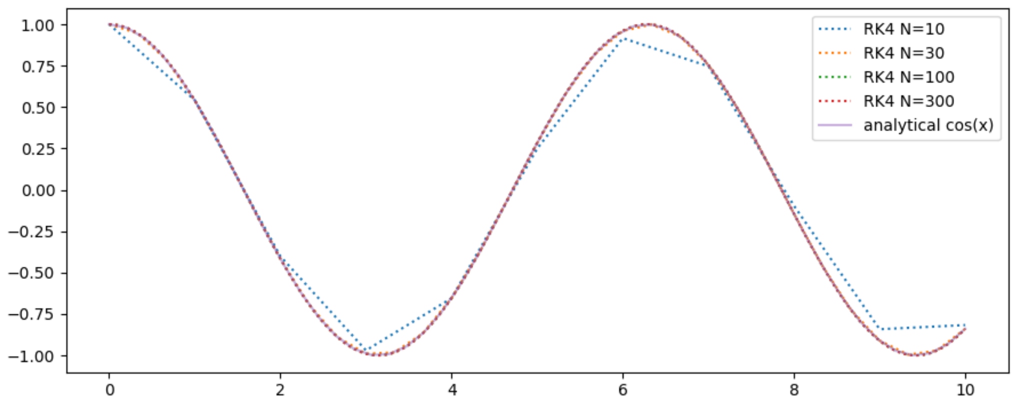
All the code runs using python3 with numpy, matplotlib and scipy as prerequisites

1

Code is at runge_kutta.py use the plot() method to show the graph

To solve the equation we substitute u = y' and get a system of 2 equations to solve.
Running the method plot() renders all approximations (including N=10 to see at least some difference) and prints the array values.

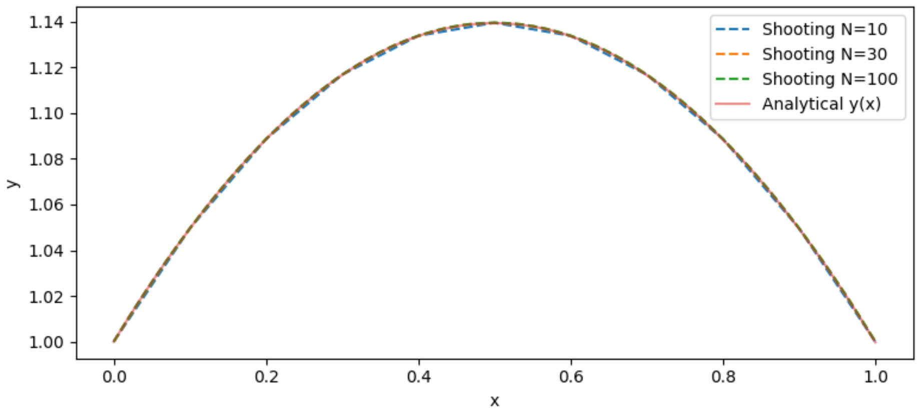


Indeed we can see that already at N=30 we get a very good solution, which is cos(x) analytically

2

Code is at shooting.py . run it to see a plot

Using the hint, we convert the problem to an initial value problem, defining a new function depending on k which includes invocation of runge_kutta_solver.
Finding the optimal k by finding its root and returning the approximation with the found k.

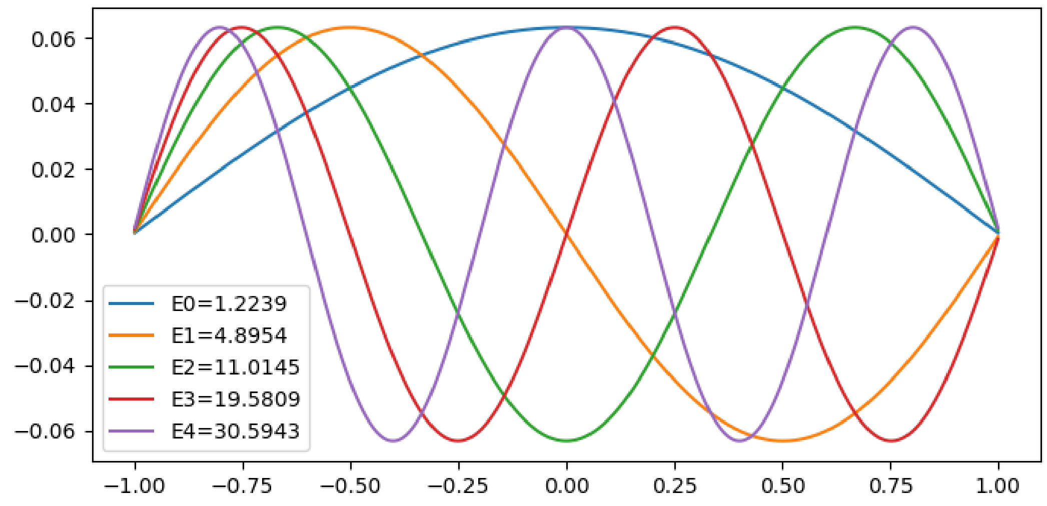


We see that the approximations fit the given analytical solution, while even N=10 (blue) gives an ok approx.

3

Code is at particle.py . Run it to get the plot.

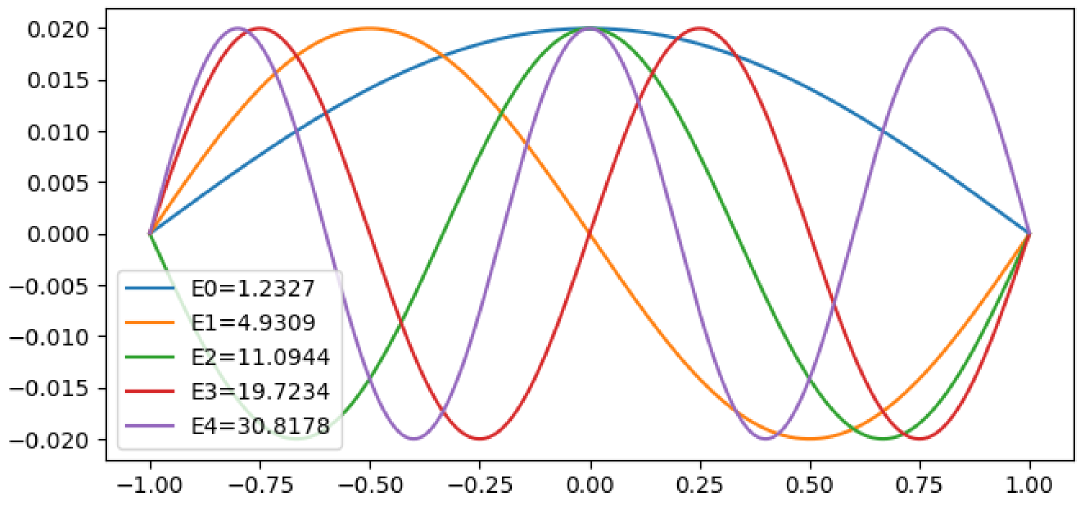
Following the three-point stencil formula from the lecture notes on the equation y'' = -2Ey (y is psi) we can just write it as Dy / -2h^2 = Ey and solve it as an eigenproblem. Running with 500 points:



We get those very nice looking wave eigenvectors with their respective eigenstates.
The states fit the analytical solution proposed in the exercise sheet

```
1.2337, 4.9348, 10.4024, 19.7392, 30.8425
```

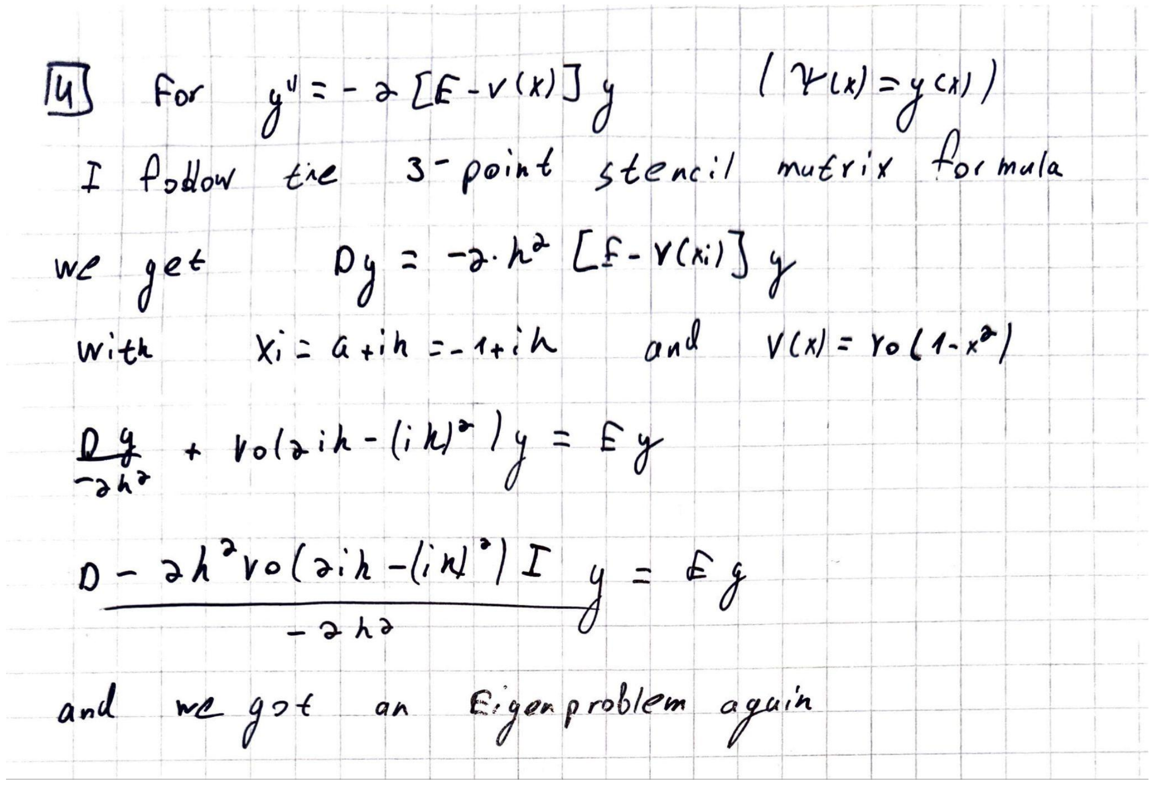
Using 500 points gives a not very accurate result, increasing it to 5000 we are able to get better ones but with a much longer running time:



4

Code is at particle_potential.py running it will print the result.

Did the following calculation:



$\boxed{4}$ For $y'' = -2[E - V(x)]y$ $(\Psi(x) = y(x))$

I follow the 3-point stencil matrix formula

we get $Dy = -2 \cdot h^2 [E - V(x_i)] y$

with $x_i = a + ih = -1 + ih$ and $V(x) = V_0(1 - x^2)$

$$\frac{Dy}{-2h^2} + V_0(2ih - (ih)^2)y = Ey$$

$$\frac{D - 2h^2 V_0(2ih - (ih)^2) I}{-2h^2} y = Ey$$

and we got an Eigenproblem again

To get the following output for the given potentials:

```
V0=1, E0=2.098761495506242
expected E0=2.101189876688455, dE0=0.002428381182213002
V0=10, E0=9.630493908609036
expected E0=9.923397338387156, dE0=0.2929034297781197
V0=30, E0=23.827166041940348
expected E0=27.30608058660649, dE0=3.478914544666143
```

We use 1500 points at the code and we indeed see that we get similar results to the perturbation theory equation. dE0 indicates the delta between the numerical and perturbation theory results.
For V0=30 the difference (error?) becomes very substantial