rosenbrock\_function,x\_k,{[}{]},{[}{]},A,b,{[}{]},{[}{]},{[}{]},options)
```

*Solutions to problems 1 and 3 were written by Kevin Chen (TF, Fall 2014). Solutions to problem 2 was written by Chris H. Rycroft. Edited by Chris H. Rycroft.

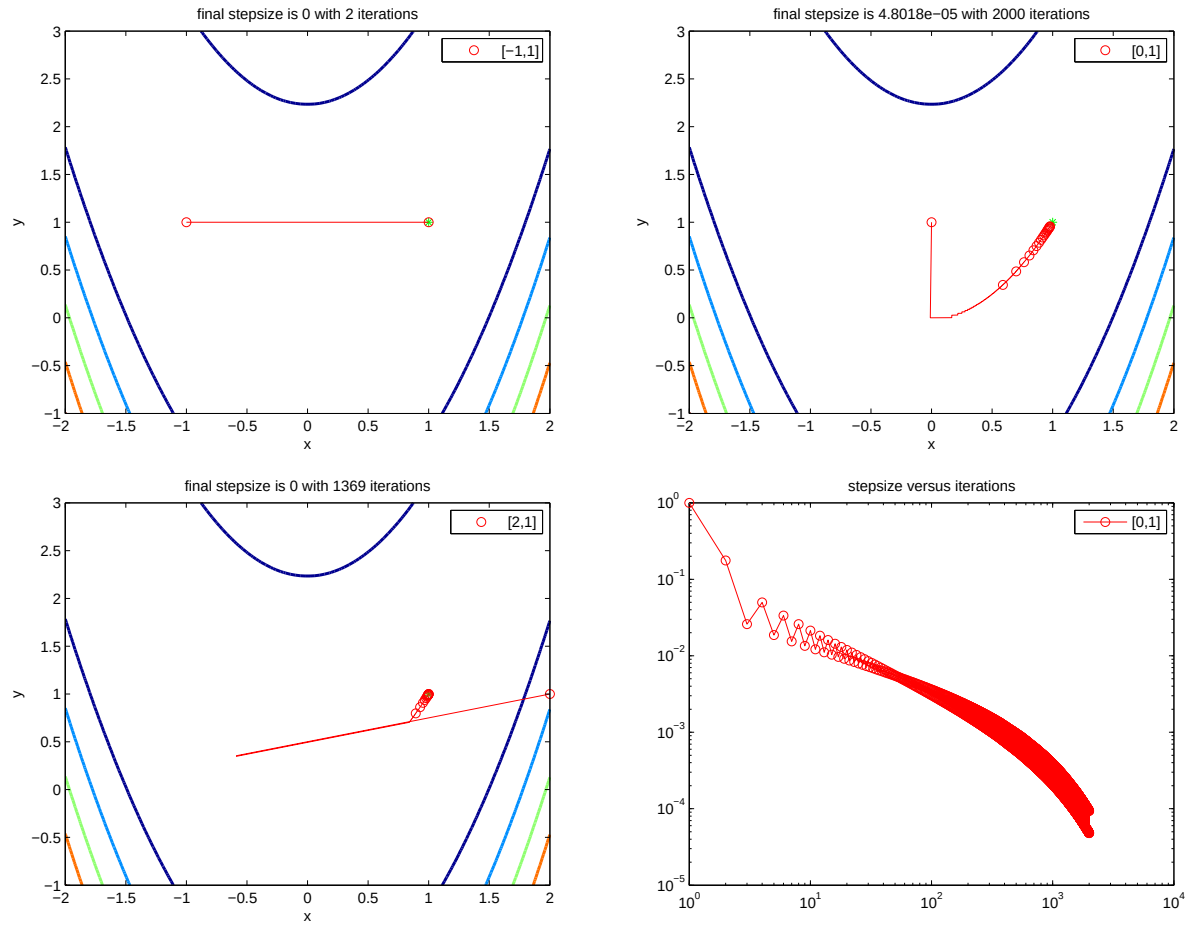


Figure 1: Plots showing the progress of the steepest descent method for finding the minimum of the Rosenbrock function, starting from (top left) $(-1, 1)$, (top right) $(0, 1)$, and (bottom left) $(2, 1)$. In the bottom right plot the step size versus the number of iterations is shown using a log–log scale, for the case of the $(0, 1)$ starting point.

Details can be found in the code `problem1aDriver`. This process is repeated until the change of consecutive steps is smaller than the tolerance. Figure 1 shows the iterations from three different starting positions. The program is terminated if number of iterations exceeds 2000. For the case of most iteration numbers, we plot the step size versus iterations on a log–log plot to show convergence. The number of iterations we obtain is

$$
\begin{aligned}
[-1, 1]^{\mathsf{T}} &\rightarrow 2 \text{ iterations}, \\
[0, 1]^{\mathsf{T}} &\rightarrow \text{more than } 2000 \text{ iterations}, \\
[2, 1]^{\mathsf{T}} &\rightarrow 1321 \text{ iterations}.
\end{aligned}
$$

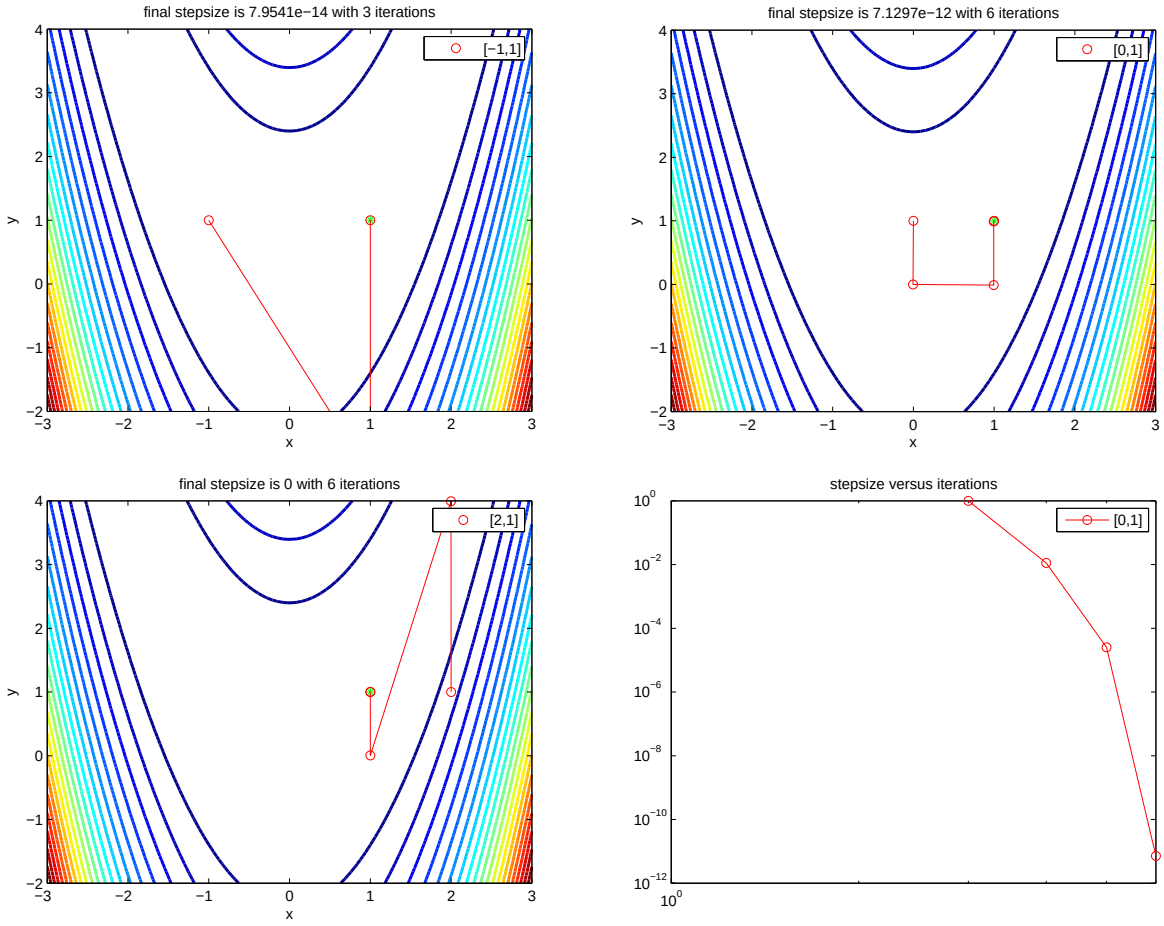


Figure 2: Plots showing the progress of Newton's method for finding the minimum of the Rosenbrock function, starting from (top left) $(-1, 1)$, (top right) $(0, 1)$, and (bottom left) $(2, 1)$. In the bottom right plot the step size versus the number of iterations is shown using a log–log scale, for the case of the $(0, 1)$ starting point.

Part (b) – Newton's method

We repeat part (a) with Newton's method. To implement Newton's method, we compute the second derivative (or Hessian matrix) as

$$H = \begin{pmatrix} -400(y - x^2) + 800x^2 + 2 & -400x \\ -400x & 200 \end{pmatrix}. \tag{7}$$

The algorithm is described in the lecture notes, and the key steps inside the loop are

$$H_f(x_k)s_k = -\nabla f(x_k), \tag{8}$$

$$x_{k+1} = x_k + s_k. \tag{9}$$

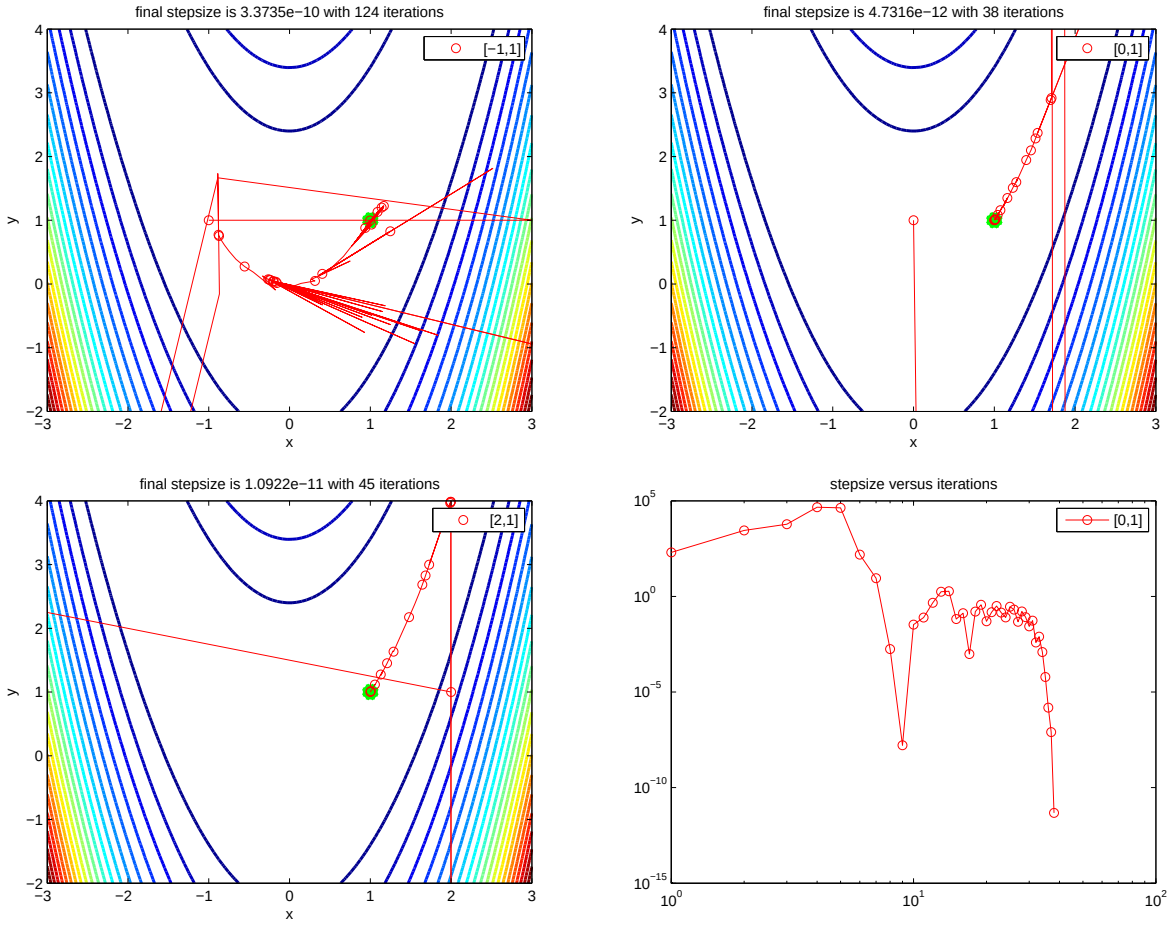


Figure 3: Plots showing the progress of the BFGS method for finding the minimum of the Rosenbrock function, starting from (top left) $(-1, 1)$, (top right) $(0, 1)$, and (bottom left) $(2, 1)$. In bottom right plot, the step size versus the number of iterations is shown using a log–log scale, for the case of the $(0, 1)$ starting point.

Figure 2 shows the result. In general, this algorithm will converge to local extrema, but not necessarily minima. However, in this problem there is only one local minimum at $(1, 1)$, so we do not need to check for alternative behavior. The number of iterations we obtain is

$$\begin{aligned} [-1, 1]^\mathsf{T} &\to 3 \text{ iterations}, \\ [0, 1]^\mathsf{T} &\to 6 \text{ iterations}, \\ [2, 1]^\mathsf{T} &\to 6 \text{ iterations}. \end{aligned}$$

Part (c) – BFGS method

The BFGS method converges faster than steepest descent but is slower than Newton's method. Instead of computing the Hessian matrix by hand, we construct an estimate B of

the Hessian, which is continually updated throughout the computation. In this question, we initialize the B matrix as an identity matrix. The algorithm is detailed in lecture notes. Figure 3 shows the path of convergence for three different initial conditions. Note that the convergence rate is faster than steepest descent but slower than Newton's method, also the step size is larger at early time. The number of iterations we obtain is

$$\begin{aligned} [-1,1]^\mathsf{T} &\to 124 \text{ iterations}, \\ [0,1]^\mathsf{T} &\to 38 \text{ iterations}, \\ [2,1]^\mathsf{T} &\to 45 \text{ iterations}. \end{aligned} \tag{10}$$

Problem 2 – an inextensible jump rope

Part (a)

The rope's vertical position as a function of x can be parameterized as

$$y(x) = \sum_{k=1}^{n} b_k \sin \frac{\pi k x}{L} \tag{11}$$

where $b = (b_1, b_2, \ldots, b_n)$ are free parameters. The derivative is

$$\frac{dy}{dx} = \frac{\pi}{L} \sum_{k=1}^{n} k b_k \cos \frac{\pi k x}{L}. \tag{12}$$

For the subsequent calculations it is useful to define

$$g(x) = \sqrt{1 + \left(\frac{dy}{dx}\right)^2} \tag{13}$$

representing the total length of rope over an interval from x to $x + dx$. The aim is to maximize the kinetic energy

$$T(\mathbf{b}) = \int_0^L \rho g \omega^2 y^2 \, dx \tag{14}$$

subject to the constraint that the length of the rope is R, which can be written as

$$S(\mathbf{b}) = \int_0^L g \, dx = R. \tag{15}$$

This constrained optimization problem can be solved by introducing a Lagrange multiplier λ and considering

$$\begin{aligned} \mathcal{L}(\lambda, \mathbf{b}) &= T(\mathbf{b}) + \lambda \left(R - S(\mathbf{b})\right) \\ &= \lambda R + \int_0^L g \left(\rho \omega^2 y^2 - \lambda\right) dx. \end{aligned} \tag{16}$$

The partial derivative of $\mathcal{L}$ with respect to λ is

$$\frac{\partial \mathcal{L}}{\partial \lambda} = R - \int_0^L g \, dx \tag{17}$$

and the partial derivative of $\mathcal{L}$ with respect to b_j is

$$\frac{\partial \mathcal{L}}{\partial b_j} = \int_0^L g \left(2\rho\omega^2 y \sin \frac{\pi j x}{L} \right) dx + \frac{\pi}{L} \int_0^L \frac{j y' (\rho\omega^2 y^2 - \lambda)}{g} \cos \frac{\pi j x}{L} \, dx. \tag{18}$$

Parts (b) and (c)

The code `jump_rope_inex.py` finds the extremal point of $\mathcal{L}$ by using the Levenberg–Marquardt algorithm to set all components of $\nabla \mathcal{L}$ to zero. It uses $n = 20$, $R = 3$, and $\omega = L = \rho = 1$. It evaluates the integrals in Eqs. 17 and 18 using the composite trapezoid rule with 251 control points. Figure 4(a) shows a plot of the initial guess when $b_1 = 1.3$, and the optimized solution. Figure 4(b) shows a plot of the initial guess when $b_2 = 0.7$, and the optimized solution.

Problem 3 – quantum eigenmodes

In non-dimensionalized form the Schrödinger equation is

$$-\frac{\partial^2 \Psi(x)}{\partial x^2} + v(x)\Psi(x) = E\Psi(x). \tag{19}$$

Part (a) – eigenvalues and eigenmodes

We first want to turn the equation into an eigenvalue problem in the form of

$$A\Psi = E\Psi, \tag{20}$$

where A is an operator matrix, Ψ is a vector and E is a scalar. We discretize the Schrödinger equation on a finite domain $[-12, 12]$ using $n = 1921$ grid points. We use a finite-difference approach, and hence the second-order accurate second-order differentiation matrix is given by

$$D_2 = \frac{1}{h^2} \begin{pmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & 1 & -2 & \ddots & \\ & & \ddots & \ddots & 1 \\ & & & 1 & -2 \end{pmatrix}, \tag{21}$$

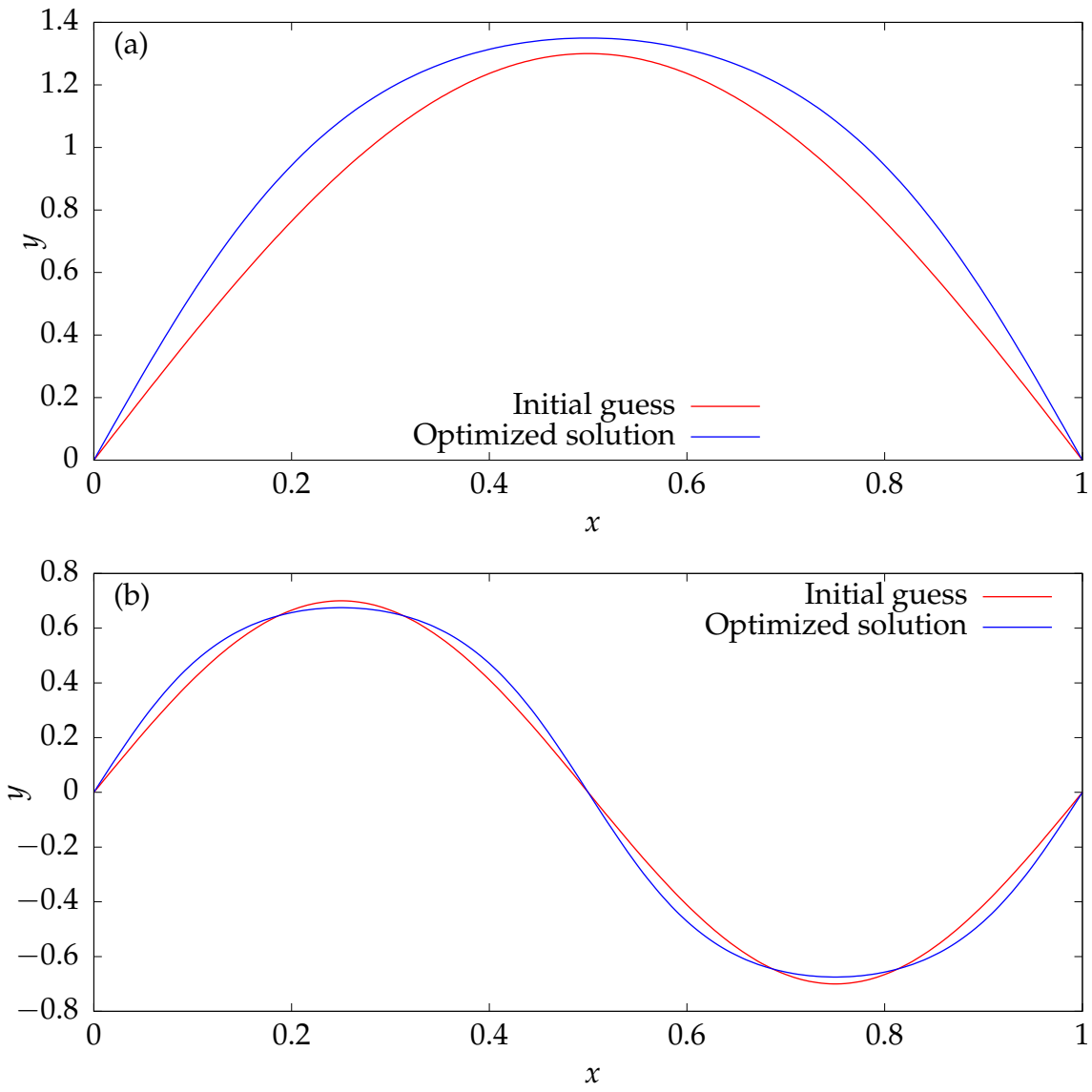
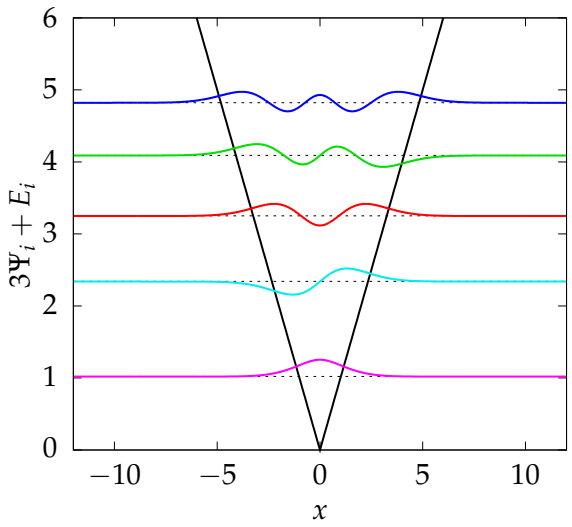


Figure 4: (a) Initial guess of $b_1 = 1.3$ and optimized solution of the steady-state shape of an incompressible jump rope. (b) Initial guess of $b_2 = 0.7$ and optimized solution of the steady-state shape of an incompressible jump rope.



E_5	4.821710730858221
E_4	4.0893267226561525
E_3	3.2492942302664067
E_2	2.338905797822476
E_1	1.0191290231979593

Figure 5: Five wavefunctions Ψ_i and corresponding energy levels E_i for the potential $v(x) = |x|$.

where $h = 24/(n-1)$. Since we have Dirichlet boundary conditions and $\Psi(-12) = \Psi(12) = 0$, we do not need to modify the differentiation matrix at the end points. The function $v(x)$ needs to be evaluated at grid points, and therefore needs to be added to the diagonals of the sparse matrix. The operator A can be written as

$$A = -\frac{1}{h^2}\begin{pmatrix} -2 & 1 & & \\ 1 & -2 & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & -2 \end{pmatrix} + \begin{pmatrix} v(x_l) & & & \\ & v(x_l+h) & & \\ & & \ddots & \\ & & & v(x_r) \end{pmatrix}. \tag{22}$$

We then use the Python function `eigs(A,5,'SM')` to find the five smallest eigenvalues and eigenvectors. The eigenvectors are the discretized solution Ψ_i and the eigenvalues are the energy levels E_i. For display purposes, we plot

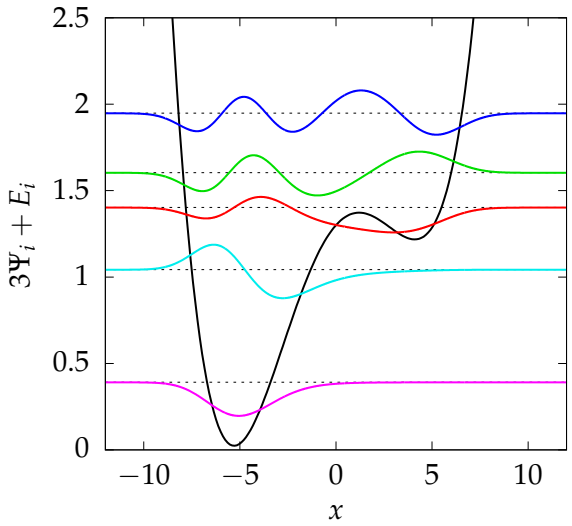
$$y_i(x) = 3\Psi_i(x) + E_i. \tag{23}$$

Figures 5, 6, and 7 show the plots of Ψ_i and tables of the corresponding energy levels.

Part (b) – probability

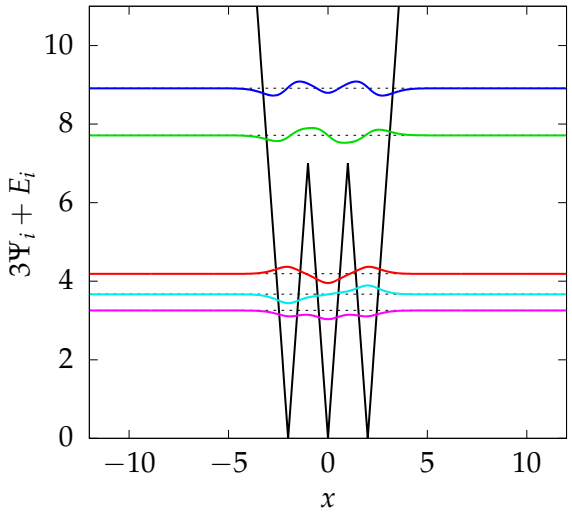
We want to compute the probability of a particle being found in an interval. We know the approximate formula is

$$p = \frac{\int_a^b |\Psi(x)|^2 dx}{\int_{-12}^{12} |\Psi(x)|^2 dx}. \tag{24}$$



E_5	1.9474321713361762
E_4	1.6033174539486321
E_3	1.401854850616602
E_2	1.0424311558478319
E_1	0.3906776858847012

Figure 6: Five wavefunctions Ψ_i and corresponding energy levels E_i for the potential $v(x) = 12(\frac{x}{10})^4 - \frac{x^2}{18} + \frac{x}{8} + 1.3$.



E_5	8.910147335373491
E_4	7.712138062039048
E_3	4.186966493986489
E_2	3.666154338642915
E_1	3.2540714519633127

Figure 7: Five wavefunctions Ψ_i and corresponding energy levels E_i for the potential $v(x) = 7|||x| - 1| - 1|$.

We use the composite Simpson's rule, which has the formula

$$\int_a^b f(x)dx \approx \frac{h}{3}\left(f(x_a) + 4f(x_1) + 2f(x_2) + 4f(x_3) + \ldots + 4f(x_{n-1}) + f(x_n)\right), \qquad (25)$$

where $h = 2(b-a)/(n-1)$. We integrate over 481 points from $x = 0$ to $x = 6$. The implementation details are included in program `quantum.py`. To four significant figures, the probabilities for the five lowest eigenmodes of potential v_2 are

$$\begin{aligned}
E_1: &\quad p(x_{\text{particle}} \in [0,6]) = 0.0003152,\\
E_2: &\quad p(x_{\text{particle}} \in [0,6]) = 0.03036,\\
E_3: &\quad p(x_{\text{particle}} \in [0,6]) = 0.7873,\\
E_4: &\quad p(x_{\text{particle}} \in [0,6]) = 0.3999,\\
E_5: &\quad p(x_{\text{particle}} \in [0,6]) = 0.5325.
\end{aligned}$$

Part (c) – fourth-order accurate method

At an interior point, the fourth-order accurate stencil is

$$(\Delta x)^2 \frac{\partial^2 \Psi_i}{\partial x^2} \approx -\frac{1}{12}\Psi_{i-2} + \frac{4}{3}\Psi_{i-1} - \frac{5}{2}\Psi_i + \frac{4}{3}\Psi_{i+1} - \frac{1}{12}\Psi_{i+2}. \qquad (26)$$

At the left end point, the fourth-order accurate one-sided stencil,

$$(\Delta x)^2 \frac{\partial^2 \Psi_1}{\partial x^2} \approx \frac{15}{4}\Psi_1 - \frac{77}{6}\Psi_2 + \frac{107}{6}\Psi_3 - 13\Psi_4 + \frac{61}{12}\Psi_5 - \frac{5}{6}\Psi_6, \qquad (27)$$

can be used. At the right end point, the same formula can be used but with the grid point ordering reversed to give

$$(\Delta x)^2 \frac{\partial^2 \Psi_n}{\partial x^2} \approx \frac{15}{4}\Psi_n - \frac{77}{6}\Psi_{n-1} + \frac{107}{6}\Psi_{n-2} - 13\Psi_{n-3} + \frac{61}{12}\Psi_{n-4} - \frac{5}{6}\Psi_{n-5}. \qquad (28)$$