In [1]:
```
%reload_ext autoreload
%autoreload 2
```

1: Excitons

Kevin Vonk, s1706896, Feb 2020

Section 1: Laying the groundwork

a)

The solution to the 1D eigenvalue problem of the harmonic oscillator is

$$E_n = \left(n + \frac{1}{2}\right)\hbar\omega; n = 0, 1, 2, \dots$$

Fortunately, comparing this to the 3D solution is rather straightforward. Making use of the fact that the solution of the 3D Schrödinger equation can be seperated into three independent 1D solutions, we can write the 3D eigenvalue as follows:

$$E_{n_x,n_y,n_z} = \left[\left(n_x + \frac{1}{2}\right) + \left(n_y + \frac{1}{2}\right) + \left(n_z + \frac{1}{2}\right)\right]\hbar\omega. \tag{1}$$

Writing $n = n_x + n_y + n_z$, we can rewrite eq. (1) as:

$$E_n = \left(n + \frac{3}{2}\right)\hbar\omega,$$

which is the energy we would expect for a 3D harmonic oscillator.

b)

Noting again that

$$\begin{aligned} n &= n_x + n_y + n_z \\ n - n_x &= n_y + n_z, \end{aligned}$$

if n and n_x are given, choosing a value for either n_y or n_z determines the other. This means that there are $n - n_x + 1$ different combinations of pairs of n_y and n_z. For the degeneracy then, we have

$$\begin{aligned} \sum_{n_x=0}^{n}(n - n_x + 1) &= \sum_{n_x=0}^{n}(n+1) + \sum_{n_x=0}^{n} n_x \\ &= (n+1)^2 - \frac{1}{2}n(n+1) \\ &= \frac{1}{2}(n+1)(2n+2-n) \\ &= \frac{1}{2}(n+1)(n+2), \end{aligned}$$

which is the expected degeneracy.

c)

Introducing the dimensionless parameters $\rho = \frac{r}{r_0}$ and $\lambda = \frac{E}{V_0}$, where $r_0 = \frac{\hbar}{\sqrt{2\mu V_0}}$ and $V_0 = \hbar\omega$, we can write the (radial) Schrödinger equation in dimensionless form. Starting from the radial Schrödinger equation,

$$\left(-\frac{\hbar^2}{2\mu}\frac{d^2}{dr^2} + \frac{\hbar^2 l(l+1)}{2\mu r^2} + V(r)\right)\zeta(r) = E\zeta(r),$$

which can be divided by V_0 to obtain

$$\left(-\frac{\hbar^2}{2\mu V_0}\frac{d^2}{dr^2} + \frac{\hbar^2 l(l+1)}{2\mu r^2 V_0} + \frac{V(r)}{V_0}\right)\zeta(r) = \frac{E}{V_0}\zeta(r).$$

Replacing expressions with the dimensionless parameters and changing $r \longrightarrow \rho$, we obtain

$$\left[-(r_0)^2\frac{d^2}{dr^2} + \frac{l(l+1)}{\rho^2} + \frac{V(r)}{V_0}\right]\zeta(r) = \lambda\zeta(r)$$
$$\left[-(r_0)^2\frac{d^2\rho}{dr^2}\frac{d^2}{d\rho^2} + \frac{l(l+1)}{\rho^2} + \frac{V(\rho)}{V_0}\right]\zeta(\rho) = \lambda\zeta(\rho)$$
$$\left[-\frac{d^2}{d\rho^2} + \frac{l(l+1)}{\rho^2} + \frac{V(\rho)}{V_0}\right]\zeta(\rho) = \lambda\zeta(\rho)$$

,

which can be written in a more suggestive form,

$$\frac{d^2}{d\rho^2}\zeta(\rho) = (W(\rho) - \lambda)\,\zeta(\rho), \tag{2}$$

where

$$W(\rho) = \frac{l(l+1)}{\rho^2} + \frac{V(\rho)}{V_0}. \tag{3}$$

In order to determine the outer turning point (OTP), we must solve $W(\rho) - \lambda = 0$. Filling in $V(\rho)$ and applying some algebra, we must solve

$$\frac{l(l+1)}{\rho^2} + \frac{1}{4}\rho^2 - \lambda_n = 0 \quad \text{with } n = 2k + l. \tag{4}$$

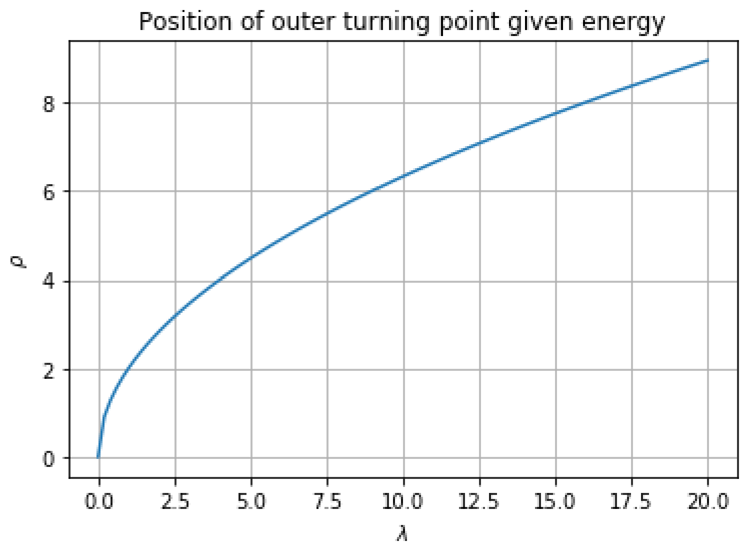
In the cases where $l \neq 0$, λ is a function of l, which means that deriving an analytical expression for the OTP as a function of λ is cumbersome. In these instances we can programatically calculate the OTP by calculating eq. (4) for every point ρ_i of a grid. The OTP is the point where the sign of the solution has changed. In the case that $l = 0$ however, we can simplify eq. (4) as

$$\frac{1}{4}\rho^2 - \lambda_n = 0,$$
$$\rho_{otp} = 2\sqrt{\lambda_n},$$

which shows that for the given potential, the position of the OTP increases with the root of the energy. A plot of this function follows.

In [2]:
```
import numpy as np
import matplotlib.pyplot as plt
from excitons import *

lambda_ = np.linspace(0, 20, 100)
pot = HarmonicPotential()
otp = [pot.outerTurningPoint(i) for i in lambda_]

plt.plot(lambda_, otp)
plt.grid()
plt.title("Position of outer turning point given energy")
plt.xlabel(r"$\lambda$")
plt.ylabel(r"$\rho$")
```

Out[2]:
```
Text(0, 0.5, '$\\rho$')
```



d)

In order to compare any computed wavefunction, it pays to compare it to the analytical solution. The solution is given by the assignment sheet as follows,

$$\zeta_{k,r}(r) = A_{kl} r^{l+1} e^{-\nu r^2} L_k^{(l+\frac{1}{2})}(2\nu r^2); \quad \text{with } \nu = \frac{\mu\omega}{2\hbar}.$$

Rewriting in dimensionless form using the parameters for the harmonic oscillator, we obtain

$$\zeta_{k,r}(\rho) = A_{kl}(\rho r_0)^{l+1} e^{-\frac{1}{4}\rho^2} L_k^{(l+\frac{1}{2})}\left(\frac{1}{2}\rho^2\right).$$

Since $(r_0)^{l+1}$ is constant for a given l, we can merge this with the constant A_{kl} to obtain the analytical solution of the wavefunction for the harmonic potential in dimensionless parameters:

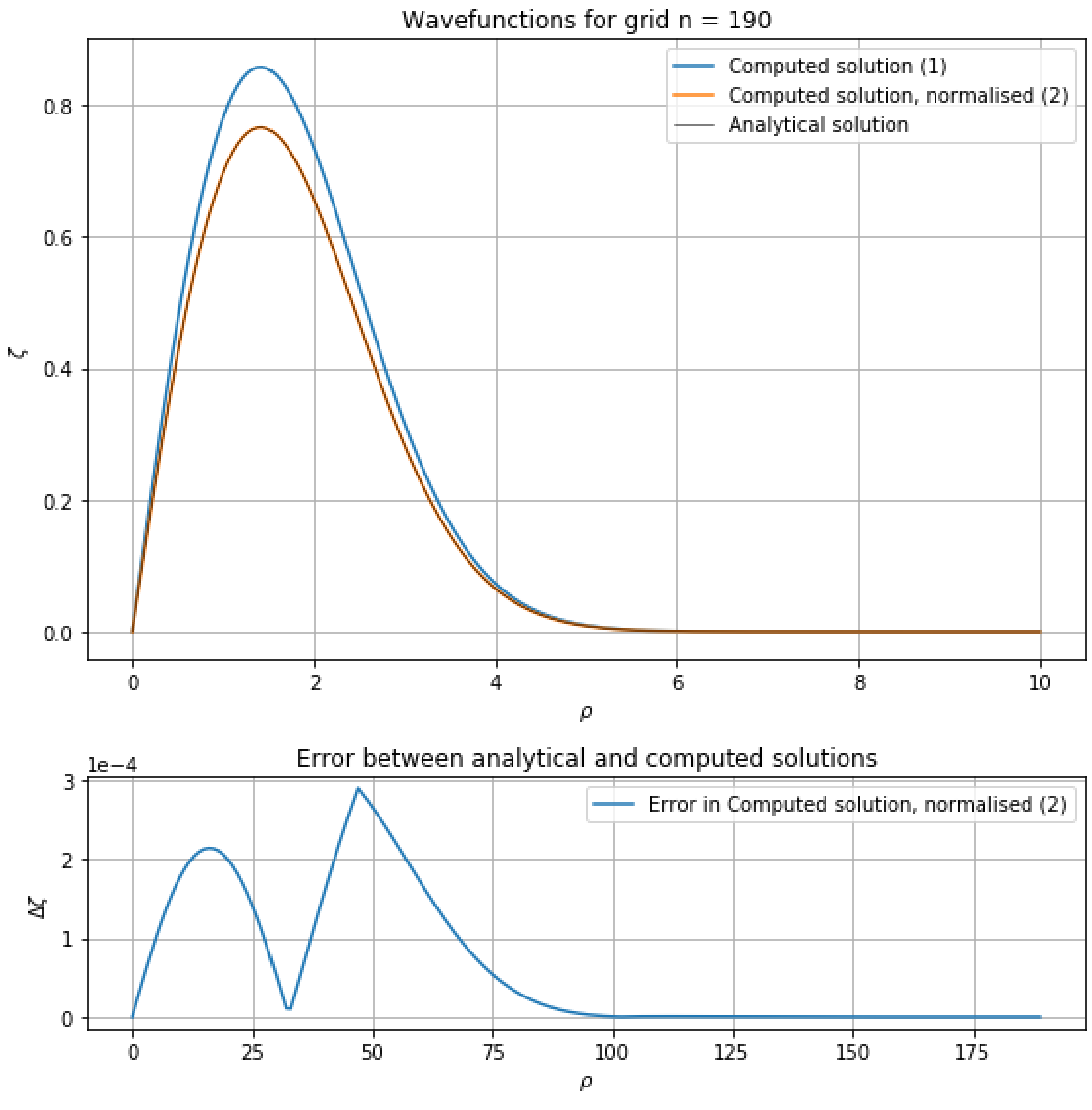
$$\zeta_{k,r}(\rho) = A_{kl}(\rho)^{l+1} e^{-\frac{1}{4}\rho^2} L_k^{(l+\frac{1}{2})}\left(\frac{1}{2}\rho^2\right).$$

Next, we define the grid parameters. In the case that $l = 0$, the first term in W (eq. 3) drops out. This allows us to pick any value for a, since ρ can take on any (positive) value. Let us then pick the easiest and most trivial point for this offset, $a = 0$. When $l \neq 0$, this first term does not drop out, which means that $a \neq 0$. Otherwise, division by zero would occur. We will define an appropriate offset value when we come across situations where $l \neq 0$. A requirement for ζ is that $\zeta \to 0$ for $\rho \to \inf$. By plotting ζ for large ρ, we were able to determine that for $\rho = 10$, $\zeta \approx 0$. This means that $\rho \in [0, 10]$. Lastly, we need a definition for the number of grid points for our computed solution. Given is the error $\epsilon < 3 * 10^{-4}$. Using trial and error, we are able to calculate the error for any number of grid points. Using this method, we were able to determine that 190 grid points, which are 189 steps, are optimal for the required accuracy. Combining all the observations and restrictions, we are able to define the grid

$$\rho = a + jh \quad \text{where } j = 0, 1, \ldots, N$$

using $a = 0$, $N = 189$ and $h \approx 0,05291$. A plot of the numerical and analytical wave functions, as well as the error between them for the defined grid is found below:

In [3]:
```
# Setup the grid, potential and wave functions
grid = Grid(190)
pot = HarmonicPotential()
wf_not_normalised = WaveFunction(grid, pot)
wf = WaveFunction(grid, pot)
ana = WaveFunction(grid, pot)

# Calculate and normalise the wave functions
wf_not_normalised.propagate(3/2)
wf.propagate(3/2)
wf.normalise()
ana.analytical()
ana.normalise()

legend = ("Computed solution (1)", "Computed solution, normalised
 (2)")

plot(grid, ana, wf_not_normalised, wf, legend=legend, err_index=1)
```



As we can observe from the error plot, at the OTP there exists a discontinuity. This is to be expected: we arrive at this point using two different boundary functions, and only ensure that at the OTP the value of both the foward and backward propagation are the same. This is the best we can do however, since the derivatives of both propagations needn't be the same.

Section 2: Applying improvements

a)

Numerov's method is a mathematical trick which allows us to improve the accuracy of our computations without the cost normally associated with such improvements. Numerov's method is only applicable in very select cases, but our problem turns out to be exactly such a case. Following par. 1.5 of the lecture notes, we start with the general form of Numerov's method,

$$\frac{d^2 f}{dx^2} = g(x) f(x). \tag{5}$$

We can rewrite this using the functions and variables of our problem, giving

$$\frac{d^2 \zeta}{d\rho^2} = \left(W(\rho) - \lambda\right) \zeta(\rho),$$

where $f(\rho) = \zeta(\rho)$ and $g(\rho) = W(\rho) - \lambda$. We can rewrite the ODE as shown in eq. (24) of the lecture notes,

$$h^2 \frac{d^2 \zeta}{d\rho^2} = f(\rho + h) q(\rho + h) - 2 f(\rho) q(\rho) + f(\rho - h) q(\rho - h),$$

where $q(\rho) = 1 - \frac{h^2}{12} g(\rho)$. If we replace the differential with eq. (5) and the given definitions for $f(\rho)$ and $g(\rho)$, we obtain

$$h^2 \left(W(\rho) - \lambda\right) \zeta(\rho) = \zeta(\rho + h) q(\rho + h) - 2\zeta(\rho) q(\rho) + \zeta(\rho - h) q(\rho - h).$$

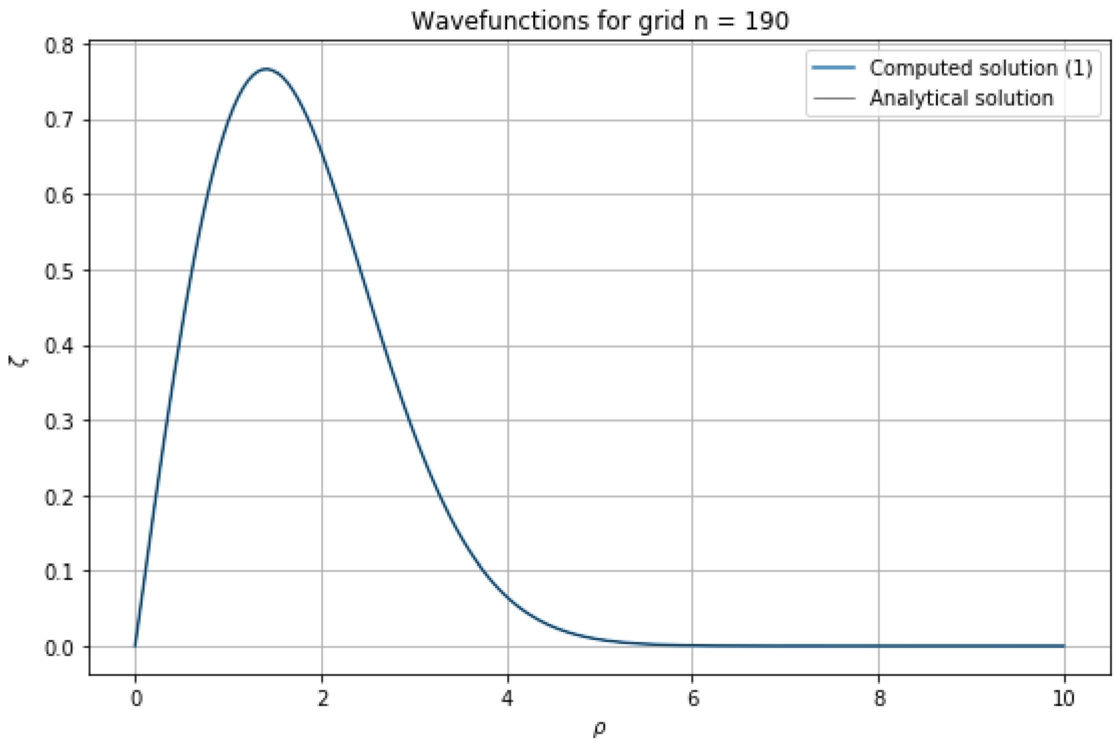
We are able to rewrite this eq. in such a way that we can extract both the forward $(\zeta(\rho + h))$ and backward $(\zeta(\rho - h))$ propagations, which become

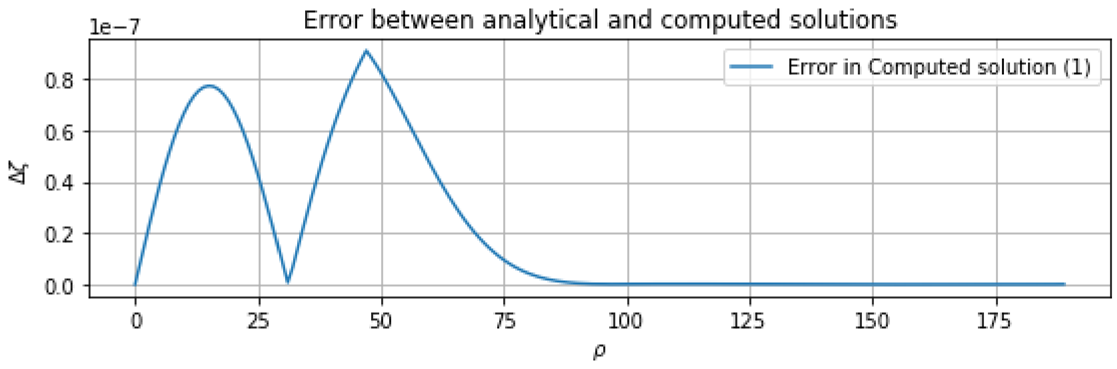
$$\zeta(\rho + h) = \frac{1}{q(\rho + h)} \left[h^2 (W(\rho) - \lambda) \zeta(\rho) + 2\zeta(\rho) q(\rho) - \zeta(\rho - h) q(\rho - h)\right],$$

$$\zeta(\rho - h) = \frac{1}{q(\rho - h)} \left[h^2 (W(\rho) - \lambda) \zeta(\rho) + 2\zeta(\rho) q(\rho) - \zeta(\rho + h) q(\rho + h)\right].$$

Having implemented Numerov's method, taking the same grid parameters as defined in 1d) gives us the following plot:

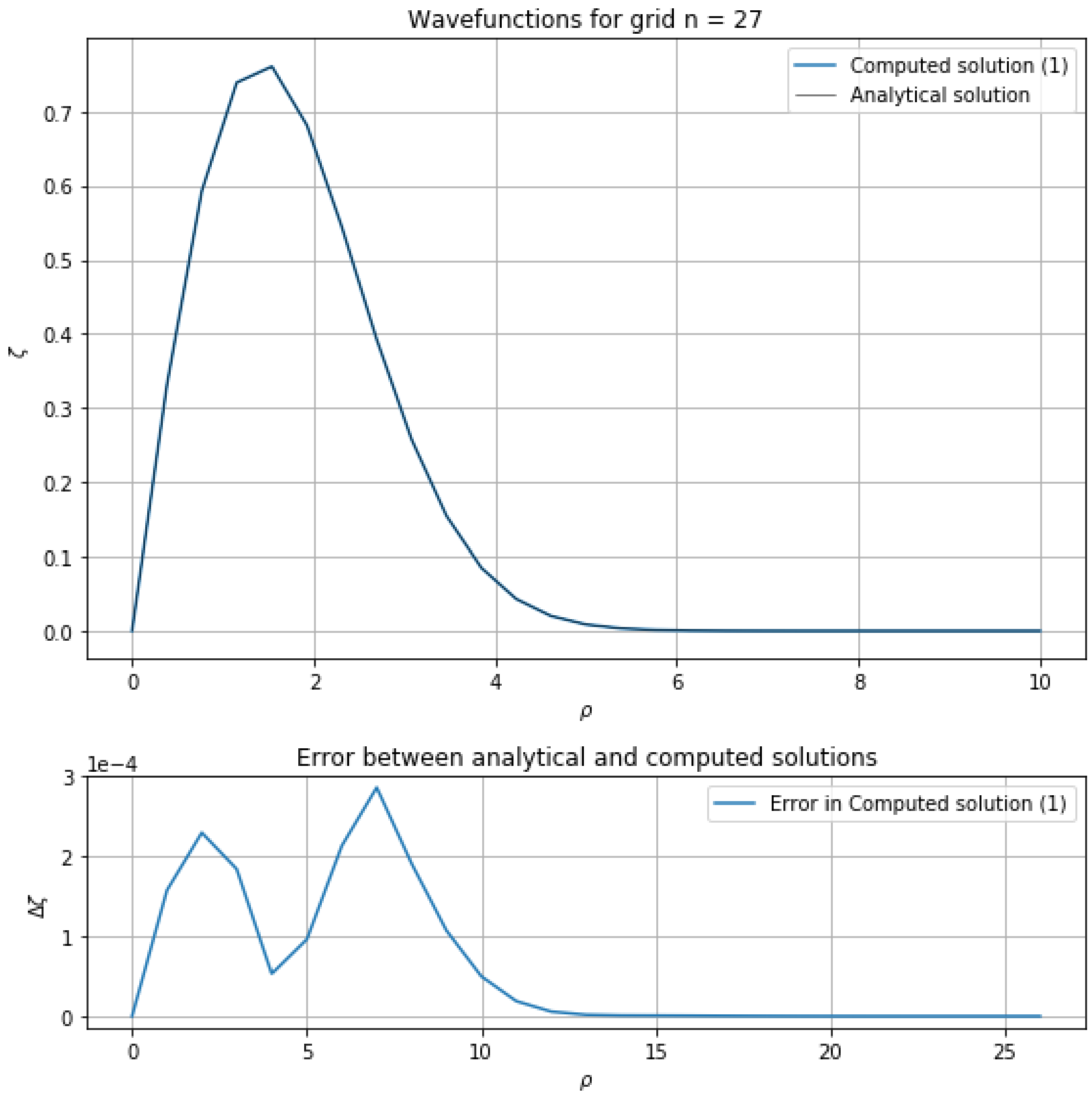
```
In [4]: wf.propagate(3/2, numerov=True)
        wf.normalise()

        plot(grid, ana, wf)
```





We can see that the accuracy has improved from $\sim 3 * 10^{-4}$ to $\sim 1 * 10^{-7}$, an order of 10^3 improvement. This means that we can increase our accuracy whilst having the same grid using Numerov's method, or we can decrease the amount of grid points to obtain the same required accuracy. Let us now redefine our grid using Numerov's method, aiming for an accuracy of $\epsilon = 3 * 10^{-4}$ again. There is no reason to redefine the limits of the grid, since Numerov's method does not influence these. We can also state that the limits are large enough, such that any edge effects are much smaller than the requested accuracy. In the case where $\rho_{max} = 8$ for example, Numerov's method introduces a deviation from the analytical solution near this limit, which dominates the accuracy of the numerical solution. Using trial and error, we were able to determine that a grid consisting of 27 grid points (26 steps) falls within the required accuracy. This then redefines $N = 26$ and $h = 0,3846$, which is a huge increase in step size compared to 1d). The plot for this grid can be found below.

```
In [5]: # Setup the grid, potential and wave functions
        grid = Grid(27)
        pot = HarmonicPotential()
        wf = WaveFunction(grid, pot)
        ana = WaveFunction(grid, pot)

        # Calculate and normalise the wave functions
        wf.propagate(3/2, numerov=True)
        wf.normalise()
        ana.analytical()
        ana.normalise()

        plot(grid, ana, wf)
```



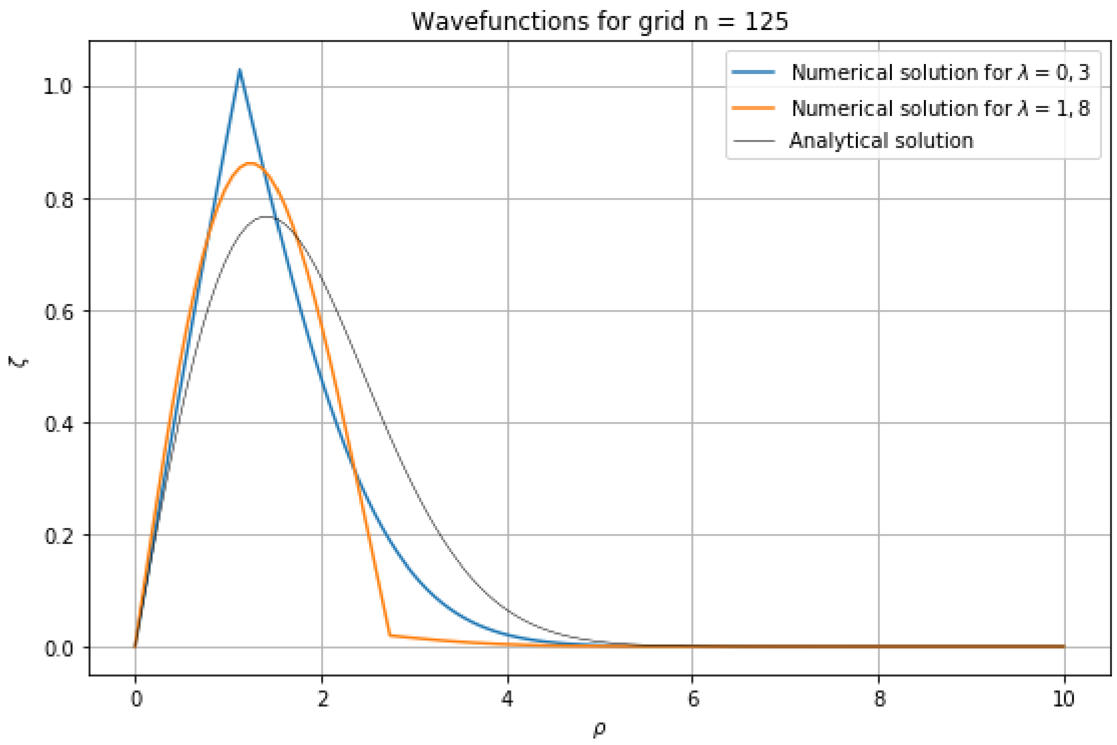
b)

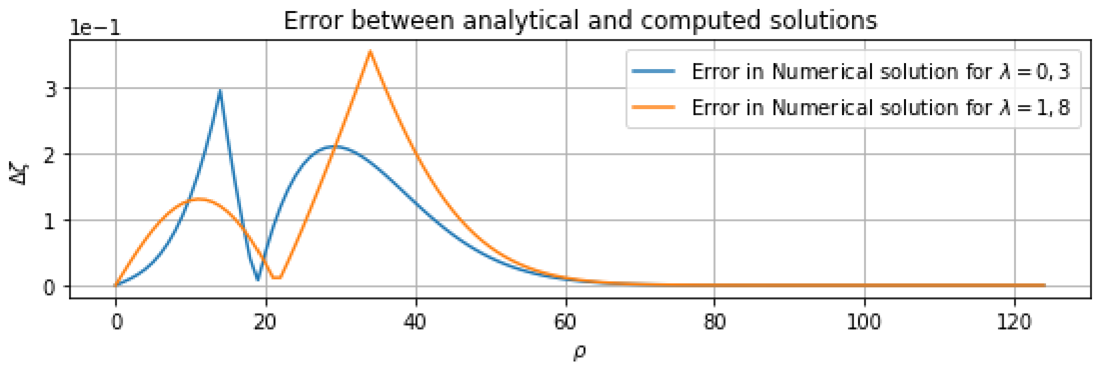
The shooting method has been implemented according to the steps of the lecture notes. The relevant plots can be found below. In order to obtain sufficient accuracy in the energy eigenvalue is was necessary to increase the amount of grid points. With a too course of a grid, it is not possible to make the fine adjustments needed to converge on the analytical value of the energy eigenvalue.

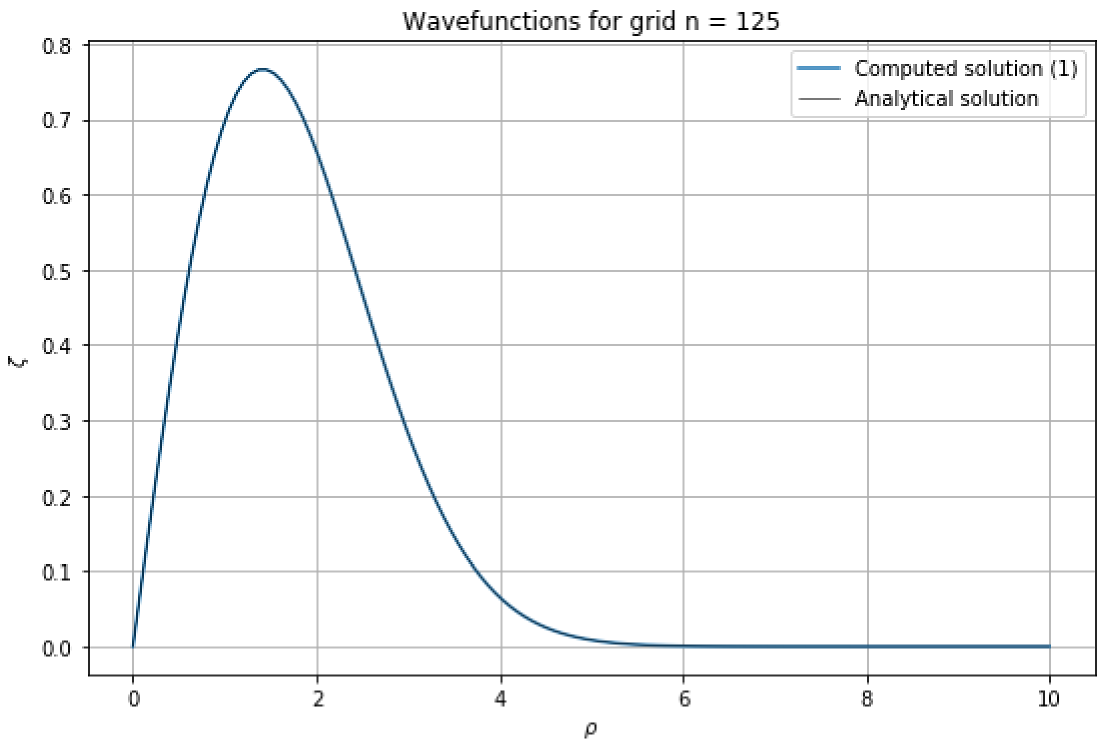
Looking at the computed wavefunctions for the initial eigenvalues, we can see large discontinuities at the outer turning point. As expected, they also do not follow the shape of the analytical solution. Looking at the convergence of the energy eigenvalue, it is of note that it looks to converge quite quickly at the start. After this initial period, the convergence rate slows down. F oscillates, where its sign changes after every two iterations. We can also see this behaviour back in the eigenvalue at an iteration index, where the eigenvalue increases when $F < 0$ and decreases when $F > 0$.

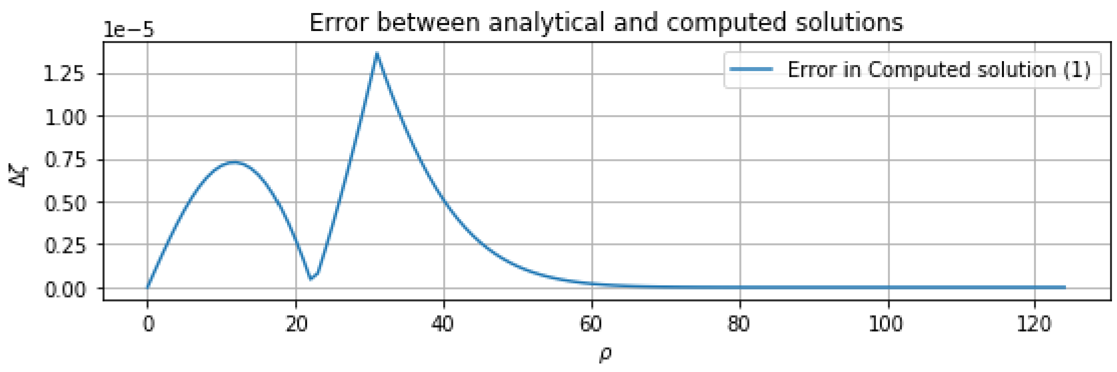
In [6]:
```python
import pandas as pd
from IPython.display import display
pd.set_option("display.precision", 10)

grid = Grid(125)
ana = WaveFunction(grid, pot).analytical().normalise()

plot(grid, \
    ana, \
    WaveFunction(grid, pot).propagate(0.3, numerov=True).normalise(),
\
    WaveFunction(grid, pot).propagate(1.8, numerov=True).normalise(),
\
    legend=(r"Numerical solution for $\lambda = 0,3$", r"Numerical so
lution for $\lambda = 1,8$"),
    err_index=(0,1)
)

solver = Solver(grid, pot)
iterations = solver.bisect(0.3, 1.8, fullreturn=True)

data = {r"$\lambda$": iterations[0], r"$F(\lambda)$": iterations[1]}
df = pd.DataFrame(data=data).rename_axis("Iteration no.", axis=1)
display(df)

plot(grid, ana, solver.wf)
```

Iteration no.	λ	$F(\lambda)$
0	1.0500000000	-0.1150147517
1	1.4250000000	-0.0241567553
2	1.6125000000	0.0414242903
3	1.5187500000	0.0064606624
4	1.4718750000	-0.0096899593
5	1.4953125000	-0.0016152011
6	1.5070312500	0.0024230181
7	1.5011718750	0.0004039257
8	1.4982421875	-0.0006056401
9	1.4997070313	-0.0001008569
10	1.5004394531	0.0001515346
11	1.5000732422	0.0000253388
12	1.4998901367	-0.0000377590
13	1.4999816895	-0.0000062101







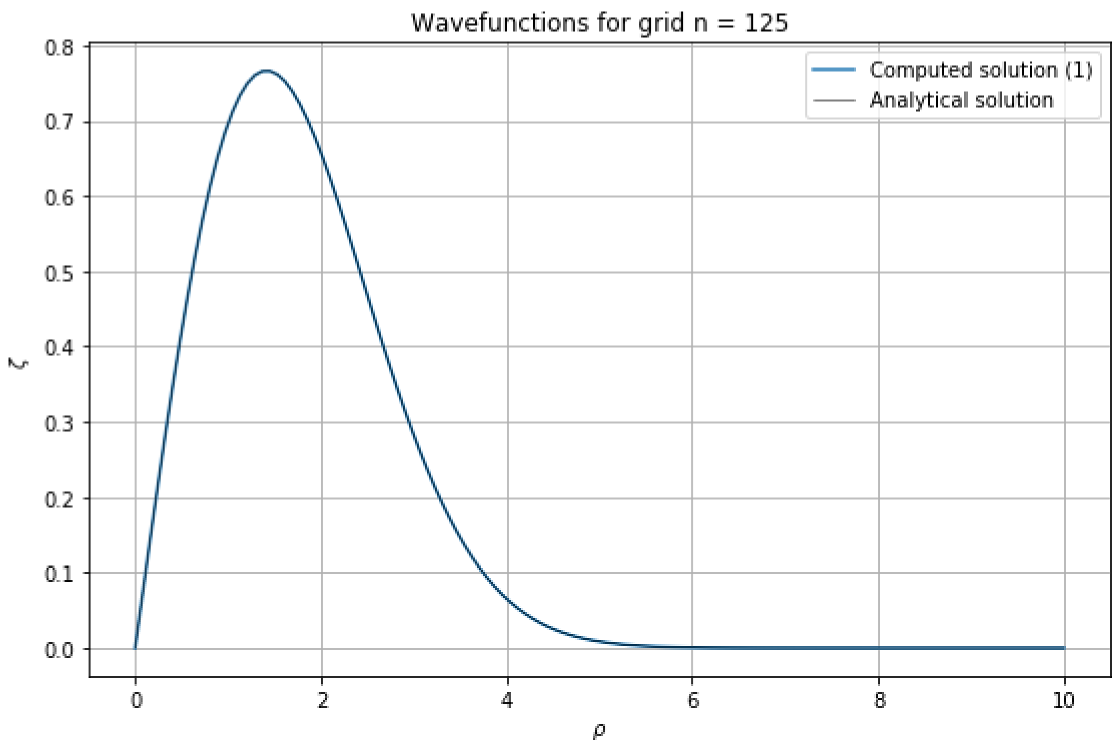


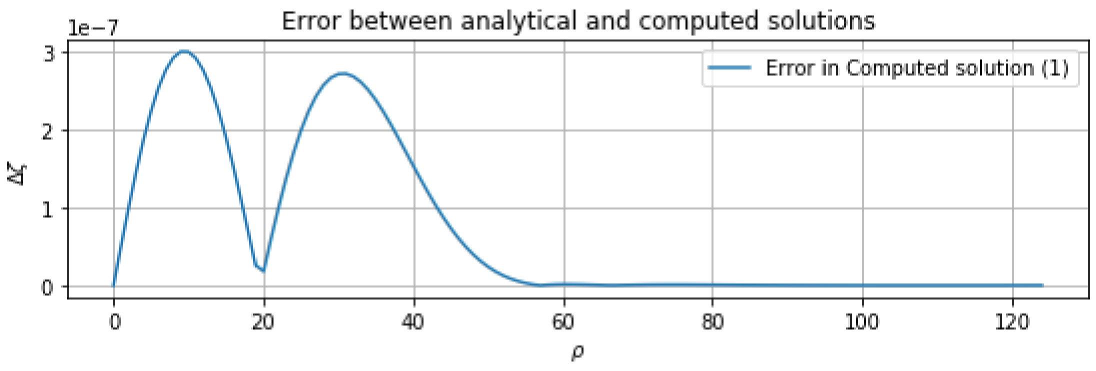
c)

The improved eigenvalue search has been implemented according to the lecture notes. The results are depicted below. Of note is the convergence table, which shows increased accuracy with less iterations. Compared to the previous assignment, the allowed error has been reduced from 10^{-4} to 10^{-7}. Thus, the eigenvalue converges quicker using this new method.

```
In [7]: solver = Solver(grid, pot, error=1E-7)
        iterations = solver.search(0.4, fullreturn=True)

        data = {r"$\lambda$": iterations[0], r"$F(\lambda)$": iterations[1]}
        df = pd.DataFrame(data=data).rename_axis("Iteration no.", axis=1)
        display(df)

        plot(grid, ana, solver.wf)
```

Iteration no.	λ	$F(\lambda)$
0	0.4000000000	-2.2748458781e-01
1	1.8123545075	1.2700853770e-01
2	1.8092408938	1.2590661499e-01
3	1.8030942612	1.2371622822e-01
4	1.7911196072	1.1939296795e-01
5	1.7684111507	1.0329284874e-01
6	1.6923859425	7.5401445399e-02
7	1.5965530512	3.5596020216e-02
8	1.5194376933	6.6975673208e-03
9	1.5006279455	2.1648870236e-04
10	1.5000003176	2.0914060406e-07
11	1.4999997106	-1.4144796445e-11





Section 3: Coulomb potential

a)

In order to use the Coulomb potential, we must first ensure the Schrödinger equation is dimensionless. Using the given parameters (where only $V(r)$ and V_0 have changed):

$$\rho = \frac{r}{r_0},\ r_0 = \frac{\hbar}{\sqrt{2\mu V_0}},\ V(r) = \frac{-e^2}{4\pi\epsilon r},\ V_0 = \frac{\mu e^4}{32\pi^2\epsilon^2\hbar^2},$$

we can follow the same steps as done in section 1c) to make the Schrödinger equation dimensionless. Since ρ and r_0 have not been altered in their definition, we needn't recalculate much. The only part which differs from 1c) is the potential part in the Schrödinger eq., namely $\frac{V(r)}{V_0}$. Replacing r by ρ in $V(r)$ gives

$$\begin{aligned} V(\rho) &= \frac{-e^2}{4\pi\epsilon\rho r_0} \\ &= \frac{-e^2\sqrt{2\mu V_0}}{4\pi\epsilon\rho\hbar} \\ &= \frac{-\mu e^4}{16\pi^2\epsilon^2\hbar^2\rho} \\ &= -\frac{2}{\rho}V_0, \end{aligned}$$

which means that $\frac{V(\rho)}{V_0} = -\frac{2}{\rho}$. We can thus write the Schrödinger equation using

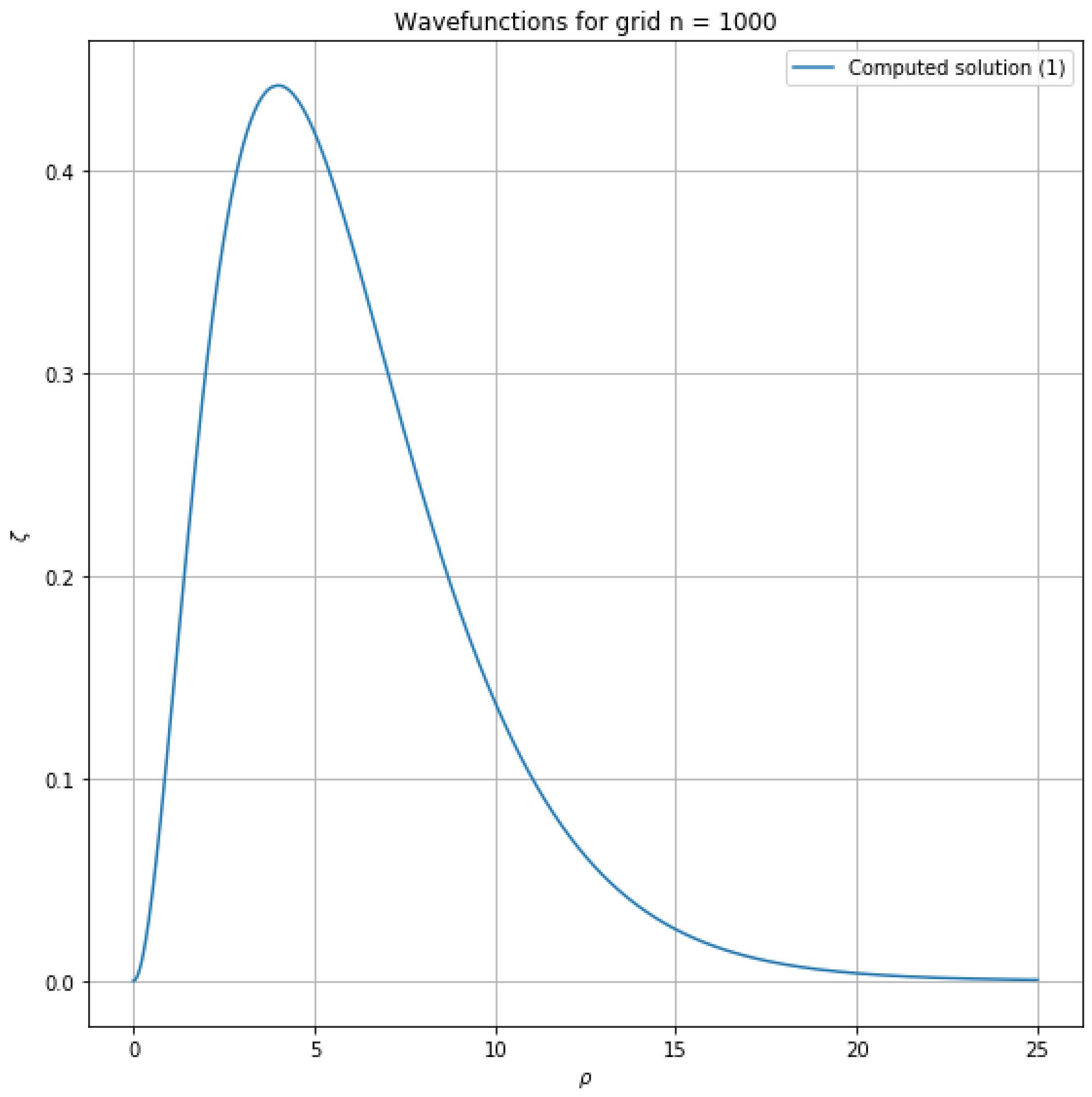
$$W(\rho) = \frac{l(l+1)}{\rho^2} - \frac{2}{\rho}.$$

Now, it is necessary to redefine the grid for this new potential. ρ cannot be zero anymore, since this would introduce a division by zero. We thus need an non-zero offset, for which I picked $a = 10^{-6}$. Using trial and error, I picked $\rho_{end} = 25$ and $N = 999$, which in turn gives $h = 0,02503$. Furthermore, I was experiencing overshooting of the F function, so I introduced a damping factor $\beta = 0,75$. The plot of the computed eigenvalue can be found below.

The found wavefunction and eigenvalue are in fact corresponding to the lowest possible eigenvalue for $l = 1$. The analytical eigenvalue is of the form $-\frac{1}{n^2}$. Since $l = 1, n \geq 2$, so for the lowest value of n, $\lambda = -0,25.$

In [8]:
```
lambda_guess = -0.3
grid = Grid(1000, start=1E-6, end=25)
pot = CoulombPotential()
solver = Solver(grid, pot, error=1E-5, l = 1)

iterations = solver.search(lambda_guess, damping=0.75, fullreturn=Tru
e)

data = {r"$\lambda$": iterations[0], r"$F(\lambda)$": iterations[1]}
df = pd.DataFrame(data=data).rename_axis("Iteration no.", axis=1)
display(df)

plot(grid, None, solver.wf)
```

Iteration no.	λ	$F(\lambda)$
0	-0.3000000000	-0.0108436812
1	-0.2222557396	0.0089077163
2	-0.2256475349	0.0078672472
3	-0.2312870793	0.0059114339
4	-0.2393926966	0.0033750309
5	-0.2466523402	0.0010685521
6	-0.2496748046	0.0001044510
7	-0.2499990203	0.0000010011

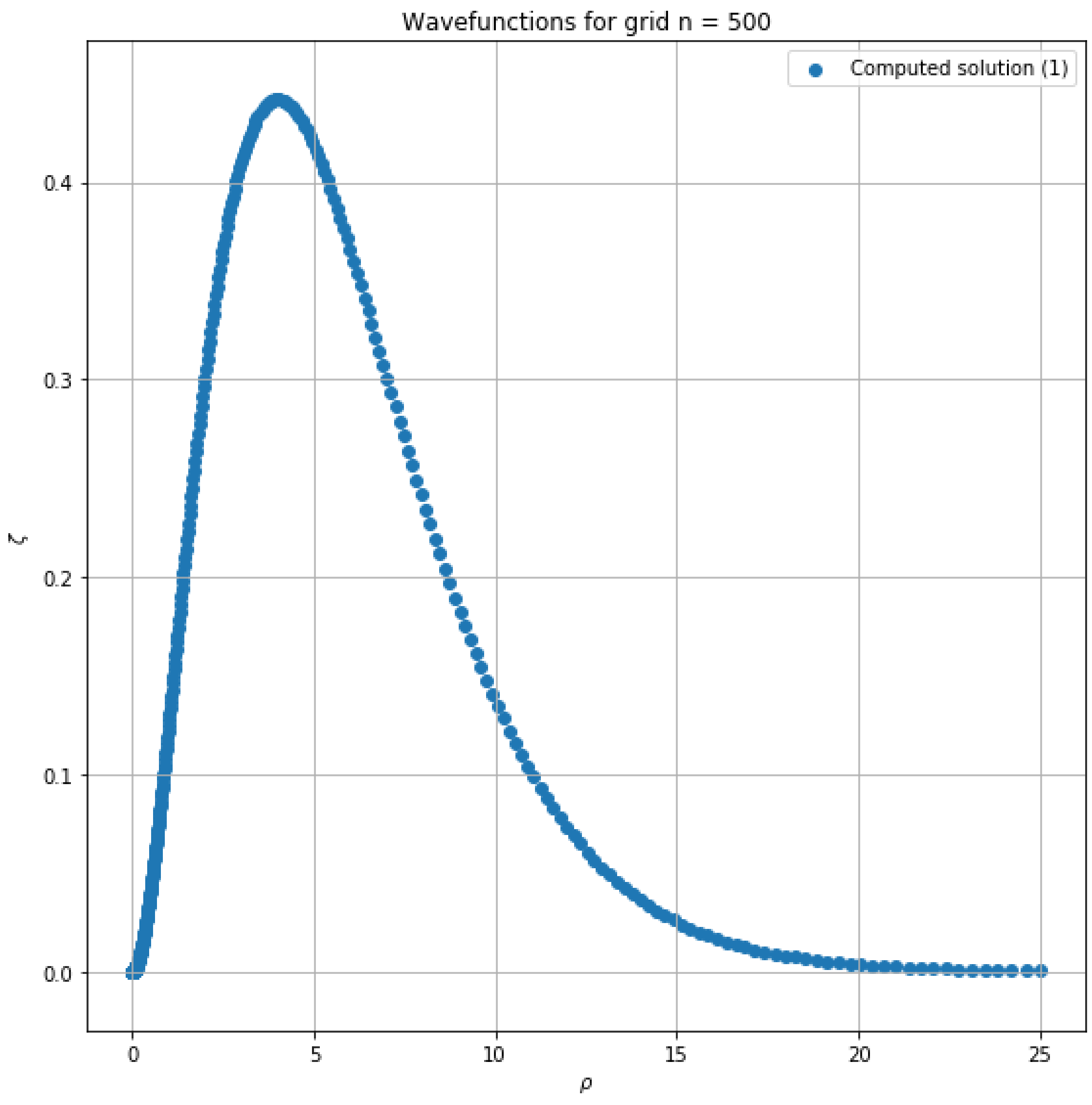


b)

Using trial and error, the grid has been defined as $a = 10^{-2}$, $h = 0,0501$, $N = 499$. The wavefunction and convergence of the eigenvalue can be found below. It is advantageous to use a logarithmic grid, since it allows for the same resulting accuracy using less grid points. Furthermore, the log grid ensures that there are more points defining the start of the wavefunction, where the most interesting behaviour lies. The tail end of the wave function is defined using a courser grid, since we can expect less erratic behaviour there.

In [9]:
```
grid = Grid(500, start=np.log(1E-2), end=np.log(25), type=GridType.LO
G)
pot = CoulombPotentialLog()
solver = Solver(grid, pot, error=1E-5, l=1)

iterations = solver.search(lambda_guess, damping=0.75, fullreturn=Tru
e)

data = {r"$\lambda$": iterations[0], r"$F(\lambda)$": iterations[1]}
df = pd.DataFrame(data=data).rename_axis("Iteration no.", axis=1)
display(df)

plot(np.exp(grid), None, solver.wf * np.sqrt(np.exp(grid) + grid.t1),
scatter=True)
```

Iteration no.	λ	$F(\lambda)$
0	-0.3000000000	-0.0074947259
1	-0.2225723018	0.0058976624
2	-0.2263742015	0.0051146038
3	-0.2325280422	0.0038138343
4	-0.2405313984	0.0020508730
5	-0.2473371352	0.0005787738
6	-0.2497969770	0.0000450695
7	-0.2500033750	0.0000002793



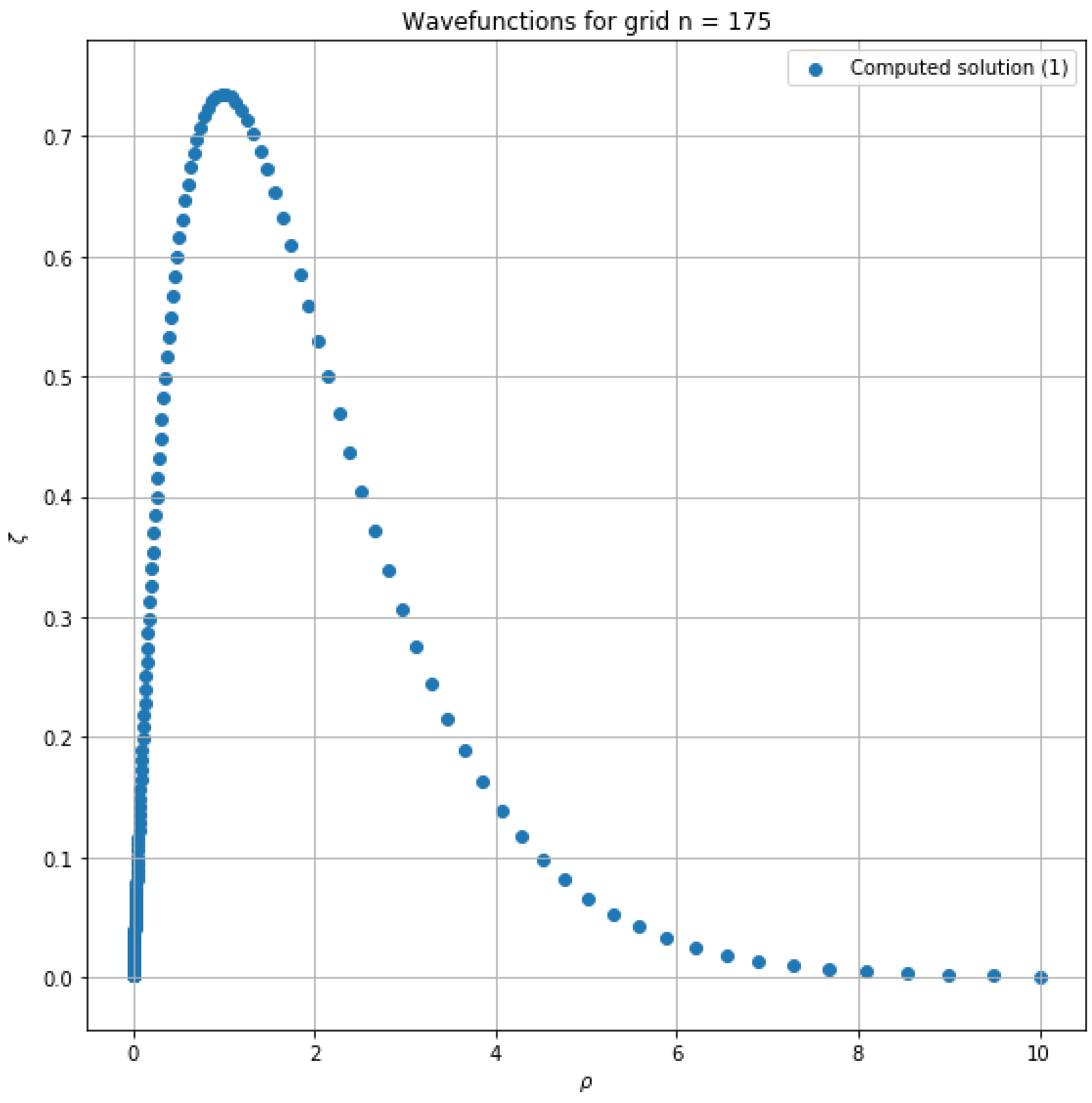
c)

Again using trial and error, the grid was found to be $a = 10^{-3}$, $h = 0,05747$, $N = 174$. The results can be found below.

The main difference in the grid compared to b) is that the grid ends sooner and that the grid is slightly courser. We are also able to obtain sufficient accuracy in fewer iterations.

In [10]:
```
lambda_guess = -1.75
grid = Grid(175, start=np.log(1E-3), end=np.log(10), type=GridType.LO
G, t1=0)
pot = CoulombPotentialLog()
solver = Solver(grid, pot, error=1E-5, l=0)

iterations = solver.search(lambda_guess, damping=1, fullreturn=True)

data = {r"$\lambda$": iterations[0], r"$F(\lambda)$": iterations[1]}
df = pd.DataFrame(data=data).rename_axis("Iteration no.", axis=1)
display(df)

plot(np.exp(grid), None, solver.wf * np.sqrt(np.exp(grid) + grid.t1),
scatter=True)
```

Iteration no.	λ	$F(\lambda)$
0	-1.7500000000	-0.1175905010
1	-0.8188021961	0.0737553725
2	-0.9006806924	0.0339189851
3	-0.9827877536	0.0050592875
4	-1.0001462154	0.0000932884
5	-1.0004726148	-0.0000000953

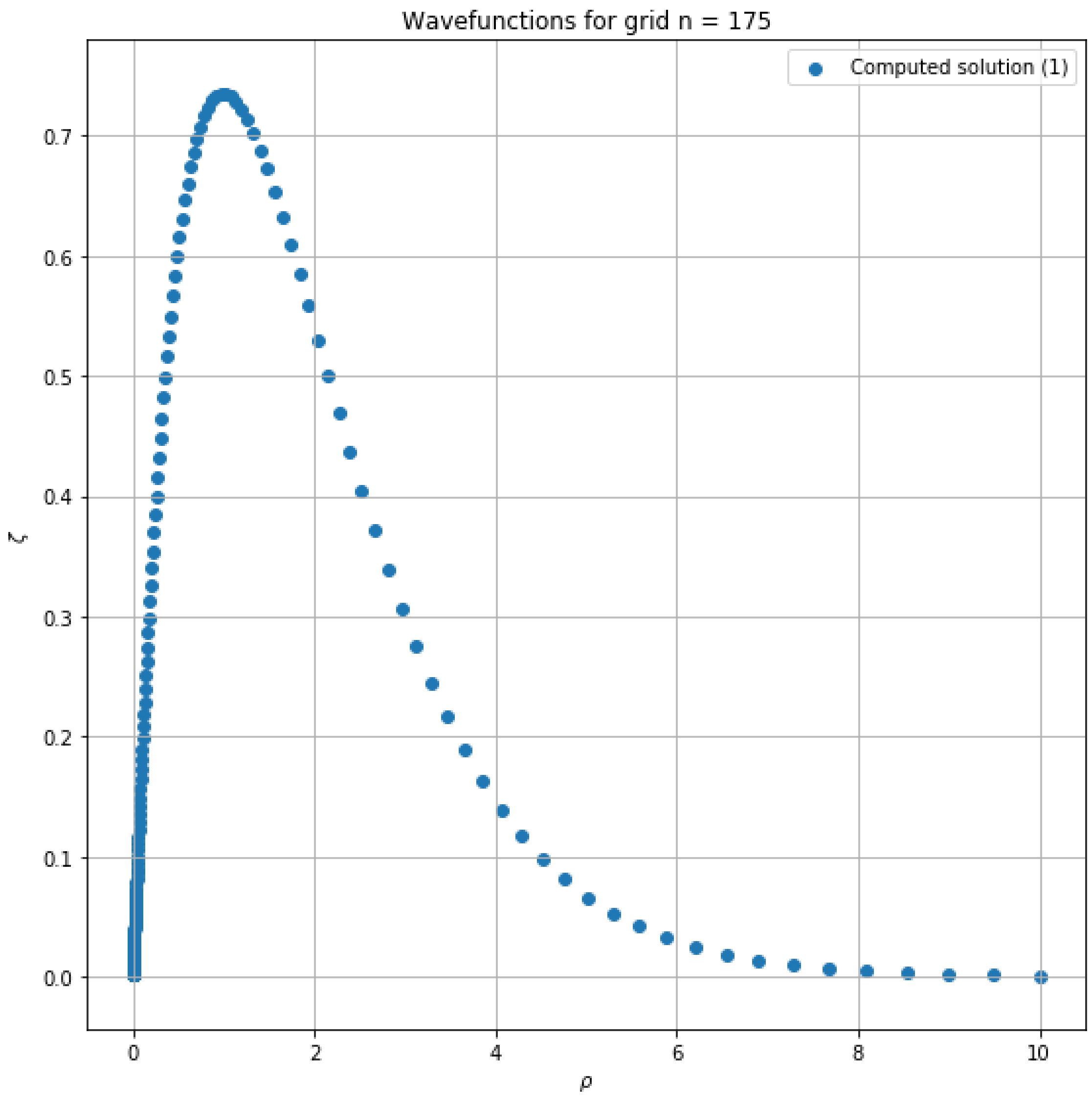


d)

The only difference in the grid compared to c) is that $\tau_1 = 10^{-4}$. Using this, we obtained slightly more accuracy in our result. This means that we could use a courser grid to calculate the energy eigenvalue. However, as we have seen previously, a courser grid means that it is harder to obtain an accurate result, since there is not enough resolution in the grid for the eigenvalue search to calculate the correct F to alter the eigenvalue.

In [11]:
```
lambda_guess = -1.75
grid = Grid(175, start=np.log(1E-3), end=np.log(10), type=GridType.LO
G, t1=1E-4)
pot = CoulombPotentialLog()
solver = Solver(grid, pot, error=1E-5, l=0)

iterations = solver.search(lambda_guess, damping=1, fullreturn=True)

data = {r"$\lambda$": iterations[0], r"$F(\lambda)$": iterations[1]}
df = pd.DataFrame(data=data).rename_axis("Iteration no.", axis=1)
display(df)

plot(np.exp(grid), None, solver.wf * np.sqrt(np.exp(grid) + grid.t1),
scatter=True)
```

Iteration no.	λ	$F(\lambda)$
0	-1.7500000000	-0.1176063708
1	-0.8186465909	0.0737689810
2	-0.9005420399	0.0339287340
3	-0.9826769435	0.0050616291
4	-1.0000444543	0.0000933665
5	-1.0003711480	-0.0000000953



Section 4: The application

a)

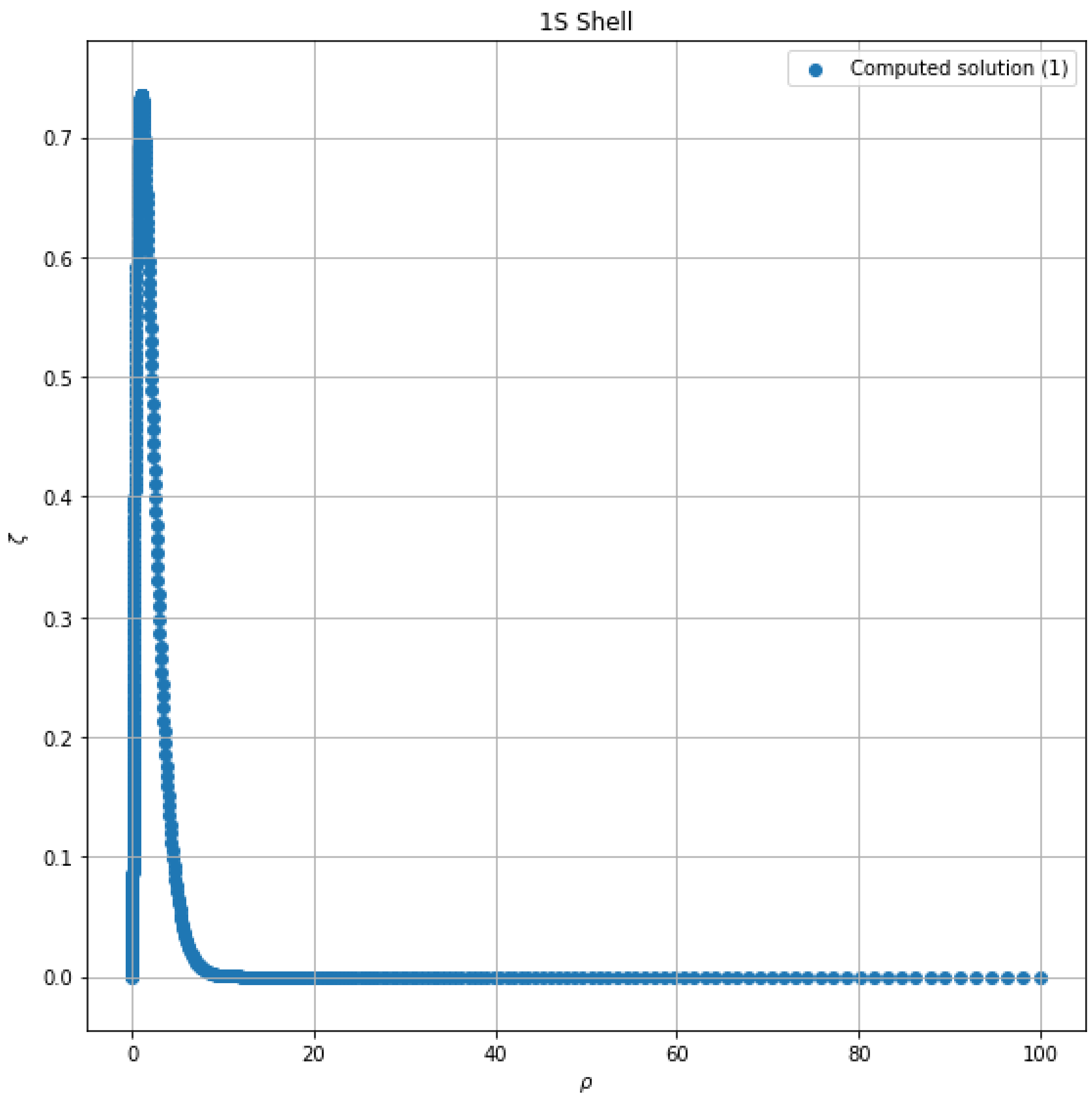
The following grid parameters were determined using trial and error: $a = 10^{-4}$, $h = 0,13351$, $N = 749$ and $\tau_1 = 10^{-4}$. A comparison between the analytical energy eigenvalues and the converged eigenvalues are given below. We are certain that these values are converged, otherwise the script would not finish executing due to not reaching the required accuracy in F of 10^{-10}. As we can see from the table, the computed eigenvalues compare rather well to the analytic solution. We can also see that the correct amount of degeneracies are present in the computed eigenvalues, but this is to be expected giving the fact that the computed eigenvalues compare rather well to the analytic solutions.

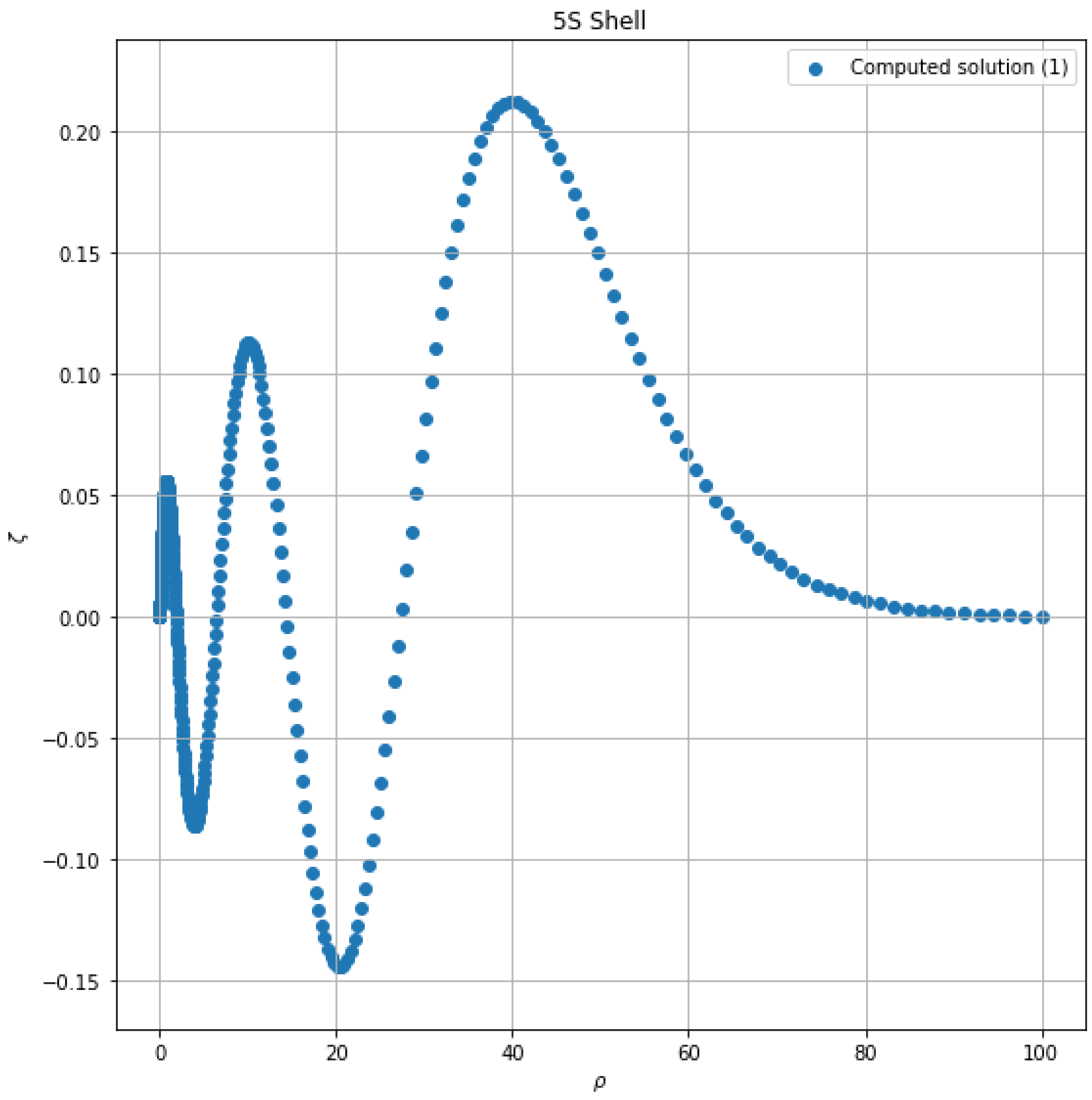
Shell	$\lambda_{analytic}$	λ_{guess}	$\lambda_{computed}$
1S	-1,0000	-1,3000	-1,0000999
2S	-0,2500	-0,3000	-0,2500316
3S	-0,1111	-0,1500	-0,1111395
4S	-0,0625	-0,0600	-0,0625281
5S	-0,0400	-0,0500	-0,0400281
2P	-0,2500	-0,3000	-0,2500065
3P	-0,1111	-0,2000	-0,1111279
4P	-0,0625	-0,0600	-0,0625214
5P	-0,0400	-0,0500	-0,0400237
3D	-0,1111	-0,1200	-0,1111157
4D	-0,0625	-0,0600	-0,0625127
5D	-0,0400	-0,0500	-0,0400175

The plots of the 1S and 5S wavefunctions can be found below. As previously discussed, the logarithmic grid ensures that the most important and interesting part of the wavefunction is described with the most accuracy. This is most prominent in the 1S case, where all the action happens between $0 \leq \rho \leq 10$. If we used a linear grid here, the accuracy would be much lower or we would need more grid points (and thus more computation time) to obtain the same accuracy as we have now.

In [12]:
```
lambda_guess = -1.3
l = 0

# 1S Shell
grid = Grid(750, start=np.log(1E-4), end=np.log(100), type=GridType.L
OG, t1=1E-4)
pot = CoulombPotentialLog()
solver = Solver(grid, pot, error=1E-10, l=l)

lambda_, f = solver.search(lambda_guess, damping=0.75)
print(f"Eigenvalue 1S shell = {lambda_}")

plot(np.exp(grid), None, solver.wf * np.sqrt(np.exp(grid) + grid.t1),
scatter=True, title="1S Shell")

# 5S shell
lambda_guess = -0.05
solver = Solver(grid, pot, error=1E-10, l=l)

lambda_, f = solver.search(lambda_guess, damping=0.75)
print(f"Eigenvalue 5S shell = {lambda_}")

plot(np.exp(grid), None, solver.wf * np.sqrt(np.exp(grid) + grid.t1),
scatter=True, title="5S Shell")
```

```
Eigenvalue 1S shell = -1.0000998755974149
Eigenvalue 5S shell = -0.04002808290707692
```




5S Shell
Computed solution (1)
ζ
ρ
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
0
20
40
60
80
100

b)

Let us start of with calculating the relevant quantities.

$$\mu = \frac{m_+ m_-}{m_+ + m_-} = 0,36 m_0$$

$$R_x = R_H \frac{\mu \epsilon_0^2}{m \epsilon^2} = 0,0875\text{eV}$$

$$r_0 = \frac{\hbar}{\sqrt{2\mu V_0}} = 1,10\text{nm}$$

Given that in a), the 5S wave resides in $0 \leq \rho \leq 100$, we can determine that $r = r_0 \rho = 110\text{nm}$. In order to compare the energy eigenvalue in eV, we must convert it to eV and add the band gap energy to this value as well to get the correct values. This means $E = E_g + E_n = E_g + \lambda_n R_x$. The band gap has been given as $E_g = 2,17202\text{eV}$. The comparison of energies between the analytical and computed solutions is given in the table below.

Shell	$E_{analytic}$ **(eV)**	$E_{computed}$ **(eV)**
1S	2,0845	2,0845112
2S	2,1501	2,1501422
3S	2,1623	2,1622953
4S	2,1666	2,1665488
5S	2,1685	2,1685175
2P	2,1501	2,1501444
3P	2,1623	2,1622963
4P	2,1666	2,1665494
5P	2,1685	2,1676429
3D	2,1623	2,1622974
4D	2,1666	2,1665501
5D	2,1685	2,1685185

Whilst the accuracy of the computed values is quite high and the values themselves are very comparable to the analytical values, the most prominent difference is that the degeneracy visible in the analytical energies is missing in the computed energies. This means that either the Coulomb potential cannot describe the system accurately enough, or the accuracy of the computed data is simply not high enough.

```
In [ ]:
```