GRADING RUBRIC

- Not all problems are the same difficulty - the solutions to easier ones will be judged more harshly than those for harder ones!
- The difficulty of a problem is uncorrelated, or perhaps even anticorrelated, with its length.
- Bonus points for each non-trivial tool written on the board (i.e. how to diagonalize, integration routines, plotting, helpful code snippets, useful libraries etc...)
- Each code should be able to run exclusively by calling it with inputs - no changing anything in the code.
- The more modularized and extendable the code is, the better.
- Parallelized programs will receive bonus points.
- Bonus points for best execution times and memory usage!
- Bonus points for a GPU implementation for linear algebra.
- Bonus points for every mistake you find in the problem statements.

TIPS AND RECOMMENDATIONS

- Decide in advance how to modularize your code and split the work among group members, i.e. plotting, input output, integration, matrices, etc...
- Use functions and custom types to organize code.
- Write general code and use multiple dispatch to optimize specific cases.
- Don't forget about abstract types, to help build your own type hierarchy.
- Use all available resources: multicore CPUs, GPUs.
- There are many plotting libraries in Julia, and it can be hard to chose one. Črt recommends Makie and using https://juliadatascience.io/ as a guide.

PROBLEM SUMMARY

- Problem one is an atomic physics-themed problem investigating the structure of alkali atoms. A successful code will involve using Special functions, numerical integration, matrix manipulation, diagonalization.
- Problem two is a condensed matter-themed problem investigating Anderson localization. A successful code will involve handling large amounts of data, random number generation, diagonalization of large systems.
- Problem three is a condensed matter-themed problem investigating topology. A successful code will involve matrix manipulation and diagonalization, numerical differentiation and integration, complex numbers.

RYDBERG SPECTRA OF ALKALI ATOMS (AND MUCH MORE)

Goal: compute the following quantities for matt's favorite atom (he might choose lithium, sodium, potassium, or rubidium): bound state spectrum, eigenfunctions, ground state polarizability

Start by solving the radial Schrödinger equation (using atomic units),

$$0 = \left(-\frac{1}{2}\frac{d^2}{dr^2} + V(r) + \frac{l(l+1)}{2r^2} - E_{nl}\right) u_{nl}(r), \tag{1}$$

for the electronic energies E_{nl} and eigenstates $\Psi_{nlm}(\vec{r}) = \frac{u_{nl}(r)}{r} Y_{lm}(\hat{r})$. Here, $Y_{lm}(\hat{r})$ are the spherical harmonics, l and m are the orbital angular momentum quantum numbers, and n is the principal quantum number. For the potential $V(r)$ we use the model potentials given in "Dispersion coefficients for alkali-metal dimers M. Marinescu, H. R. Sadeghpour, and A. Dalgarno Phys. Rev. A 49, 982 (1994)." Specifically, these have the form

$$V(r) = -\frac{Z_l(r)}{r} - \frac{\alpha_c}{2r^4}\left(1 - e^{-(r/r_c)^6}\right), \tag{2}$$

where

$$Z_l(r) = 1 + (z-1)e^{-a_1 r} - r(a_3 + a_4 r)e^{-a_2 r}. \tag{3}$$

The coefficients for these model potentials, for Li, Na, K, and Rb, are given in the table provided on github, organized like Table I in the Marinescu et al paper referenced above. Your code should import this data file (don't just type it in! you will add typos!). z is the nuclear charge, which is 1 for H, 3 for Li, 11 for Na, 19 for K, and 37 for Rb. Of course, for hydrogen the potential is just $V(r) = -1/r$.

An excellent method for solving Eq. 1 is by expanding $u_{nl}(r)$ in a basis of *Sturmian* functions, non-orthogonal basis states are similar in many respects to Coulomb wave functions. They are defined using

$$S_{kl}(r) = \sqrt{\frac{k!}{2(2l+1+k)!}} e^{-qr}(2qr)^{l+1} L_k^{2l+1}(2qr), \tag{4}$$

where l should be the same as the angular momentum of the states you wish to calculate and $0 \le k \le \infty$ is the index of the Sturmian. L_k^{2l+1} is a generalized Laguerre polynomial. You should choose a reasonably sized maximum value of k which gives converged results (test it!) but is not so high that numerical instabilities result (test it!!). **bonus: use higher precision computing to test if this increases the stability for large basis sizes**. The parameter q can be tuned to accelerate convergence, and a value around $q = 0.7$ seems to give good results for the problem at hand. If you get different converged results for different q values (e.g. $q = 1.5$) this is probably a good sign that your code is broken and that you have problems!!).

You will need to compute matrix elements of the Hamiltonian in this basis, turning Eq. 1 into a matrix equation, to obtain the eigenvalues E_{nl}. What this means is that you write $u_{nl}(r) = \sum_k c_k S_{kl}(r)$, insert this into Eq. 1, multiply by $S_{k'l}(r)$ from the left, and integrate to convert everything into a matrix equation.

To be specific, the matrix equation is

$$H_{ii'}c_{i'} = EO_{ii'}c_{i'} \tag{5}$$

where

$$O_{ii'} = \int_0^\infty S_{kl}(r) S_{k'l}(r) dr \tag{6}$$

$$H_{ii'} = T_{ii'} + V_{ii'} \tag{7}$$

$$T_{ii'} = -\frac{1}{2}\int_0^\infty S_{kl}(r)\frac{d^2}{dr^2} S_{k'l}(r) dr \tag{8}$$

$$V_{ii'} = \int_0^\infty S_{kl}(r)\left[V(r) + \frac{l(l+1)}{2r^2}\right] S_{k'l}(r) dr \tag{9}$$

Hints:

- the change of variable $r = x^2$ is very useful to improve the convergence of the numerical integration.

- Don't forget that you will need to solve a generalized eigenvalue equation in order to account for the non-orthogonality of these states.
- Use Green's theorem to express $-\frac{1}{2}\int S_{nl}(r)\frac{d^2}{dr^2}S_{n'l}(r)dr = \frac{1}{2}\int S'_{nl}(r)S'_{n'l}(r)dr$. You can either symbolically calculate these derivatives in a computer algebra program such as Mathematica (or try to do this in Julia for bonus points!), or just do these derivatives numerically.
- The overlap matrix should be sparse; confirm this on a small matrix and then use this feature to accelerate the computation.
- you can check that everything is working by comparing with analytic results for hydrogen, using $V(r) = -1/r$, i.e. that the energy levels obey the Rydberg formula $E_{nl} = -1/(2n^2)$.
- To check if it works: visit the NIST atomic energy level database and compare with the values there, converting from cm^{-1} to atomic units via

$$0.5 \text{ (atomic units)} = 109737.3 * 2\mathrm{cm}^{-1}. \tag{10}$$

Plot a few of the resulting eigenstates for the lowest bound states of the atom. **Ask Matt for help or reassurance at any point!! there are many ways to make mistakes and get nonsense!!!**

Hint: a successful computation of the energy levels of lithium, for example, should give you (in atomic units): -1.87852, -0.198134, -0.0741789.... Notice that the first energy is very deeply bound. This is in fact a (somewhat poor) value for the energy of the 1s electronic state, when in fact we are interested in the energy levels just of the outer valence electron which for lithium is of course the $2s$ state. Therefore you should neglect this first value from your calculation (etc, *mutatis mutandis*, for other atoms and other angular momenta.) Of course, the lowest allowed p state is $2p$, so the lowest eigenvalue computed for that symmetry will be the correct energy of the $2p$ state. Be careful!

For as many bound states as you trust to be accurate, plot the quantity $\frac{1}{\sqrt{-2E_{nl}}}$ as a function of n and fit this to a line. Congratulations!!!, you have computed the quantum defects μ_l such that the atomic energies are described by the Rydberg formula $E_n = -\frac{1}{2(n-\mu_l)^2}$.

Finally, compute the ground state polarizability,

$$\alpha = 2 \sum\!\!\!\!\!\!\int \frac{|\langle n_0 00|\vec{r}|n10\rangle|^2}{E_n - E_1} = \frac{2}{3} \sum\!\!\!\!\!\!\int \frac{1}{E_n - E_1}\left[\int_0^\infty u_{n_0 0}(r) r u_{n1} dr\right]^2, \tag{11}$$

where $\sum\!\!\!\!\!\!\int$ denotes a sum over all excited bound states n and an integral over continuum states up to infinite energy. Compute the static polarizability by evaluating the radial integrals for the bound states you calculated (remember, these are expressed as linear combinations of Sturmians) and performing the sum in Eq. 11. Since you only have computed the discrete spectrum of the atom, this calculation is missing any continuum contributions. Be careful that you identify the correct ground state and excited states to include (ask Matt for help if needed - for example $n_0 = 2$ for Li and $n_0 = 1$ for hydrogen).

An elegant way to include this missing continuum contribution without having to evaluate the infinite sum and integral over continuum states is to use the famous *Dalgarno-Lewis* method. You can compute

$$\alpha = \frac{2}{3}\int u_{n_0 0}(r) r \Phi(r) dr \tag{12}$$

where $\Phi(r)$ is a solution of the inhomogenous differential equation

$$-\frac{1}{2}\Phi''(r) + \left(\frac{1(1+1)}{2r^2} + \frac{Z(r)}{r} - E_{n_0 0}\right)\Phi(r) = r u_{n_0 0}(r). \tag{13}$$

You can solve for $\Phi(r)$ in the same way as you solved for $u_{nl}(r)$ above, by expanding it into the Sturmian basis and turning this into a matrix equation, since $u_{n_0 0}(r)$ is represented also in this basis using the eigenvectors calculated above. Of course, since this is an inhomogenous equation, you will need to solve the resulting linear system to obtain a solution vector rather than diagonalizing it to obtain eigenstates.

Bonus: For hydrogen, this should give the exact answer of $\alpha = 9/2$ (matt will ask if it does!!), whereas the sum evaluated above is a severe underestimate. The continuum is much more important for hydrogen than for other alkali atoms! Give a cool answer for why this is for many bonus points!

Bonus: Give the user the option of doing this calculation by discretizing the second derivative. To do this, create a grid of r values from 0 to some r_0 larger than the size of the desired wave functions (some convergence checks may be necessary) in steps of $h \ll 1$. This turns the solution of Eq. 1 into diagonalization of the matrix

$$-\frac{1}{2h^2}T_{ij} + V_{ii} \tag{14}$$

where $T_{ij} = 1$ if $|i - j| = 1$, $T_{ij} = -2$ if $i = j$, and $T_{ij} = 0$ otherwise. V_{ii} is a diagonal matrix of the potential evaluated at each r value in the grid. **Hint: use sparse arrays.**

Bonus 2: give the user the option of using a quadratic grid, which converges much more rapidly than the linear grid. For the matrix elements of the second derivative operator in this case, see Eq. 9 of F Robicheaux 2012 J. Phys. B: At. Mol. Opt. Phys. 45 135007

ANDERSON LOCALIZATION

Goal: study the properties of a disorder tight-binding Hamiltonian, drawing conclusions about localization by computing participation ratios and level statistics. Start with the problem of a particle moving in a 1D tight-binding lattice of N sites, as described by the Hamiltonian

$$H = \sum_{n}^{N} E_n |n\rangle\langle n| + \sum_{n,n'}^{N} V_{nn'} |n\rangle\langle n'|. \tag{15}$$

Compute the eigenspectrum ϵ_ν of this Hamiltonian for constant on-site potentials $E_n = E$ and nearest-neighbor interactions $V_{nn'} = V, |n - n'| = 1$. The eigenstates, $|\psi\rangle_\nu = \sum_n c_n^\nu |n\rangle$, should be extended over the whole lattice - check this by plotting the coefficients $|c_n^\nu|^2$. Does the result change if periodic boundaries are included (i.e., $|N+1\rangle = |1\rangle$)?

Add disorder to this model, either by drawing the diagonal elements E_n from a random distribution with variance σ or by letting the hopping amplitudes $V_{nn'}$ vary. As the disorder increases, you should start to see localization in the eigenstates, as can be verified by again plotting $|c_n^\nu|^2$. Study how this localization happens in this system as a function of both N and σ, making sure to appropriately average over disorder realizations. A typical quantity characterizing localization is the *inverse participation ratio* $I_\nu = \sum_n |c_n^\nu|^4$. For a localized states, $c_n^\nu \to \delta_{nn'}$ and $I_\nu = 1$. For an extended state, $c_n^\nu \to \frac{1}{\sqrt{N}}$ and $I_\nu = \frac{1}{N^2}$. A plot of I_ν as N and σ vary will therefore tell us a lot about the localization of this system.

Another useful quantity is the AGR,

$$AGR = \left\langle \frac{\min(s_\nu, s_{\nu-1})}{\max(s_\nu, s_{\nu-1})} \right\rangle, \tag{16}$$

where $s_\nu = \epsilon_\nu - \epsilon_{\nu-1}$ and $\langle\rangle$ denotes an average over the entire spectrum and disorder realizations. Plot this quantity as a function of N and σ.

Now, extend your code to let the user choose from a variety of hopping options (**Bonus: let the user input this as an arbitrary function**). Power law interactions of the form $V_{nn'} = |n - n'|^{-d}$ are particularly interesting; do a parameter scan to see what happens to the localization, characterized by IPR and AGR, as a function of d. You could also look at models with all-to-all hoppings. Do this for periodic boundary conditions as well, which is NOT TRIVIAL TO DO.

Hints: this is a rather open-ended assignment. Take advantage of that freedom to explore and generate your own results, otherwise you will lose points! Can you extend to other lattices? Maybe you can implement the Hamiltonian of problem 3 and see how its eigenspectrum changes under disorder?

TOPOLOGICAL PHASES OF A MATT LATTICE

Goal: calculate the band spectrum of the Matt Lattice, identify the parameters where edge states exist, and calculate the Zak phase for these parameters to determine if these edge states are topologically protected.

Consider the following matt lattice (mattice):

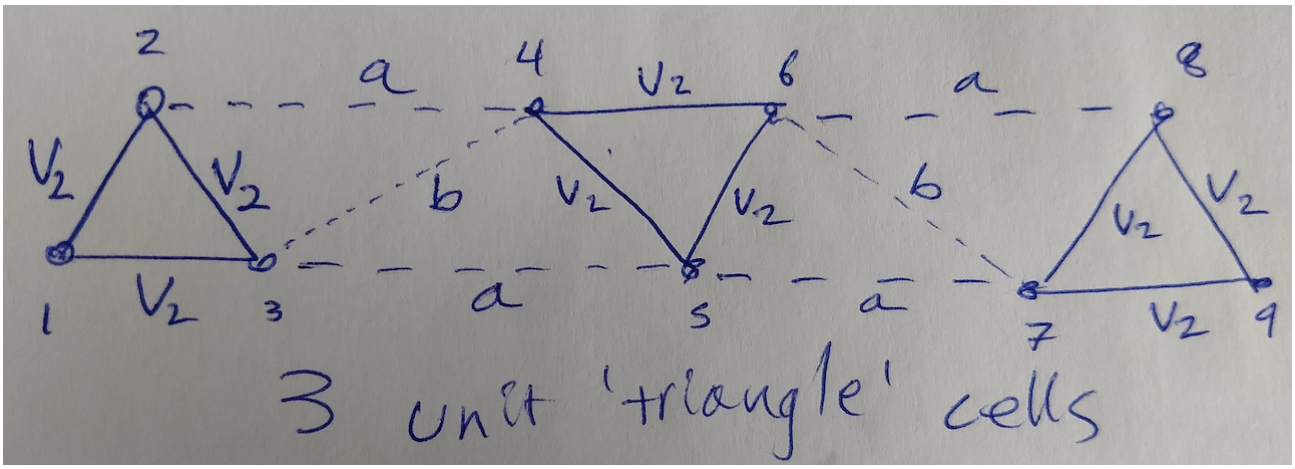

FIG. 1. the mattice

The Mattice consists of triangle unit cells with intra-trimer couplings v_2, 'horizontal' couplings a, and 'diagonal' couplings b. The Hamiltonian matrix (shown here for an example with 3 unit cells) is

$$H = \begin{pmatrix} o & v_2 & v_2 & 0 & 0 & 0 & 0 & 0 & 0 \\ v_2 & o & v_2 & a & 0 & 0 & 0 & 0 & 0 \\ v_2 & v_2 & o & b & a & 0 & 0 & 0 & 0 \\ 0 & a & b & o & v_2 & v_2 & 0 & 0 & 0 \\ 0 & 0 & a & v_2 & o & v_2 & a & 0 & 0 \\ 0 & 0 & 0 & v_2 & v_2 & o & b & a & 0 \\ 0 & 0 & 0 & 0 & a & b & o & v_2 & v_2 \\ 0 & 0 & 0 & 0 & 0 & a & v_2 & o & v_2 \\ 0 & 0 & 0 & 0 & 0 & 0 & v_2 & v_2 & o \end{pmatrix}. \tag{17}$$

The on-site potential, o, is physically irrelevant, but it is helpful for making sure that the band structure is sorted numerically as you will want it to be. I would suggest setting $o = -100$, and then adding 100 to your eigenvalues later to center the results around zero energy. Make sure your code can generate a matrix H for arbitrary N unit cells.

Your first goal is to calculate the eigenvalues of this Hamiltonian. Your code should be able to do this for an input N, v_2, a, and b of the user's choice. You should also be able to produce plots of the eigenvalues as a function of either a or b, for fixed values of the other parameters. It's also useful to just have a plot of the eigenvalues as a function of an index in order to see degeneracies more clearly. We can then investigate these spectra to look for topological edge states and phase transitions.

Ideally, one would like to look at the eigenstate coefficients in the site basis to see if there are edge states in the band gaps. Make sure your code can make the appropriate plots and check if you get edge states. With these you can publish high-impact papers in high-impact journals!

Plot the eigenspectra of this model as a function of a or b for a few different parameter combinations and N values. See what you can conclude from these. You should see that teh eigenstates form three bands. Hopefully, you see that there are, in some parameter regimes, states lying in the gaps between the three different bands which have nearly degenerate energy. The energies should be associated with localized edge eigenstates!!

A working code will allow Matt to input parameters of his choosing to compare with his results, so ask him for help in case you don't understand what he wants!

To characterize this more generally, we need to study the bulk Hamiltonian of this system. Taking periodic boundary

conditions, we can write the same Hamiltonian in the k representation

$$H(k) = \begin{pmatrix} o & v_2 + ae^{-ik} & v_2 + be^{-ik} \\ v_2 + ae^{ik} & o & v_2 + ae^{-ik} \\ v_2 + be^{ik} & v_2 + ae^{ik} & o \end{pmatrix}, \tag{18}$$

where $0 \le k \le 2\pi$. Diagonalizing $H(k)$ gives us the band spectrum as a function of k, namely the three energies $\epsilon_j(k)$ $(1 \le j \le 3)$, and eigenvectors $\vec{\psi}_j(k)$. You should plot some of these band structures!

Since this is an inversion-symmetric system, we can compute the *Zak phase* and use that to characterize the topology of our system as a function of a, b, and v_2. The Zak phase of the jth band is defined (mod 2π) as

$$Z_j = i \int_0^{2\pi} \vec{\psi}_j(k)^\dagger \cdot [\partial_k \vec{\psi}_j(k)]. \tag{19}$$

Our first attempt at calculating the Zak phase is the obvious brute force one, computing the derivative numerically and integrating the dot product numerically.

Hints:

- You can either do the entire derivative and integration numerically on a grid of k values, or learn how to use interpolating functions to do it entirely with those.
- BIG HINT: the eigenvector at a given value $k = k_0$, $\vec{\psi}_j(k_0)$, is is fixed only up to an overall phase factor $e^{i\theta}$. This will cause problems when you try to take the derivative of $\vec{\psi}_j(k)$ because the eigenstates will not be smooth functions of k unless this (physically irrelevant) phase factor is eliminated. An excellent way to do this with complex eigenvectors such as ours is to ensure that the first element of the eigenvector is real, i.e. redefine $\vec{\psi}_j(k) \to \vec{\psi}_j(k)/\psi_{j,1}(k)$, at each k value. Don't forget to normalize your eigenvectors again after this step.
- The Zak phase is defined modulo 2π, so to get sensible results you should include this - which might be nontrivial for numerical values - in your code.
- If everything works you should only get values of $Z_j = 0$ or $Z_j = \pi$ from your calculation. Deviations from these values either suggest a bug in your code or numerical issues.

A much cleverer way to compute the Zak phase eliminates the need for any numerical derivatives. Your code should implement both approaches and let the user select one. This method starts with our old friend Mr. Taylor and his series:

$$f(k + ih) \approx f(k) + ihf'(k), \tag{20}$$

where h is a small parameter. This implies that, if we have a function which depends on a real parameter k, we can let k become complex and obtain its derivative directly from its imaginary part. W0w!

$$f(k) = Re[f(k + ih)] \tag{21}$$

$$f'(k) = Im[f(k + ih)]/h. \tag{22}$$

(thanks to Panos for pointing out this trick) Let's try to do this with our system, which will unfortunately a bit more complicated since we have a *complex Hamiltonian* and cannot so naively analytically continue k. Fortunately, we can write $\tilde{H}$ as a purely real symmetric matrix of twice the dimension of H, such that the first and fourth elements of its eigenvector give the real and imaginary parts of the first element of H's eigenvector, etc. This results in the *purely real* matrix

$$\tilde{H}(k) = \begin{pmatrix} o & v_2 + a\cos k & v_2 + b\cos k & 0 & a\sin k & b\sin k \\ v_2 + a\cos k & o & v_2 + a\cos k & -a\sin k & 0 & a\sin k \\ v_2 + b\cos k & v_2 + a\cos k & o & -b\sin k & -a\sin k & 0 \\ 0 & -a\sin k & -b\sin k & o & v_2 + a\cos k & v_2 + b\cos k \\ a\sin k & 0 & -a\sin k & v_2 + a\cos k & o & v_2 + a\cos k \\ b\sin k & a\sin k & 0 & v_2 + b\cos k & v_2 + a\cos k & o \end{pmatrix}. \tag{23}$$

We then solve the eigenvalue equation

$$\tilde{H}(k + \vec{b}(k + ih) = E\vec{b}(k + ih) \tag{24}$$

by diagonalizing $\tilde{H}$. Let's define, for the jth band (eigenvalue),

$$\vec{c}_j = Re(\vec{b}_j) \tag{25}$$

$$\vec{d}_j = Im(\vec{b}_j)/h. \tag{26}$$

From these we can construct the eigenvector ψ_j as follows. For the mth component, we have

$$C_m = \psi_{j,m}(k) = c_{j,m} + ic_{j,m+3}. \tag{27}$$

(C_m, where the index j is dropped, will be a conveniently short notation for the ugly equations which follow).

We now have the eigenvector and, defining

$$D_m = \psi'_{j,m}(k) = d_{j,m} + id_{j,m+3}, \tag{28}$$

we have its derivative - without taking a single numerical derivative! Of course, h must be a small value and you should take care that you get results which are independent of its exact value, otherwise there are problems!!!

Now, to wrap things up, we have to still do two things. Since we will numerically integrate over k, we still have to remove the phase ambiguity. We also have to make sure that $\vec{C}_j$ is properly normalized. We again divide $\vec{C}_j$ by $C_{j,1}$ to remove the phase ambiguity and normalize the result, getting in the end the normalized eigenstate

$$\vec{\psi}_j \equiv \tilde{C}_m = \frac{C_m}{N},\ N = \sqrt{\vec{C}_m^\dagger \vec{C}_m \frac{C_1}{C_1^*}}. \tag{29}$$

It is this *normalized and smoothly varying* eigenvector that we need to differentiate. We proceed:

$$\begin{aligned}
\partial_k \psi_{j,m}(k) \equiv \tilde{D}_m &= \partial_k \tilde{C}_m = \frac{D_m}{N} - \frac{C_m}{N^3} N' \\
&= D_m/N - \frac{C_m}{2N^3}\left(\vec{C}_m^\dagger \cdot \vec{C}_m \frac{C_1}{C_1^*}\right)' \\
&= D_m/N - \frac{C_m}{2N^3}\left(\left[\vec{D}_m^\dagger \cdot \vec{C}_m + \vec{C}_m^\dagger \cdot \vec{D}_m\right]\frac{C_1}{C_1^*} + (\vec{C}_m^\dagger \cdot \vec{C}_m)\left[\frac{D_1 C_1^* - D_1^* C_1}{(C_1^*)^2}\right]\right),
\end{aligned}$$

where † indicated conjugate-transpose and * indicates complex conjugation. This expression, albeit complicated, can be evaluated using exclusively the terms constructed from the real and imaginary parts of the original eigenvector $\vec{b}$, and no numerical derivatives are necessary. All that you have to do is type it into your code without any typos, not a problem!!!

We can now easily calculate the Zak phase,

$$Z = i\int_0^{2\pi}\left(\sum_m \tilde{C}_m(k)^\dagger \tilde{D}_m(k)\right)\mathrm{d}k. \tag{30}$$

Check that it agrees with your earlier results!

Finally, as talented physicists, we'd oftentimes rather invest our mental energy into avoiding coding as much as possible. We can therefore do some very magimatical derivations to see that the Zak phase can also be calculated using the simple formula below, where ψ_j is evaluated on a grid of k values between 0 and 2π and this notation implies that the product is done over all consecutive pairs of grid points.

$$Z = -\mathrm{Im}\ln\Pi_k \vec{\psi}_j^\dagger(k)\cdot\vec{\psi}_j(k+1), \tag{31}$$

Wow! where did that come from?? Check that this amazing result is consistent with your hard-earned numerical results above. Plot Z_j, calculated from all three methods, as a function of a / b for fixed v_2 and b / a. Ponder what this says about the stability of these methods and of your future career.

Now that you have the Zak phase for each band, compute the Zak phase in the band gaps by summing accordingly. The Zak phase of the lower band gap is Z_1 mod 2π; the Zak phase of the upper band gap is Z_1+Z_2 mod 2π. **Objective: Your code should compute everything mentioned above for arbitrary N, a, b, and v_2: the band structure and Zak phases. It should have functionality to plot the bands as a function of parameters a and b and to investigate edge states based on the resulting bands. Finally, a successful code will map out a TOPOLOGICAL PHASE DIAGRAM by plotting the Zak phase of the upper and lower band gaps as a function of a and b. This should reproduce (for $v_2 = -5.5$) the following figure...**

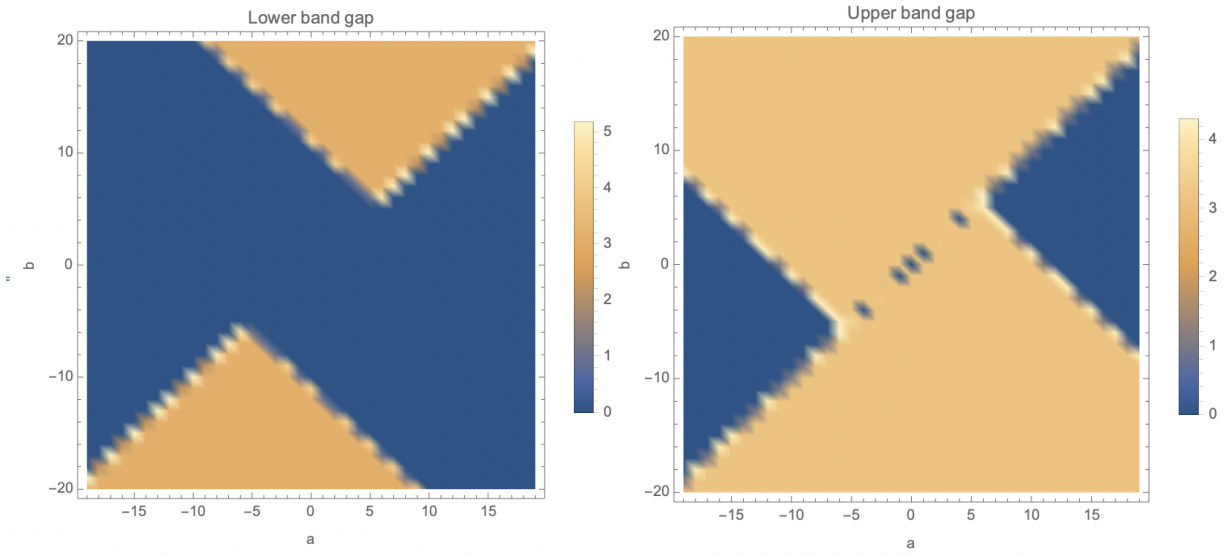

FIG. 2. Phase diagrams of the mattice. Yellow regions have $Z = \pi$ and thus topologically protected edge states; blue regions have $Z = 0$. Topological phase transitions can occur along the lines $a = b$ and $b = \pm 2v_2 - a$.