STFC Summer School – Python projects

Alex Gezerlis

August 2026

When producing your solutions, you may find some of the codes at
`https://numphyspy.org/codes/`
helpful as building blocks.

1. The *theory of S-wave scattering* is typically covered near the end of a course on quantum mechanics; far from being an "advanced" topic, it is absolutely crucial to the connection of subatomic theory with experiment. Courses usually limit themselves to analytically solvable scenarios (e.g., hard-sphere scattering); this may leave students unsure how to generalize things. In what follows, we will discuss a spherically symmetric short-range potential, $V(r)$; in the center-of-momentum frame, employing the relative coordinate r, the relevant differential equation takes the form:
$$-\frac{\hbar^2}{2\mu}\frac{d^2u(r)}{dr^2}+V(r)u(r)=Eu(r) \tag{1}$$
where $\mu = m_0 m_1/(m_0+m_1)$ is the reduced mass (and the scattering is between two particles of mass m_0 and m_1). Here $u(r)$ is intimately connected to the (radial part of the) scattering wave function $\psi(r) = u(r)/r$. It is conventional to express the energy in terms of the relative wave number, $E = \hbar^2 k^2/(2\mu)$.

   Based on how you probably first learned quantum mechanics, you may be thinking that this is an eigenvalue problem for which only discrete energies are allowed. However, what distinguishes scattering from bound-state problems is precisely the fact that the present problem has a solution for any (positive) value of E. Similarly, while the initial value $u(0) = 0$ (ensuring regularity at the origin) is something that you encounter for both scattering and bound-state problems, the situation regarding the boundary condition at some distance R (taken to be outside the range of the potential) is different: for bound-state problems the wave function rapidly goes to zero when $V(r) = 0$ whereas, in the present case, when the potential vanishes the wave function becomes a linear combination of the non-interacting solutions, which can be expressed as:
$$u(R) = A\sin\left(kR+\delta_0\right) \tag{2}$$
Here A is a normalization constant and $\delta_0 = \delta_0(k)$ is the S-wave *phase shift*.

   Practically speaking, the situation we are now faced with is quite different from what one is faced with when tackling a discrete-energy QM problem (where one guesses the starting derivative, not caring about the normalization, and then combines an initial-value-problem (IVP) solver with a root-finder to compute the eigenvalue). Here, any E (or equivalently k) will work, so it's not clear how one should go about matching the numerical solution of Eq. (1) with the analytically determined Eq. (2), since we still don't know the phase shift δ_0 in Eq. (2). To put it differently: our task is precisely the determination of the phase shift δ_0 for a given (input) wave number k.

   The way forward is to realize that we will be tackling the IVP in Eq. (1) using RK4, by setting up the second-order differential equation as two coupled first-order differential equations. This

means that when we arrive at $r = R$ we will not only have (numerical) access to $u(R)$ but also to $u'(R)$. On the analytical front, differentiating Eq. (2) immediately gives:[1]

$$u'(R) = Ak\cos(kR + \delta_0) \tag{3}$$

Combining Eq. (2) and Eq. (3) we get:

$$k\frac{u(R)}{u'(R)} = \tan(kR + \delta_0) \tag{4}$$

Our strategy is now clear: we will numerically solve Eq. (1), with $u(0) = 0$ and an immaterial guess for $u'(0)$, and then plug the values of $u(R)$ and $u'(R)$ into Eq. (4) to compute δ_0.[2]

Let us specialize to (low-energy) *neutron-neutron scattering*. This means we can take:

$$V(r) = -c\frac{\hbar^2}{2\mu}\frac{\beta^2}{\cosh^2(\beta r)} \tag{5}$$

This potential is of the modified Pöschl–Teller form; you can use $c = 1.814164$ and $\beta = 0.79959$ fm^{-1}. Since the potential range is a few fm, you can take $R = 15$ fm. Plot the phase shift δ_0 (measured in degrees) for $k = 0.01, \ldots, 2.0$ fm^{-1}. You should make sure to use the fact that the phase shift will here always be positive to shift the output of your inverse-tangent function (thereby ensuring that it looks continuous).

2. One phenomenological approach to nuclear binding energies involves thinking of the nucleus as being an incompressible, charged *liquid drop*. Appropriately generalized, this gives rise to the *semi-empirical mass formula* for the binding energy per nucleon:

$$\frac{E_B}{N+Z} = a_V - \frac{a_S}{(N+Z)^{1/3}} - a_C\frac{Z(Z-1)}{(N+Z)^{4/3}} - a_{sym}\frac{(N-Z)^2}{(N+Z)^2} - \lambda\frac{a_p}{(N+Z)^{3/2}} \tag{6}$$

Here N is the number of neutrons and Z is the number of protons in a given nucleus whose binding energy is E_B. The structure of each term is motivated by physical considerations; the coefficients are, accordingly, labelled to correspond to the volume, surface, Coulomb, symmetry, and pairing terms. The latter is not always present, since the λ parameter takes the values:

$$\lambda = \begin{cases} +1, & \text{odd } N\text{-odd } Z \\ 0, & \text{odd } N\text{-even } Z, \text{ or even } N\text{-odd } Z \\ -1, & \text{even } N\text{-even } Z \end{cases} \tag{7}$$

In the file `AME2003_data.txt` you will find an appropriately pruned version of results from the AME2003 atomic mass evaluation; the three columns are the N, Z, and $E_B/(N+Z)$ values, respectively. Your task is to fit the model in Eq. (6) to the data, thereby determining the a_V, a_S, a_C, a_{sym}, and a_p coefficients. Specifically:

(a) Observe that you will need to carry out *linear* least-squares fitting; the dependence on N and Z is clearly nonlinear, but the dependence on the a's is just as clearly linear. A couple of new features jump out when you think of how to generalize `normalfit.py` so it can handle Eq. (6): first, note that $E_B = E_B(N, Z)$, i.e., we are faced with *two*

[1] This is a bit sloppy. More properly, we first differentiate $u(r)$ with respect to r and then set $r = R$.

[2] In the literature, this idea is known as the matching of the logarithmic derivative. Note that the normalization constant cancels when you form the quotient in Eq. (4).

independent variables. It may help you to think of the approximating function of Eq. (6) as being a function of a vector, i.e., $p(\mathbf{x}) = E_B^{\rm model}/(N+Z)$; in the present case this $\mathbf{x}$ would be made up of two numbers (N and Z). Second, you will have to figure out how to programmatically encode λ.

(b) Note that the input data table that we provided does not come with associated error bars. Your first instinct may be to take all the σ_j's to be equal to one. However, it should be easy to see that this would treat $\sigma_j = 1$ as an absolute error (but what if the quantities you were trying to describe had a magnitude of, say, 10^{-3}?). One way to proceed is to assume that all error bars are all equal (so $\sigma_j = \sigma$) and that you did a good job fitting (so $\chi^2 \approx n-m$, where n is the number of data points and m the number of parameters). Practically speaking, this means that you can set $\sigma_j = 1$ in your code, produce a value of $\chi^2/(n-m)$ as per:

$$\chi^2 = \sum_{j=0}^{n-1} \left(\frac{y_j - p(\mathbf{x}_j)}{\sigma_j} \right)^2 \tag{8}$$

where $y_j = E_B^{\rm data}/(N+Z)$. Similarly, you can get the standard deviation in each parameter, from:

$$\sigma_a^2 = \sum_{j=0}^{n-1} \left(\frac{\partial a}{\partial y_j} \right)^2 \sigma_j^2 \tag{9}$$

Finally, rescale σ_a by multiplying it with $\sqrt{\chi^2/(n-m)}$. (If you don't remember much about the variance-covariance matrix, simply use `scipy.optimize.curve_fit()` to carry out the fit, passing in `absolute_sigma=False`.)

(c) In the same figure, plot the input binding energies per nucleon ($E_B^{\rm data}/(N+Z)$) and the binding energies per nucleon corresponding to the a_V, a_S, a_C, a_{sym}, and a_p values you determined earlier in this problem ($E_B^{\rm model}/(N+Z)$). In both cases, your x axis should be $N+Z$ and your y axis should be $E_B/(N+Z)$.

(d) Plot the residuals in your predictions for the binding energies. That means that your x axis should be $N+Z$ and your y axis should be $E_B^{\rm data} - E_B^{\rm model}$. Observe that the residuals exhibit systematic trends at given $N+Z$ values, reflecting the absence of shell corrections in Eq. (6).

3. We now turn to the problem of determining *stellar structure* by solving the Tolman–Oppenheimer–Volkoff (TOV) equations:

$$\frac{dP(r)}{dr} = -\frac{Gm(r)\rho(r)}{r^2}\left[1+\frac{P(r)}{\rho(r)c^2}\right]\left[1+\frac{4\pi r^3 P(r)}{m(r)c^2}\right]\left[1-\frac{2Gm(r)}{c^2 r}\right]^{-1} \tag{10}$$

Here, G is Newton's gravitational constant, r is the radial coordinate, and $m(r)$ is the mass inside a sphere of radius r, which satisfies:

$$\frac{dm(r)}{dr} = 4\pi r^2 \rho(r) \tag{11}$$

Note that in the last two relations P, ρ, and m are functions of only r; as a result we are dealing with a pair of *ordinary* differential equations. From the equation of state, given $\rho(r)$ you can determine $P(\rho(r))$, meaning that Eq. (10) and Eq. (11) are two first-order differential

equations for $\rho(r)$ and $m(r)$; crucially, these cannot be solved independently of each other, since both of them involve $m(r)$ and $\rho(r)$; we say that these are *simultaneous differential equations* (or *coupled* differential equations).

One way to proceed is to use the equation of state, giving us $P(\rho)$, and convert Eq. (10) such that it involves a derivative of $\rho(r)$. To be explicit, this means using:

$$\frac{dP(r)}{dr} = \left(\frac{dP(\rho)}{d\rho}\right)\left(\frac{d\rho}{dr}\right) \tag{12}$$

and then keeping only $d\rho/dr$ on the left-hand side of the equation. This is the natural approach to take if you're employing an equation of state in the form $P(\rho)$.

Here we will explore an alternative scenario, where you have been given the equation of state in the form $\rho(P)$. (Obviously, in trivial cases one can simply invert to go from one formulation to the other.) Thus, we will end up with versions of Eq. (10) and Eq. (11) in which both the left-hand sides and the right-hand sides involve only $P(r)$ and $m(r)$.

(a) Instead of getting lost in a sea of units and factors of c^2, it is convenient to make things dimensionless as follows:

$$m = M_\odot \overline{m}, \quad r = R_S \overline{r}, \quad P = \gamma \overline{P} \tag{13}$$

where $M_\odot = 1.989 \times 10^{30}$ kg is the mass of our sun, $R_S = 2GM_\odot/c^2 = 2.954$ km is the corresponding Schwarzschild radius, and γ will be chosen below to make our lives easier. Crucially, $\overline{m}$, $\overline{r}$, and $\overline{P}$ are dimensionless. Think about the units of P, γ, and ρ and find a way to produce a dimensionless mass density, $\overline{\rho}$, by re-using γ (possibly appropriately rescaled).

Analytically show that when they are rewritten in terms of the above dimensionless quantities, Eq. (10) and Eq. (11) take the form:

$$\begin{aligned} \frac{d\overline{P}}{d\overline{r}} &= -\frac{\overline{m}\,\overline{\rho}}{2}\left[1+\frac{\overline{P}}{\overline{\rho}}\right]\left[1+\frac{4\pi\overline{r}^3\overline{P}}{\overline{m}}\frac{\gamma R_S^3}{M_\odot c^2}\right]\left[\frac{1}{\overline{r}^2-\overline{m}\,\overline{r}}\right] \\ \frac{d\overline{m}}{d\overline{r}} &= 4\pi\overline{r}^2\overline{\rho}\frac{\gamma R_S^3}{M_\odot c^2} \end{aligned} \tag{14}$$

Notice that both equations involve $\gamma R_S^3/(M_\odot c^2)$: if you now take $\gamma = M_\odot c^2/R_S^3$ (as you are free to do), then that term goes away and both equations look particularly simple.

(b) We now prepare to solve our two simultaneous ODEs numerically. Since there is a numerical singularity at $r = 0$, you should start the integration of the TOV equations at a small (but finite) r_Δ. You can analytically integrate Eq. (11) from 0 to r_Δ to find:

$$m(r_\Delta) = \frac{4}{3}\pi r_\Delta^3 \rho_c \tag{15}$$

which can, in turn, be expressed in terms of the dimensionless quantities. The only missing piece of the puzzle is the equation of state. Here we employ the following phenomenological form:

$$\overline{\rho} = 0.871\overline{P}^{3/5} + 2.867\overline{P} \tag{16}$$

which has been tailored to capture both non-relativistic and relativistic microphysics (in the first and second terms, respectively) for a neutron star. It's probably also wise to

programmatically account for the, numerical, possibility of negative pressure; make sure your code produces a tiny positive $\overline{\rho}$ in that case.

Guess a value of the dimensionless central pressure $\overline{P}_c = 0.01$, use it to get $\overline{\rho}_c$, and from there compute $\overline{m}(\overline{r}_\Delta)$; then, use RK4 to solve Eq. (14). Observe that you don't actually need to know the numerical values of $M_\odot$, R_S, and γ to do this: you will simply find the total radius $\overline{R}$ and total mass of the star $\overline{M}$ (determined when the pressure goes to zero) as dimensionless quantitities.

Note a further complication: in the language of RK4, we don't know ahead of time up to where we wish to integrate, i.e., we don't know which b to pass in. You should employ two different strategies: (i) guess a large b value and keep only that portion of the solution that corresponds to positive pressures that are larger than some threshold, and (ii) treat this as a boundary-value problem, where you know $\overline{P}(0) = \overline{P}_c$ and $\overline{P}(\overline{R}) = 0$; use a root-finder to determine $\overline{R}$. In both approaches, your end goal is to compute the mass $\overline{M}$ (in units of the solar mass, $M_\odot$) and the radius R (in km).

(c) Repeat the above calculation for a whole range of initial $\overline{P}_c$ values, thereby producing a *mass-radius plot* with the total radius in the x axis and the total mass in the y axis. Discuss the plot's qualitative features. Hint: you should be careful when choosing the number of integration points, n.