Error control in scientific modelling (MATH 500, Herbst)

Project 2: Band structures with guaranteed error bars

To be handed in via moodle by 12.01.2024

```
begin
    using Brillouin
    using DFTK
    using LinearAlgebra
    using Plots
    using PlutoTeachingTools
    using PlutoUI
    using Printf
    using LaTeXStrings
    using Statistics
    using Measurements
    using DoubleFloats

end
```

```
Code paths for generic floating-point types activated in DFTK. Remember to add 'u
sing GenericLinearAlgebra' to your user script. See https://docs.dftk.org/stable/
examples/arbitrary_floattype/ for details.
```

Selection deleted

Table of Contents

Selection deleted

Introduction

In this project we will compute approximations of parts of the eigenspectra of a specific class of Hamiltonians

$$H = -\frac{1}{2}\Delta + V.$$

Precisely we will consider simple potentials V, which are periodic with respect to some lattice $\mathbb{L}$. For this setting we will further develop mathematically guaranteed estimates for the total numerical error of our eigenvalue computations, such that the exact answer is provably within our error bound. Following the lecture we will use Bloch-Floquet theory to reformulate this problem in terms of Bloch fibers

$$H_k = \frac{1}{2}(-i\nabla_x + k)^2$$

where the k-points are taken from the Brillouin zone Ω^*. Of main interest and the focus of our work is the so-called band structure, that is the dependency of the few lowest eigenvalues of the H_k on k. At the end of this project you will have coded up a routine to compute the band structure of simple periodic potentials V in a way that the error is fully tracked and mathematically guaranteed error bars can be annotated to the computed band structure.

For our numerical computation in this project we will mostly employ the density-functional toolkit (DFTK), which has already been installed alongside this notebook. You can find the source code on dftk.org and documentation as well as plenty of usage examples on docs.dftk.org. In particular an alternative introduction to periodic problems, which focuses more on the numerics than the mathematics is given on https://docs.dftk.org/stable/guide/periodic_problems/.

Selection deleted

As example systems in this work we will only consider face-centred cubic crystals in the diamond structure. As the name suggests this family of crystals includes diamond, but also materials such as bulk silicon, germanium or tin. For these the lattice

$$\mathbb{L} = \mathbb{Z}a_1 + \mathbb{Z}a_2 + \mathbb{Z}a_3$$

can be constructed from the unit cell vectors

$$a_1 = \frac{a}{2}\begin{pmatrix}0\\1\\1\end{pmatrix} \qquad a_2 = \frac{a}{2}\begin{pmatrix}1\\0\\1\end{pmatrix} \qquad a_3 = \frac{a}{2}\begin{pmatrix}1\\1\\0\end{pmatrix}$$

where a is the lattice constant, which differs between the materials. Keeping the notation from the lecture we will use Ω to refer to the unit cell

$$\Omega = \left\{x \in \mathbb{R}^3 \,\middle|\, |x - R| > |x| \quad \forall R \in \mathbb{L} \setminus \{0\}\right\},$$

and employ $\mathbb{L}^*$ to refer to the reciprocal lattice with vectors b_1, b_2, b_3 with $a_i \cdot b_j = 2\pi\delta_{ij}$. Ω^* denotes the first Brillouin zone (unit cell of $\mathbb{L}^*$).

We introduce the normalised plane waves

$$e_G(x) = \frac{e^{iG\cdot x}}{\sqrt{|\Omega|}} \quad \forall G \in \mathbb{L}^*,$$

which allow to express the Fourier series of the potential as

$$V(x) = \frac{1}{\sqrt{|\Omega|}} \sum_{G\in\mathbb{L}^*} \hat{V}(G) e^{iG\cdot x} = \sum_{G\in\mathbb{L}^*} \hat{V}(G) e_G(x).$$

To simplify our notation we will assume that G-vectors always run over the countably infinite reciprocal lattice $\mathbb{L}^*$ without making explicit reference to this set.

An early model to describe these materials theoretically is the Cohen-Bergstresser model[CB1966]. The special and simplifying property of this model is that only a relatively small number of Fourier coefficients is non-zero. In particular we can define a cutoff $\mathcal{E}_V$ such that

Selection deleted

$$V(x) = \sum_{|G|<\sqrt{2\mathcal{E}_V}} \hat{V}(G) e_G(x),$$

where each $\hat{V}(G)$ is finite.

[CB1966]:

> M. L. Cohen and T. K. Bergstresser Phys. Rev. **141**, 789 (1966) DOI 10.1103/PhysRev.141.789

Task 1: Cohen-Bergstresser Hamiltonians

(a) Using the results of the lecture show that H is self-adjoint on $L^2(\mathbb{R}^3)$ with the domain $D(H) = H^2(\mathbb{R}^3)$ if V is the Cohen-Bergstresser potential. *Hint:* Show and use that the Cohen-Bergstresser potential is bounded and $\mathbb{L}$-periodic.

Selection deleted

(a) Solution:

A function is called bounded if there exists a constant C such that:

$$|V(\boldsymbol{x})| \leq C \qquad \forall \boldsymbol{x} \in \mathbb{R}^3.$$

Applying triangular inequality and using the fact that $|e^{i\boldsymbol{G}\cdot\boldsymbol{x}}| = 1$ as a modulus of a complex number in a polar form we have:

$$\begin{aligned} |V(\boldsymbol{x})| &\leq \frac{1}{\sqrt{|\Omega|}} \sum_{|G|<\sqrt{2\mathcal{E}_V}} |\hat{V}(G)||e^{i\boldsymbol{G}\cdot\boldsymbol{x}}| = \\ &= \sum_{|G|<\sqrt{2\mathcal{E}_V}} \frac{1}{\sqrt{|\Omega|}} |\hat{V}(G)| = C \qquad \forall \boldsymbol{x} \in \mathbb{R}^3, \end{aligned}$$

where $|\boldsymbol{\Omega}|$ is a constant since it is the volume of the parallelepiped spanned by three vectors that form a unit cell. Only a small number of Fourier coefficients is non-zero because there is a finite number of vectors $\boldsymbol{G}$ which is from the reciprocal lattice $\mathbb{L}^*$ and also inside the circle with the radius $\sqrt{2\mathcal{E}_V}$. Since $\hat{\boldsymbol{V}}(\boldsymbol{G})$ is given to be finite we can conclude that $\boldsymbol{C}$ is indeed a constant.

Therefore, the Cohen-Bergstresser potential is bounded.

On the other hand, from the periodicity of the complex exponential $e^{i\boldsymbol{G}\cdot\boldsymbol{x}}$ on the lattice $\mathbb{L}$ follows the $\mathbb{L}$-periodicity of $\boldsymbol{e_G(x)}$. Since $\boldsymbol{V(x)}$ is a finite sum of $\mathbb{L}$-periodic functions, it follows that $\boldsymbol{V(x)}$ is itself $\mathbb{L}$-periodic.

Moreover, function $\boldsymbol{V(x)} \in \boldsymbol{L^2(\Omega)}$ which means that $\int_\Omega |V(\boldsymbol{x})|^2 \, d\boldsymbol{x}$ is finite. Using Hölder's inequality, we have:

$$\begin{aligned} \int_\Omega |V(\boldsymbol{x})|^{3/2} \, d\boldsymbol{x} &\leq \left(\int_\Omega \left(|V(\boldsymbol{x})|^{3/2} \right)^{4/3} d\boldsymbol{x} \right)^{3/4} \left(\int_\Omega 1^4 \, d\boldsymbol{x} \right)^{1/4} = \\ &= \left(\int_\Omega |V(\boldsymbol{x})|^2 \, d\boldsymbol{x} \right)^{3/4} \left(\int_\Omega 1^4 \, d\boldsymbol{x} \right)^{1/4} < \infty, \end{aligned}$$

where $\left(\int_\Omega 1^4 \, d\boldsymbol{x} \right)^{1/4}$ is finite as a measure of the unit cell $\boldsymbol{\Omega}$ and $\left(\int (|V(\boldsymbol{x})|^2) \, d\boldsymbol{x} \right)^{3/4}$ is also finite by the assumption that $\boldsymbol{V(x)}$ is in $\boldsymbol{L^2}$.

Selection deleted

Therefore, $\boldsymbol{V(x)} \in \boldsymbol{L^{3/2}_{per}(\Omega)}$ and by applying theorem $\mathbf{10.1}$ from the lectures the operator $\boldsymbol{H}$ is self adjoint.

A consequence of *(a)* is that the Bloch-Floquet transformation yields Bloch fibers

$$H_k = T_k + V \quad \text{where} \quad T_k = \frac{1}{2}(-i\nabla_x + k)^2$$

which are self-adjoint on $L^2_{\text{per}}(\Omega)$ with $D(H_k) = H^2_{\text{per}}(\Omega)$. The natural inner product for our problem is thus the $\langle \cdot | \cdot \rangle_{L^2_{\text{per}}(\Omega)}$ inner product, which is the one we will employ unless a different one is specified.

For a particular k-point we can thus discretise the problem using the plane-wave basis

$$\mathbb{B}_k^{\mathcal{E}} = \left\{ e_G \,\middle|\, G \in \mathbb{L}^* \text{ and } \frac{1}{2}|G+k|^2 < \mathcal{E} \right\}$$

where $\mathcal{E}$ is a chosen cutoff.

(b) Show that

$$\langle e_G | e_{G'} \rangle = \delta_{GG'}$$

and that

$$\langle e_G | T_k e_{G'} \rangle = \delta_{GG'} \frac{1}{2}|G+k|^2.$$

From this conclude that for any $e_G \in \mathbb{B}_k^{\mathcal{E}}$ we have

$$|\Delta G| > \sqrt{2\mathcal{E}_V} \quad \Longrightarrow \quad \langle e_{G+\Delta G} | H e_G \rangle = 0, \qquad (1)$$

i.e. that each plane wave only couples via the Hamiltonian with these plane waves, that differ in the wave vector by no more than $\sqrt{2\mathcal{E}_V}$.

Selection deleted

Solution (b):

We have

$$\langle e_G | e_{G'} \rangle = \int_\Omega e_G^*(x) e_{G'}(x)\, dx = \frac{1}{|\Omega|} \int_\Omega e^{i(G'-G)\cdot x}\, dx$$

In case when $G = G'$ we have:

$$\int_\Omega e^{i(G'-G)\cdot x}\, dx = \int_\Omega 1\, dx = |\Omega|$$

In case of $G \neq G'$ denoting $(G - G') = G''$ which is also an element of the reciprocal lattice $\mathbb{L}^*$:

$$\int_\Omega e^{iG''\cdot x}\, dx = \int_\Omega \left(\cos G'' \cdot x + i \sin G'' \cdot x\right) dx = \int_\Omega \cos G'' \cdot x\, dx.$$

Imaginary part is equal to zero since sine is an odd function. On the other hand, since we can reprecent the vectors x and G'' as follows:

$$x = \alpha_1 a_1 + \alpha_2 a_2 + \alpha_3 a_3$$

and

$$G'' = \beta_1 b_1 + \beta_2 b_2 + \beta_3 b_3,$$

we have

$$\cos G'' \cdot x = \cos\left(2\pi\left(\alpha_1\beta_1 + \alpha_2\beta_2 + \alpha_3\beta_3\right)\right) = 0$$

since $a_i \cdot b_j = 2\pi\delta_{ij}$. Finally, we can see that

$$\langle e_G | e_{G'} \rangle = \frac{1}{|\Omega|}\delta_{G,G'} \cdot |\Omega|.$$

Considering

Selection deleted
$$\begin{aligned}\langle e_G | T_k e_{G'} \rangle &= \int_\Omega e^{-iG\cdot x} \cdot \frac{1}{2}(-i\nabla_x + k)^2 e^{iG'\cdot x}\, dx \\ &= \frac{1}{2}\int_\Omega e^{-iG\cdot x}\left(-i\nabla_x + k\right)\left(-i\nabla_x e^{iG'\cdot x} + k e^{iG'\cdot x}\right) dx \\ &= \frac{1}{2}|G+k|^2 \langle e_G | e_{G'} \rangle = \frac{1}{2}\delta_{GG'}|G+k|^2\end{aligned}$$

Finally, considering

$$H = -\frac{1}{2}\Delta + V.$$

We have

$$\begin{aligned}\langle e_{G+\Delta G}|He_G\rangle = -\frac{1}{2}|G|^2\langle e_{G+\Delta G}|e_G\rangle + \langle e_{G+\Delta G}|Ve_G\rangle = \\ = 0 + \int_\Omega \sum_{|G'|<\sqrt{2\mathcal{E}_V}} \hat{V}(G')e^{i\cdot G'\cdot x}e^{-i\Delta G\cdot x}\,dx = 0\end{aligned}$$

using the fact that $|\Delta G| > \sqrt{2\mathcal{E}_V}$ and that $e_G \in \mathbb{B}_k^{\mathcal{E}}$.

Familiarisation and Bauer-Fike estimates

Work through https://docs.dftk.org/stable/guide/periodic_problems/ as well as https://docs.dftk.org/stable/guide/tutorial/ to get some basic familiarity with DFTK.

Following loosely another documented example, this code sets up a calculation of silicon using the Cohen-Bergstresser potential and $\mathcal{E} = 10$.

```
begin
    Si = ElementCohenBergstresser(:Si)
    atoms = [Si, Si]
    positions = [ones(3)/8, -ones(3)/8]
    lattice = Si.lattice_constant / 2 .* [[0 1 1.]; [1 0 1.]; [1 1 0.]]
end;
```

Selection deleted

```
PlaneWaveBasis discretization:
    architecture         : DFTK.CPU()
    num. mpi processes   : 1
    num. julia threads   : 4
    num. blas  threads   : 1
    num. fft   threads   : 1

    Ecut                 : 10.0 Ha
    fft_size             : (24, 24, 24), 13824 total points
    kgrid type           : Monkhorst-Pack
    kgrid                : [1, 1, 1]
    num. irred. kpoints  : 1

    Discretized Model(custom, 3D):
        lattice (in Bohr)    : [0         , 5.13061   , 5.13061   ]
                               [5.13061   , 0         , 5.13061   ]
                               [5.13061   , 5.13061   , 0         ]
        unit cell volume     : 270.11 Bohr³

        atoms                : Si₂
        atom potentials      : DFTK.ElementCohenBergstresser(Si)
                               DFTK.ElementCohenBergstresser(Si)

        num. electrons       : 8
        spin polarization    : none
        temperature          : 0 Ha

        terms                : Kinetic()
                               DFTK.AtomicLocal()
```

```
begin
    model = Model(lattice, atoms, positions; terms=[Kinetic(), AtomicLocal()])
    basis_one = PlaneWaveBasis(model; Ecut=10.0, kgrid=(1, 1, 1));
end
```

The `kgrid` parameter selects the $\boldsymbol{k}$-point mesh $\mathbb{K} \subset \overline{\Omega^*}$, which is employed by determining the number of $\boldsymbol{k}$-points in each space dimension. In this case only a single $\boldsymbol{k}$-point is used, namely the origin of $\overline{\Omega^*}$:

```
[KPoint([     0,      0,      0], spin = 1, num. G vectors =   411)]
```

```
basis_one.kpoints
```

The origin of the Brillouin zone has a special name and is usually called Γ-point. Before treating the case of band structure computations (where $\boldsymbol{k}$ is varied), we stick to using only a single $\boldsymbol{k}$-point.

Each `KPoint` in DFTK automatically stores a representation of the basis $\mathbb{B}_{\boldsymbol{k}}^{\mathcal{E}}$. If you are curious, the $\boldsymbol{G}$-vectors defining the respective plane waves $\boldsymbol{e}_{\boldsymbol{G}}$ can be looked at using the functions

Selection deleted

```
[StaticArraysCore.SVector{3, Float64}: [0.0, 0.0, 0.0], StaticArraysCore.SVector{3, Float
```

```
G_vectors_cart(basis_one, basis_one.kpoints[1])  # G-vectors of 1st k-point
```

Similarly a discretised representation of H restricted to the fibres matching the chosen $\boldsymbol{k}$-point mesh can be obtained using

```
ham_one =
  Hamiltonian(PlaneWaveBasis discretization:                                    , [DftHami
                  architecture         : DFTK.CPU()
```

```
1 ham_one = Hamiltonian(basis_one)
```

Let us denote by $P_k^{\mathcal{E}}$ the projection into the plane-wave basis $\mathbb{B}_k^{\mathcal{E}}$ and further by $H_k^{\mathcal{E}\mathcal{E}} \equiv P_k^{\mathcal{E}} H_k P_k^{\mathcal{E}}$ the discretised fibers with elements

$$\left(H_k^{\mathcal{E}\mathcal{E}}\right)_{GG'} = \left(P_k^{\mathcal{E}} H_k P_k^{\mathcal{E}}\right)_{GG'} = \langle e_G,\, H_k\, e_{G'} \rangle \qquad \text{for } G, G' \in \mathbb{B}_k^{\mathcal{E}}.$$

Representations of $H_k^{\mathcal{E}\mathcal{E}}$ in DFTK are available via indexing of `Hamiltonian` objects. E.g. `ham[1]` corresponds to the fiber of `basis.kpoints[1]`, `ham[2]` to `basis.kpoints[2]` and so on. This can also be seen from comparing their sizes with the size of $\mathbb{B}_k^{\mathcal{E}}$ (available via the `G_vectors_cart`):

`DFTK.DftHamiltonianBlock`

```
1 begin
2     @show length(G_vectors_cart(basis_one, basis_one.kpoints[1]))
3     @show size(ham_one[1])
4     @show typeof(ham_one[1])
5 end
```

```
length(G_vectors_cart(basis_one, basis_one.kpoints[1])) = 411
size(ham_one[1]) = (411, 411)
typeof(ham_one[1]) = DFTK.DftHamiltonianBlock
```

These `DftHamiltonianBlock` objects very much behave like matrices, e.g. they can be multiplied with vectors, but also used in iterative solvers.

The `diagonalize_all_kblocks` function does some optimisations and book-keeping to solve for the `n_bands` lowest eigenpairs at each H_k. Here we solve for 6 eigenpairs:

```
(λ = [[-0.079228, 0.3843, 0.3843, 0.3843, 0.510143, 0.510143]], X = [411×6 Matrix{ComplexF
                                                                      -0.503153+0.806492
```

```
1 begin
2     n_bands = 6
3     eigres_one = diagonalize_all_kblocks(DFTK.lobpcg_hyper, ham_one, n_bands)
4 end
```

Selection deleted

The returned object `eigres_one` now contains the eigenvalues in `band_data.λ` and the Bloch waves in `band_data.X`. Again both are a vector along the k-points, further `band_data.X[ik]` is of size `length(G_vectors_cart(basis, kpoint))` × `n_bands`:

```
(411, 6)
```

```
begin
    @show size(eigres_one.X[1])
    @show (length(G_vectors_cart(basis_one, basis_one.kpoints[1])), n_bands)
end
```

```
size(eigres_one.X[1]) = (411, 6)
(length(G_vectors_cart(basis_one, basis_one.kpoints[1])), n_bands) = (411, 6)
```

As a result if we wanted to compute at each k-point the residual

$$H_k^{\mathcal{E}\mathcal{E}}\widetilde{X}_{kn} - \widetilde{\lambda}_{kn}\widetilde{X}_{kn} \qquad (2)$$

after we have been able to diagonalise $H_k^{\mathcal{E}\mathcal{E}}$, we can simply iterate as such:

```
for (ik, kpt) in enumerate(basis_one.kpoints)
    hamk = ham_one[ik]
    λk   = eigres_one.λ[ik]
    Xk   = eigres_one.X[ik]

    residual_k = hamk * Xk - Xk * Diagonal(λk)
    println(ik, "  ", norm(residual_k))
end
```

```
1  1.906458902168061e-6
```

Task 2: Γ-point calculations

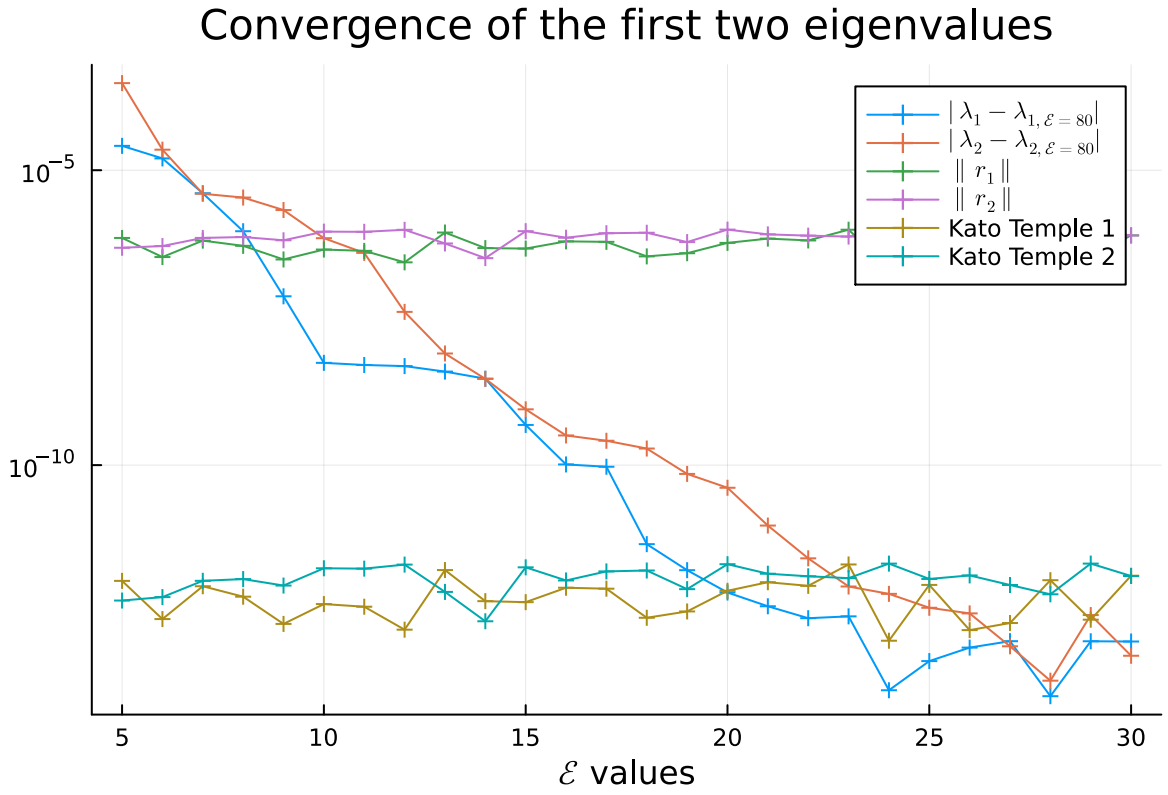
(a) For the case of employing only a single k-point (`kgrid = (1, 1, 1)`) vary $\mathcal{E}$ (i.e. `Ecut`) between 5 and 30. Taking $\mathcal{E} = 80$ as a reference, plot the convergence of the first two eigenpairs of H_k at the Γ point.

Selection deleted

```
9257×2 Matrix{ComplexF64}:
    -0.229462-0.922464im       7.86019e-8+2.26517e-8im
    -0.025416-0.102175im       -0.0820593+0.238735im
  -0.00235226-0.00945631im    -0.00077991+0.00226897im
   4.47873e-5+0.000180067im  -0.000226507+0.000658962im
   3.26618e-7+1.31129e-6im     1.46728e-5-4.26949e-5im
   2.83319e-7+1.13659e-6im    -2.04392e-7+5.95927e-7im
 -6.02277e-10-2.78884e-9im     2.36527e-8-6.86782e-8im
                ⋮
  -3.61888e-8-1.45086e-7im     6.02275e-9-3.6302e-7im
  -7.97043e-8-3.22212e-7im    -4.34045e-6+1.35597e-5im
  -3.88321e-5-0.000156105im    8.52411e-5-2.79917e-5im
  -2.73819e-5-0.000110068im    0.00180911+0.00496245im
  -0.00247903-0.009966im        0.0435335+0.00199638im
     0.025416+0.102175im         0.262493+0.254536im
```

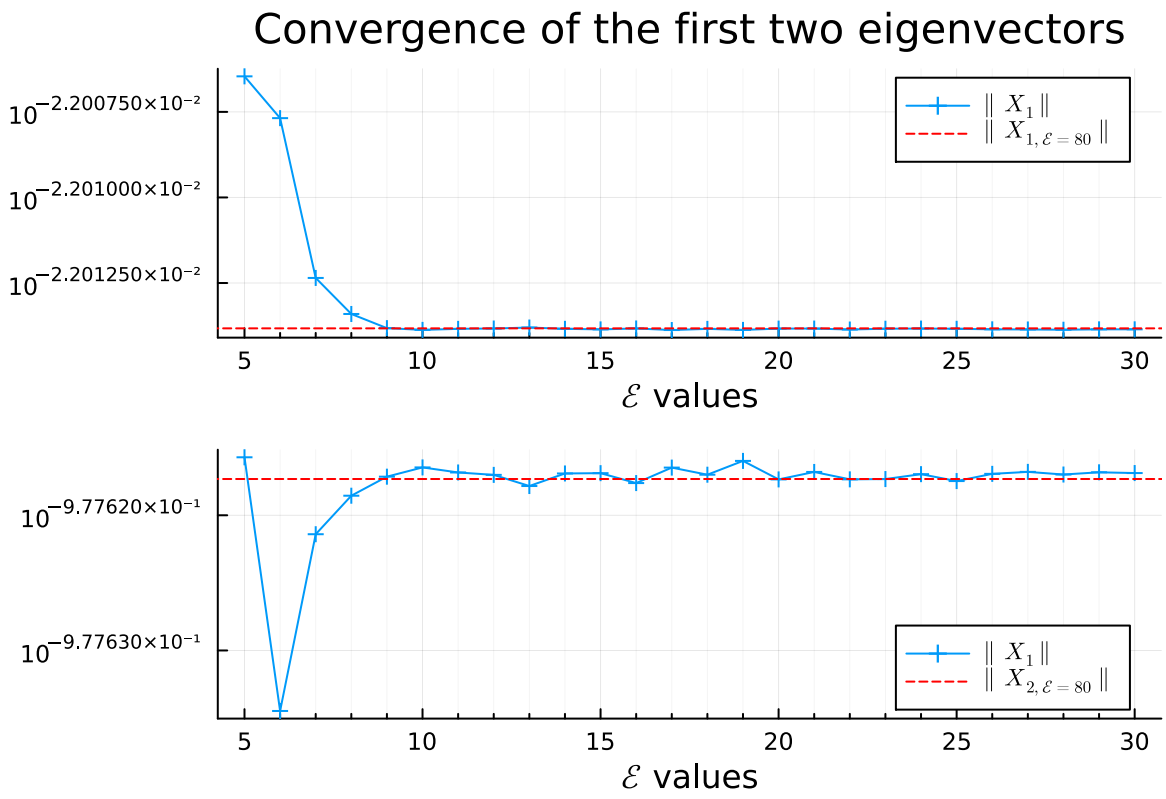
```
begin

    Ecuts = 5:30
    Ecut_ref = 80
    n_bands_2a = 2
    ik = 1

    eigenvalues_2a = zeros(length(Ecuts), n_bands_2a)
    eigenvec1 = []
    eigenvec2 = []

    for (i, Ecut) in enumerate(Ecuts)
        basis_2a = PlaneWaveBasis(model; Ecut=Ecut, kgrid=(1, 1, 1))
        ham_2a = Hamiltonian(basis_2a)
        eigres_2a = diagonalize_all_kblocks(DFTK.lobpcg_hyper, ham_2a, n_bands_2a)

        eigenvalues_2a[i, :] = eigres_2a.λ[ik][1:n_bands_2a]
        Xk = eigres_2a.X[ik]
        push!(eigenvec1, Xk[1])
        push!(eigenvec2, Xk[2])

    end

    basis_r = PlaneWaveBasis(model; Ecut=Ecut_ref, kgrid=(1, 1, 1))
    ham_r = Hamiltonian(basis_r)
    eigres_r = diagonalize_all_kblocks(DFTK.lobpcg_hyper, ham_r, n_bands_2a)
    λ_ref = eigres_r.λ[ik]
    X_ref = eigres_r.X[ik]

end

```

Selection deleted




```
begin
    p = plot(yaxis=:log, xlabel=L"$\mathcal{E}$ values")
    λ1 = abs.(eigenvalues_2a[:, 1] .- λ_ref[1])
    λ2 = abs.(eigenvalues_2a[:, 2] .- λ_ref[2])


    plot!(p, Ecuts ,λ1, shape=:cross, label=L"|\lambda_1-\lambda_{1,\mathcal{E}=80}|")
    plot!(p, Ecuts, λ2, shape=:cross, label=L"|\lambda_2-\lambda_{2,\mathcal{E}=80}|")

    plot!(p, Ecuts, ρ_bauer_fike[:, 1], shape=:cross, label=L"\parallel r_1
    \parallel")
    plot!(p, Ecuts, ρ_bauer_fike[:, 2], shape=:cross, label=L"\parallel r_2
    \parallel")

    plot!(p, Ecuts, ρ_kato_temple[:, 1], shape=:cross, label="Kato Temple 1")
    plot!(p, Ecuts, ρ_kato_temple[:, 2], shape=:cross, label="Kato Temple 2")

    title!(p, "Convergence of the first two eigenvalues")
end
```

Selection deleted




```
begin
    p_x= plot(layout=(n_bands_2a, 1), xlabel=L"$\mathcal{E}$ values", yscale=:log10, minorgrid=true)
    X1=norm.(eigenvec1)
    X2=norm.(eigenvec2)

    plot!(p_x, Ecuts, X1, shape=:cross, subplot=1, label=L"\parallel X_1 \parallel")
    hline!([norm.(X_ref[1])], line=:dash, color=:red, subplot=1, label=L"\parallel X_{1, \mathcal{E}=80}\parallel")

    plot!(p_x, Ecuts, X2, shape=:cross, subplot=2, label=L"\parallel X_1 \parallel")
    hline!([norm.(X_ref[2])], line=:dash, color=:red, subplot=2, label=L"\parallel X_{2, \mathcal{E}=80}\parallel")

    title!(p_x, subplot=1, "Convergence of the first two eigenvectors")


end
```

Selection deleted

We want to compare this convergence behaviour with a first estimate of the eigenvalue discretisation error based on the Bauer-Fike bound. Given an approximate eigenpair $(\widetilde{\lambda}_{kn}, \widetilde{X}_{kn})$ of the fiber H_k we thus need access to the residual

$$r_{kn} = H_k \widetilde{X}_{kn} - \widetilde{\lambda}_{kn} \widetilde{X}_{kn}.$$

If we performed our computation using the plane-wave basis $\mathbb{B}_k^{\mathcal{E}}$, then by construction

$$P_k^{\mathcal{E}} \widetilde{X}_{kn} = \widetilde{X}_{kn},$$

such that

$$\begin{aligned} P_k^{\mathcal{E}} r_{kn} &= P_k^{\mathcal{E}} H_k P_k^{\mathcal{E}} \, P_k^{\mathcal{E}} \widetilde{X}_{kn} - \widetilde{\lambda}_{kn} P_k^{\mathcal{E}} \widetilde{X}_{kn} \\ &= H_k^{\mathcal{E}\mathcal{E}} \widetilde{X}_{kn} - \widetilde{\lambda}_{kn} \widetilde{X}_{kn}, \end{aligned}$$

which is exactly the quantity we computed in (2) above. In other words $P_k^{\mathcal{E}} r_{kn}$ is the residual corresponding to the iterative diagonalisation, thus leading to the **algorithm error**.

However, we are also interested in the **discretisation error**, which we obtain from the missing residual term in (2), namely

$$Q_k^{\mathcal{E}} r_{kn} = (1 - P_k^{\mathcal{E}}) r_{kn} = Q_k^{\mathcal{E}} H_k P_k^{\mathcal{E}} \widetilde{X}_{kn} = H_k^{\mathcal{E}^\perp \mathcal{E}} \widetilde{X}_{kn}$$

where $Q_k^{\mathcal{E}} = (1 - P_k^{\mathcal{E}})$ is the orthogonal projector to $P_k^{\mathcal{E}}$ and we introduced the notation

$$H_k^{\mathcal{E}^\perp \mathcal{E}} = Q_k^{\mathcal{E}} H_k P_k^{\mathcal{E}}.$$

While computing $P_k^{\mathcal{E}} r_{kn}$ is generally easy (we just did it above in 5 lines), obtaining $Q_k^{\mathcal{E}} r_{kn}$ is impossible for general potentials V: since $H_k P_k^{\mathcal{E}} \widetilde{X}_{kn}$ can have support anywhere on $L^2_{\text{per}}(\Omega)$, estimating $Q_k^{\mathcal{E}} r_{kn}$ usually requires making a mathematical statement about the behaviour of H_k on all of $L^2_{\text{per}}(\Omega)$. Estimating discretisation errors is thus substantially more challenging than estimating algorithm errors.

(b) In our case of Cohen-Bergstresser Hamiltonians we are more lucky. Using (1) show that there exists a cutoff $\mathcal{F} > \mathcal{E}$, such that

Selection deleted

$$r_{kn} = P_k^{\mathcal{F}} r_{kn} = H_k^{\mathcal{F}\mathcal{F}} \widetilde{X}_{kn} - \widetilde{\lambda}_{kn} \widetilde{X}_{kn}.$$

Having diagonalised the Hamiltonian using cutoff $\mathcal{E}$ we can thus obtain the *full residual* by a computation of the Hamiltonian-vector product using an elevated cutoff $\mathcal{F}$.

Solution (b)

We need to find such cutoff $\mathcal{F} > \mathcal{E}$ that

$$r_{kn} = P_k^{\mathcal{F}} r_{kn} \iff Q_k^{\mathcal{F}} r_{kn} = 0 \iff H_k^{\mathcal{F}^\perp \mathcal{E}} \widetilde{X}_{kn} = 0,$$

where

$$\left(H_k^{\mathcal{F}^\perp \mathcal{E}}\right)_{GG'} = \langle e_G,\, H_k\, e_{G'} \rangle$$

If $e_G \in (\mathbb{B}_k^{\mathcal{F}})^\perp$ and $e_{G'} \in \mathbb{B}_k^{\mathcal{E}}$ which means that

$$\frac{1}{2}|G+k|^2 < \mathcal{F} \quad \text{and} \quad \frac{1}{2}|G'+k|^2 < \mathcal{E},$$

from which it follows that

$$e_G \in (\mathbb{B}_k^{\mathcal{F}})^\perp \iff \frac{1}{2}|G+k|^2 \geq \mathcal{F} \qquad (*).$$

Considering $e_G \in \mathbb{B}_k^{\mathcal{F}}$ and $e_{G'} \in \mathbb{B}_k^{\mathcal{E}}$ and applying the reverse triangular inequality we have

$$\begin{aligned} |G - G'| = |G + k - G' - k| \geq \big||G+k| - |G'+k|\big| \geq \\ \geq \sqrt{2\mathcal{F}} - \sqrt{2\mathcal{E}} \end{aligned}$$

From (1) we can see that for any $e_G \in \mathbb{B}_k^{\mathcal{F}}$

$$|\Delta G| > \sqrt{2\mathcal{E}_V} \quad \Longrightarrow \quad \langle e_{G+\Delta G} | H_k e_G \rangle = 0.$$

To have $\langle e_{G+\Delta G} | H_k e_G \rangle \neq 0$ and since $\mathbb{B}_k^{\mathcal{E}} \subset \mathbb{B}_k^{\mathcal{F}}$ for $e_G \in \mathbb{B}_k^{\mathcal{F}}$ and $e_{G'} \in \mathbb{B}_k^{\mathcal{E}}$ we need

$$|\Delta G| = |G - G'| \leq \sqrt{2\mathcal{E}_V}$$

From both inequalities we have

$$\sqrt{\mathcal{F}} - \sqrt{\mathcal{E}} = \sqrt{\mathcal{E}_V}.$$

Therefore, if we take

Selection deleted

$$\mathcal{F} = \left(\sqrt{\mathcal{E}_V} + \sqrt{\mathcal{E}}\right)^2.$$

it will follow that

$$\left(H_k^{\mathcal{F}^\perp\mathcal{E}}\right)_{GG'} = \langle e_G,\, H_k\, e_{G'}\rangle = \langle e_G,\, V\, e_{G'}\rangle =$$
$$= \delta_{GG'}\frac{1}{2}|G+k|^2 + \int_\Omega \sum_{|G''|<\sqrt{2\mathcal{E}_V}} \hat{V}(G'')e^{i\cdot G''\cdot x}e^{i(G-G')\cdot x}\,dx = 0$$

since $G' \neq G$ from $(*)$.

(c) In DFTK nobody stops you from using multiple bases of different size. Moreover, having obtained a set of bloch waves `X_small` using `basis_small` you can obtain its representation on the bigger `basis_large` using the function

```
X_large = transfer_blochwave(X_small, basis_small, basis_large)
```

Using this technique you can compute the application of the Hamiltonian using a bigger basis (just as `Hamltonian(basis_large) * X_large`). Use this setup to vary $\mathcal{F}$ and use this to estimate $\mathcal{E}_V$ numerically. Check your estimate for various values of $\mathcal{E}$ to ensure it is consistent. Rationalise your results by taking a look at the Cohen Bergstresser implementation in DFTK.

Solution (c):

Selection deleted

```
begin

    E_list = 5:10
    F_found_list = []
    for E in E_list

        local basis_small = PlaneWaveBasis(model; Ecut=E, kgrid=(1, 1, 1))
        local ham = Hamiltonian(basis_small)
        local eigres = diagonalize_all_kblocks(DFTK.lobpcg_hyper, ham, 2)

        F_list = E:5*E
        local ik = 1

        res_norms = []

        for (i, F) in enumerate(F_list)
            basis_large = PlaneWaveBasis(model; Ecut=F, kgrid=(1, 1, 1))
            ham_large = Hamiltonian(basis_large)
            eigres_large = diagonalize_all_kblocks(DFTK.lobpcg_hyper, ham_large, 2)

            Xk_large = transfer_blochwave(eigres.X , basis_small, basis_large)

            residual = ham_large[ik] * Xk_large[ik] - Xk_large[ik] *
            Diagonal(eigres.λ[ik])
            push!(res_norms, norm(residual))

            if i > 1 && abs(res_norms[i] - res_norms[i-1]) < 1e-8
                push!(F_found_list, F)
                break
            end

        end

    end
end

```

```
Ev_estimated = 2.1720066733367176
```

```
Ev_estimated = mean((sqrt.(F_found_list) - sqrt.(E_list)).^2)
```

Since the reciprocal lattice constant $b = \frac{2\pi}{a}$ and from the Cohen Bergstresser implementation in DFTK the potential is calculated for the sum over G such that

Selection deleted

$$|G|^2 < 11\left(\frac{2\pi}{a}\right)^2 \approx 4.124$$

by definition given above

$$|G|^2 < 2\mathcal{E}_V \quad \Longrightarrow \quad \mathcal{E}_V > 2.062$$

```
2.0621726989864846
```

```
0.5 * 11 * (2 * π / Si.lattice_constant) ^ 2
```

On the other hand, G' such that

$$|G'|^2 > 11\left(\frac{2\pi}{a}\right)^2$$

are not included while computing V meaning that

$$|G'|^2 > 2\mathcal{E}_V$$

giving an upper bound for $\mathcal{E}_V$.

```
2.24965
```

```
begin
    l_bound = 11 * (2π / Si.lattice_constant) ^ 2
    G_norms = (norm.(G_vectors_cart(basis_one, basis_one.kpoints[1]))) .^ 2
    0.5 * sort([g for g in round.(G_norms, sigdigits=5) if g > l_bound])[1]
end
```

Therefore,

$$2.06 < \mathcal{E}_V < 2.25$$

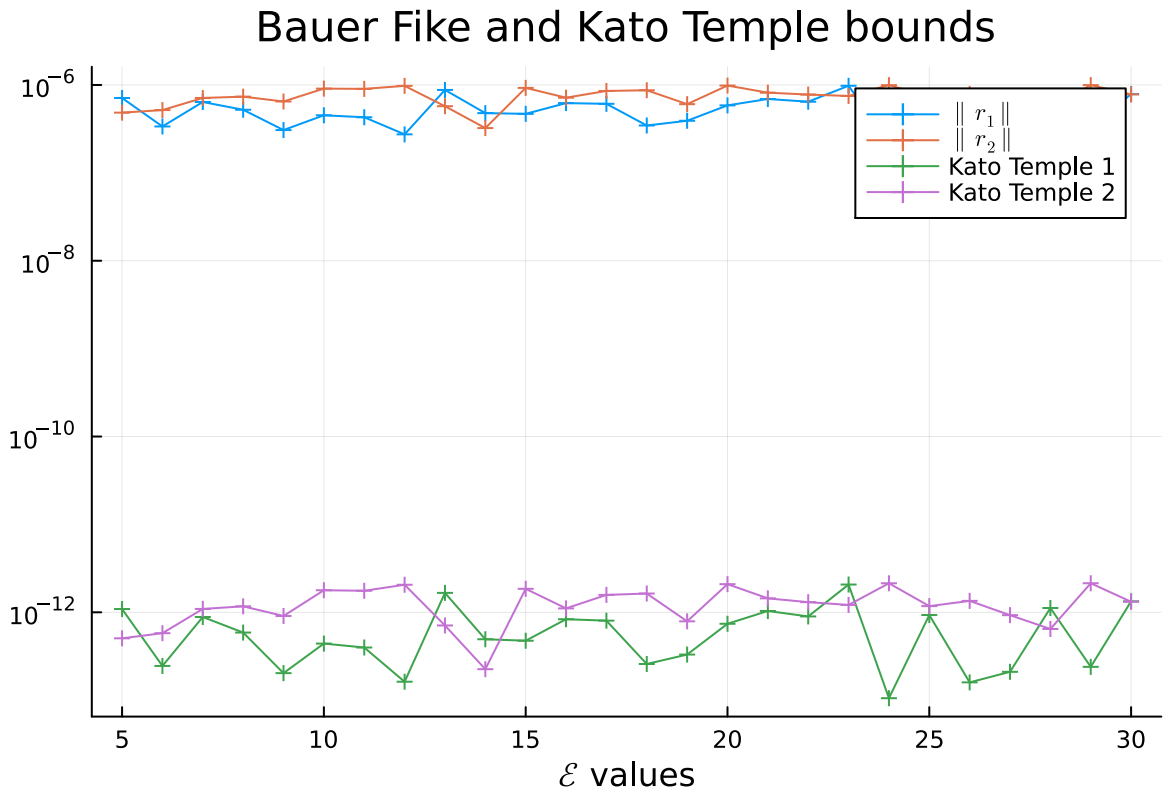
which is in line with the value estimated above.

(d) Based on the Bauer-Fike bound estimate the algorithm and arithmetic error for the first two eigenpairs at the Γ point and for using cutoffs between $\mathcal{E} = 5$ and $\mathcal{E} = 30$. Add these estimates to your plot in (a). What do you observe regarding the tightness of the bound ?

Selection deleted

```
begin
    local n_bands = 2
    ρ_bauer_fike = zeros(length(Ecuts), n_bands)

    for (i, Ecut) in enumerate(Ecuts)
        basis_2a = PlaneWaveBasis(model; Ecut=Ecut, kgrid=(1, 1, 1))
        ham_2a = Hamiltonian(basis_2a)
        eigres_2a = diagonalize_all_kblocks(DFTK.lobpcg_hyper, ham_2a, n_bands_2a)

        eigenvalues_2a[i, :] = eigres_2a.λ[ik][1:n_bands_2a]
        Xk = eigres_2a.X[ik]


        residual_k = ham_2a[ik] * Xk - Xk * Diagonal(eigres_2a.λ[ik])
        ρ_bauer_fike[i, :] = norm.(eachcol(residual_k))

    end


end
```

Selection deleted




```
begin

    local p= plot(yaxis=:log, xlabel=L"$\mathcal{E}$ values")

    plot!(p, Ecuts, ρ_bauer_fike[:, 1], shape=:cross, label=L"\parallel r_1 \parallel")
    plot!(p, Ecuts, ρ_bauer_fike[:, 2], shape=:cross, label=L"\parallel r_2 \parallel")

    plot!(p, Ecuts, ρ_kato_temple[:, 1], shape=:cross, label="Kato Temple 1")
    plot!(p, Ecuts, ρ_kato_temple[:, 2], shape=:cross, label="Kato Temple 2")

    title!(p, "Bauer Fike and Kato Temple bounds")
end
```

We can notice that Bauer Fike bounds for both eigenvalues are not very tight but for the first $\mathcal{E}$ we can see that discretization error dominates knowing that Bauer Fike dives a bound for arithmetic and algorithmic errors.

Selection deleted

Band structure computations

Next we turn our attention to band structure computations. Recall that the n-th band is the mapping from k to the n-th eigenvalue λ_{kn} of the Bloch fiber H_k. For standard lattices — like the diamond-FCC lattices we consider — there are tabulated standard 1D paths through the Brillouin zone Ω^*, that should be used to maximise the physical insight one may gain from the computation.

In DFTK this path can be automatically determined. The following function provides a corresponding list of k-points to consider (the warnings can be ignored in this function call):

```
kpath =
```
```
Brillouin.KPaths.KPathInterpolant{3}: [StaticArraysCore.SVector{3, Float64}: [0.0, 0.0, 0
```
```
kpath = interpolate(irrfbz_path(model); density=15)
```

The `density` parameter in the above call determines how many k-points are considered along the path. A larger density leads to more points.

Based on this data we can use the `compute_bands` function to diagonalise all H_k along the path using a defined `density` and cutoff `Ecut`. Internally this function uses `diagonalize_all_kblocks`, so the returned data structure is alike.

```
compute_bands_auto (generic function with 1 method)
```
```
function compute_bands_auto(model; Ecut, kwargs...)
    basis = PlaneWaveBasis(model; Ecut, kgrid=(1, 1, 1))
    compute_bands(basis, kpath; show_progress=false, kwargs...)
end
```

```
band_data =
```
```
(basis = PlaneWaveBasis discretization:                                    , λ = [[-0.07
              architecture         : CPU()
```
```
band_data = compute_bands_auto(model; n_bands=6, tol=1e-3, Ecut=10)
```

For example we obtain the eigenvalues of the 3rd k-point

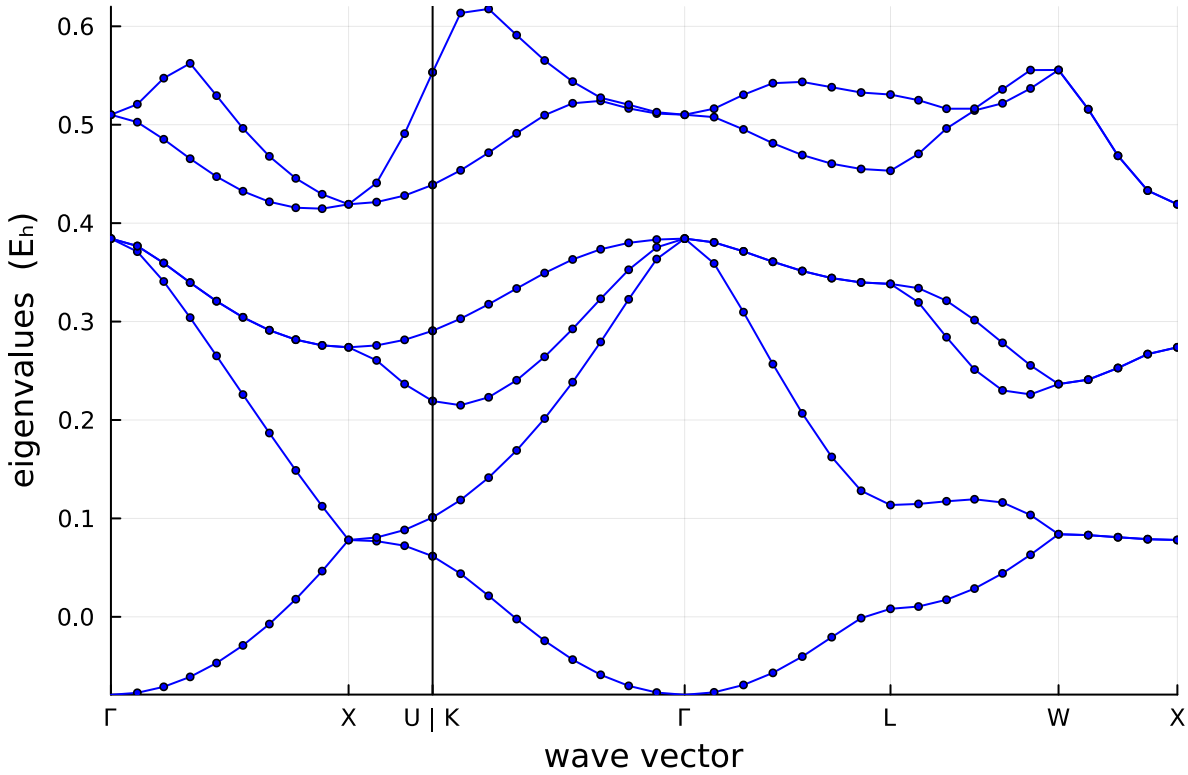
Selection deleted

```
KPoint([ 0.111,      0,  0.111], spin = 1, num. G vectors =   399)
```
```
band_data.basis.kpoints[3]
```

using the expression

```
[-0.0711114, 0.34071, 0.359448, 0.359448, 0.485213, 0.547298]
```
```
band_data.λ[3]
```

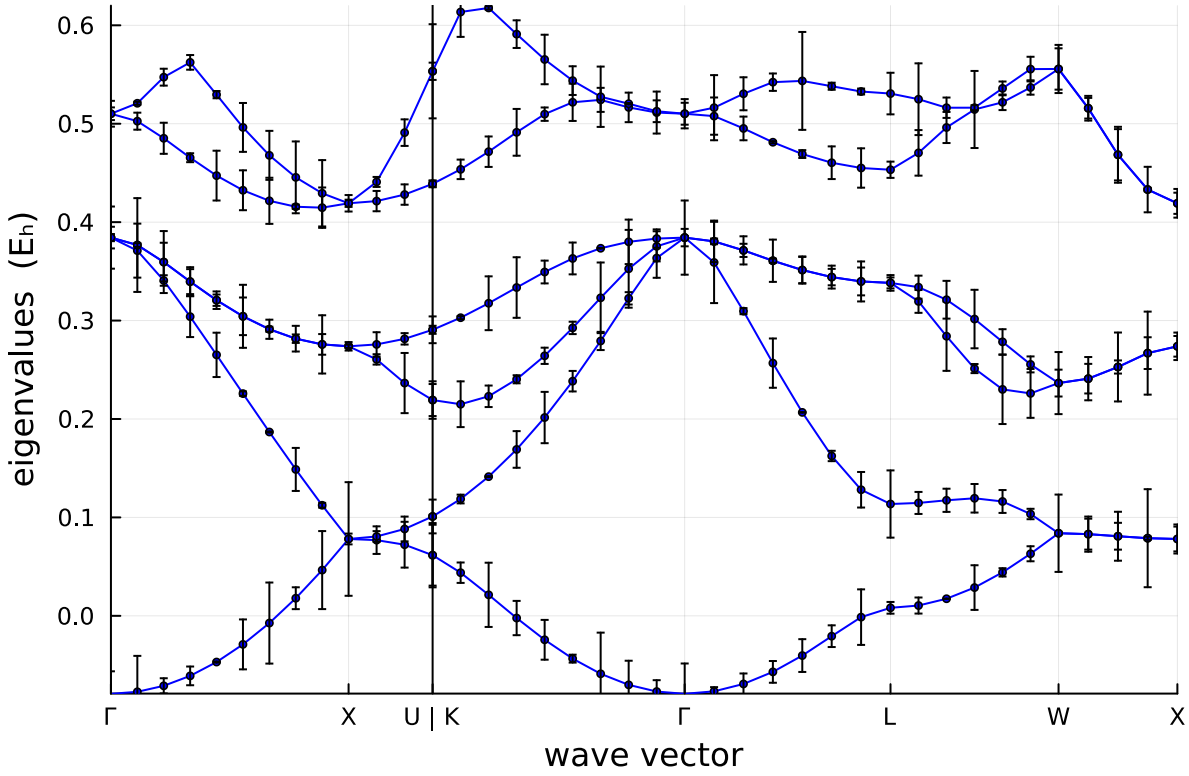
Once the data has been computed, a nice bandstructure plot can be produced using




```
1 DFTK.plot_band_data(kpath, band_data)
```

Error bars for indicating eigenvalue errors can also be easily added:

Selection deleted




```
begin
    λerror = [0.02 * abs.(randn(size(λk))) for λk in band_data.λ]  # dummy data
    data_with_errors = merge(band_data, (; λerror))
    DFTK.plot_band_data(kpath, data_with_errors)
end
```

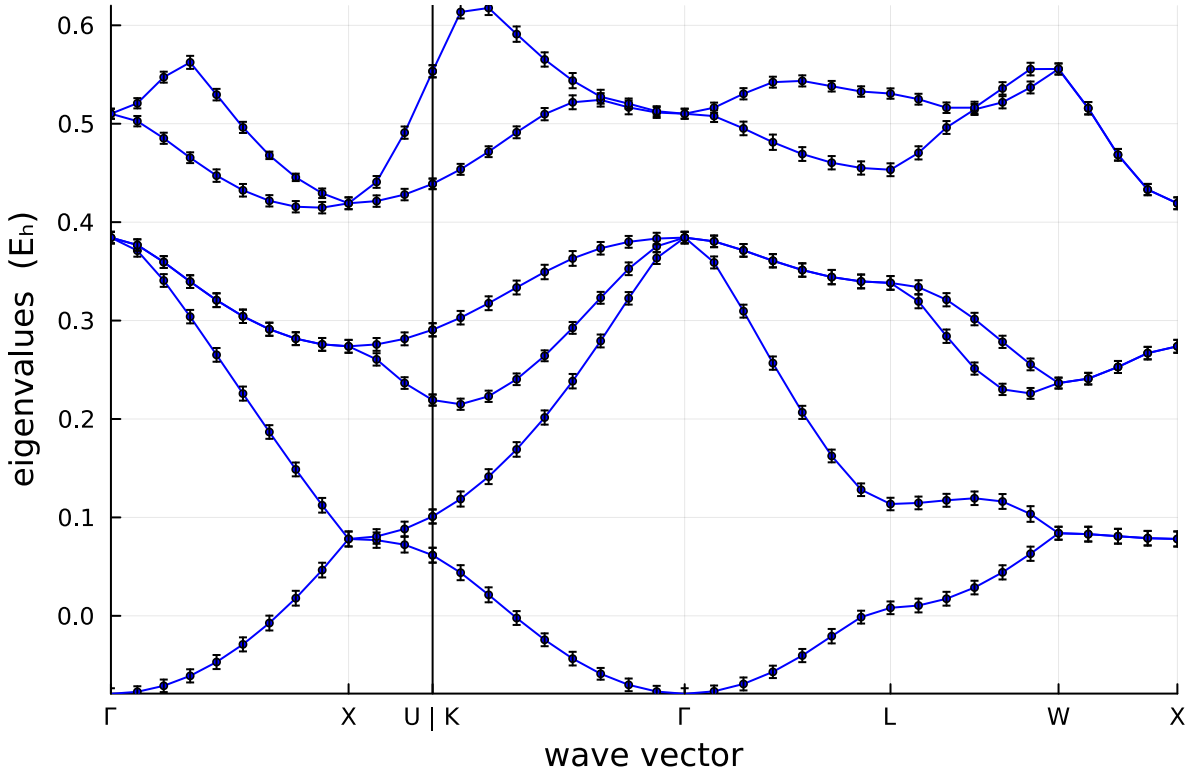
Task 3: Bands with Bauer-Fike estimates

For $\mathcal{E} = 7$ a k-point density of 15 and the 6 lowest eigenpairs plot a band structure in which you annotate the bands with error bars estimated from Bauer-Fike. To compute the Hamiltonian-vector product employing a larger basis set with cutoff `Ecut_large`, employ the following piece of code, which forwards the k-Point coordinates from one basis to another:

```
basis_change_Ecut (generic function with 1 method)
```

```
function basis_change_Ecut(basis_small, Ecut_large)
    PlaneWaveBasis(basis_small.model, Ecut_large,
                   basis_small.kcoords_global,
                   basis_small.kweights_global)
end
```

Selection deleted




```
begin
    local Ecut = 7
    Fcut = (sqrt(Ev_estimated) + sqrt(Ecut)) ^ 2
    local band_data = compute_bands_auto(model; n_bands=6, tol=1e-3, Ecut=Ecut)
    local basis_large = basis_change_Ecut(band_data.basis, Fcut)

    local X_large = transfer_blochwave(band_data.X , band_data.basis, basis_large)
    local ham = Hamiltonian(basis_large)

    ρ_bauer_fike3 = []

    for (ik, kpt) in enumerate(band_data.basis.kpoints)
        residual_k = ham[ik] * X_large[ik] - X_large[ik] * Diagonal(band_data.λ[ik])
        push!(ρ_bauer_fike3, norm.(eachcol(residual_k)))
    end

    band_data3_bf=merge(band_data,(;λerror=ρ_bauer_fike3))
    DFTK.plot_band_data(kpath, band_data3_bf)

end
```

Selection deleted

Kato-Temple estimates

Following up on Task 2 (d) we want to improve the bounds using Kato-Temple estimates. Similar to our previous developments in this regard the challenging ingredient is estimating a lower bound to the gap. If λ_{kn} denotes the eigenvalue closest to the approximate eigenvalue $\widetilde{\lambda}_{kn}$, we thus need a lower bound to

$$\delta_{kn} = \min_{s \in \sigma(H_k) \setminus \{\lambda_{kn}\}} |s - \widetilde{\lambda}_{kn}|.$$

The usual approach we employed so far (see e.g. Sheet 6) was to assume that our obtained approximations $\widetilde{\lambda}_{kn}$ for $n = 1, \dots, N$ did not miss any of the exact eigenvalues. With this assumption we would employ a combination of Kato-Temple and Bauer-Fike bound to obtain a lower bound to δ_{kn}.

Already for estimating the eigenvalue error of iterative eigensolvers, this can be a far-fetched assumptions as we saw in previous exercises. For estimating the discretisation error where the discretisation by nature of projecting into a finite-dimensional space inevitably changes the properties of the spectrum, this is even more far fetched. In fact in practice it is not rare that an unsuited basis changes the order of eigenpairs, completely neglects certain eigenpairs etc. Since the goal of *a posteriori* estimation is exactly to detect such cases, we will in Task 5 also develop techniques based on weaker assumptions.

Before doing so, we first extend the usual technique to infinite dimensions.

Selection deleted

Task 4: Simple and non-guaranteed Kato-Temple estimates

(a) Notice that the Kato-Temple theorem requires the targeted eigenvalue λ_{kn} to be isolated, see the conditions of Theorem 9.9. and 9.8. in the notes. Considering our setting of periodic Schrödinger operators in L^2. Looking at the spectral properties of H and H_k argue why there is even hope that Kato-Temple can be employed in our setting?

(b) Assume that our numerical procedure (discretisation + diagonalisation) for all considered cutoffs is sufficiently good, such that no eigenpairs are missed. Employ the prevously discussed (e.g. Sheet 6) technique of combining Kato-Temple and Bauer-Fike estimates to obtain an improved estimate for the error in the eigenvalue. Add your estimate for the first eigenvalue at the Γ-point to your plot in Task 2 (d). What do you observe ?

(c) Perform a band structure computation at cutoff $\mathcal{E} = 7$ annotated with the error estimated using the estimate developed in *(b)*. For approximate eigenvalues where you cannot obtain a Kato-Temple estimate (e.g. degeneracies), fall back to a Bauer-Fike estimate. Play with the cutoff. What do you observe ? Do the error bars correspond to the expectations of a variational convergence ? Apart from the probably unjustified assumption in *(b)*, what is the biggest drawback of the Kato-Temple estimate ?

Solution (a):

From the notes, we know that eigenvalues of H_k are well separated. And that the spectrum of H is a union of spectrums of H_k. As discussed in the lecture an energy band gives us a well-defined minimum and maximum which can be employed for using the Kato-Temple theorem.

Solution (b):

Selection deleted

```
get_KT_bound (generic function with 1 method)
```

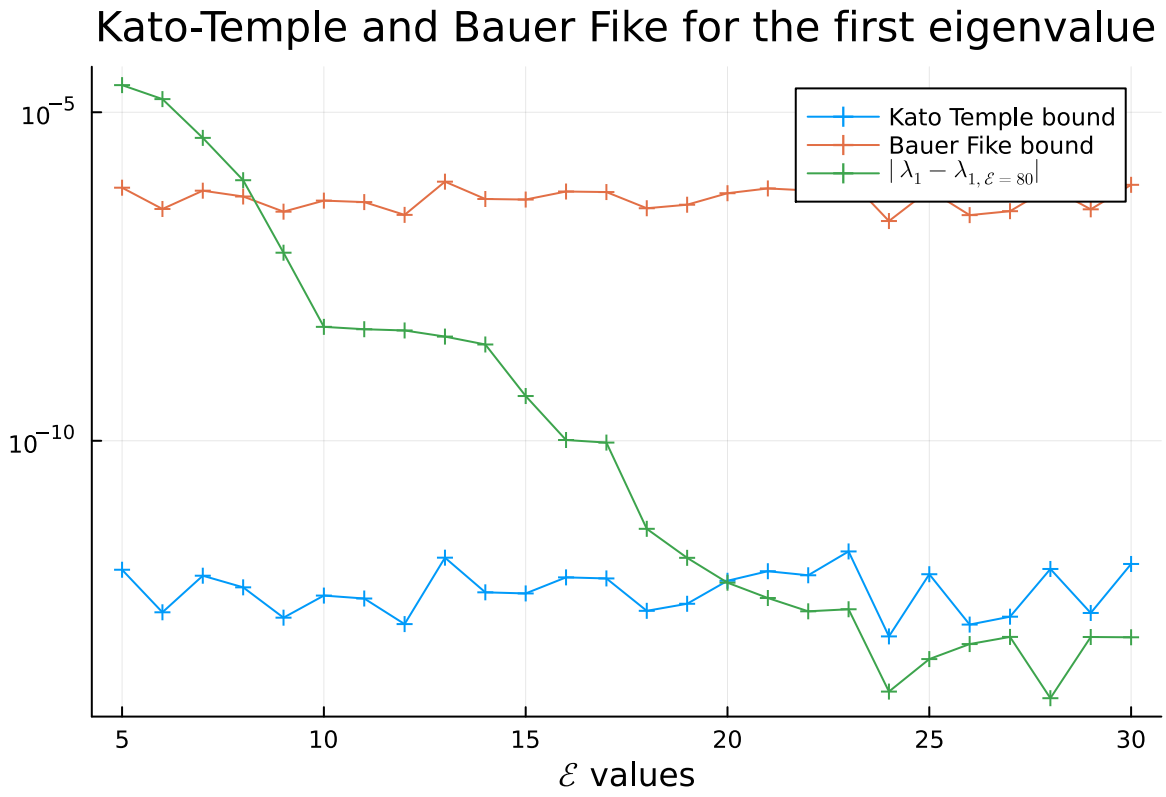
```
function get_KT_bound(computed_evalues, res_norms)
    n = size(res_norms)[1]

    δ_list = zeros(n)
    δ_list[1] = abs(computed_evalues[1] - computed_evalues[2]) -
    res_norms[2]
    δ_list[end] = abs(computed_evalues[end] - computed_evalues[end-1]) -
    res_norms[end-1]
    for i in 2:Int64(n - 1)
        δ_left = abs(computed_evalues[i] - computed_evalues[i-1]) -
        res_norms[i-1]
        δ_right = abs(computed_evalues[i] - computed_evalues[i+1]) -
        res_norms[i+1]
        δ_list[i] = min(δ_left, δ_right)
    end

    δ = [max(0, i) for i in δ_list]
    error_Kato_Temple = res_norms .^2 ./ δ
    error_Kato_Temple

end
```

```
begin
    local ik = 1
    ρ_kato_temple = zeros(length(Ecuts), n_bands_2a)

    for (i, Ecut) in enumerate(5:30)

        # Kato-Temple
        ρ_kato_temple[i, :]  = get_KT_bound(eigenvalues_2a[i, :], ρ_bauer_fike[i, :])

    end

end
```

```
[1.08969e-12, 2.45311e-13, 8.84173e-13, 5.88207e-13, 2.03148e-13, 4.40665e-13, 3.9757e-13,
```

```
ρ_kato_temple[:, 1]
```

Selection deleted

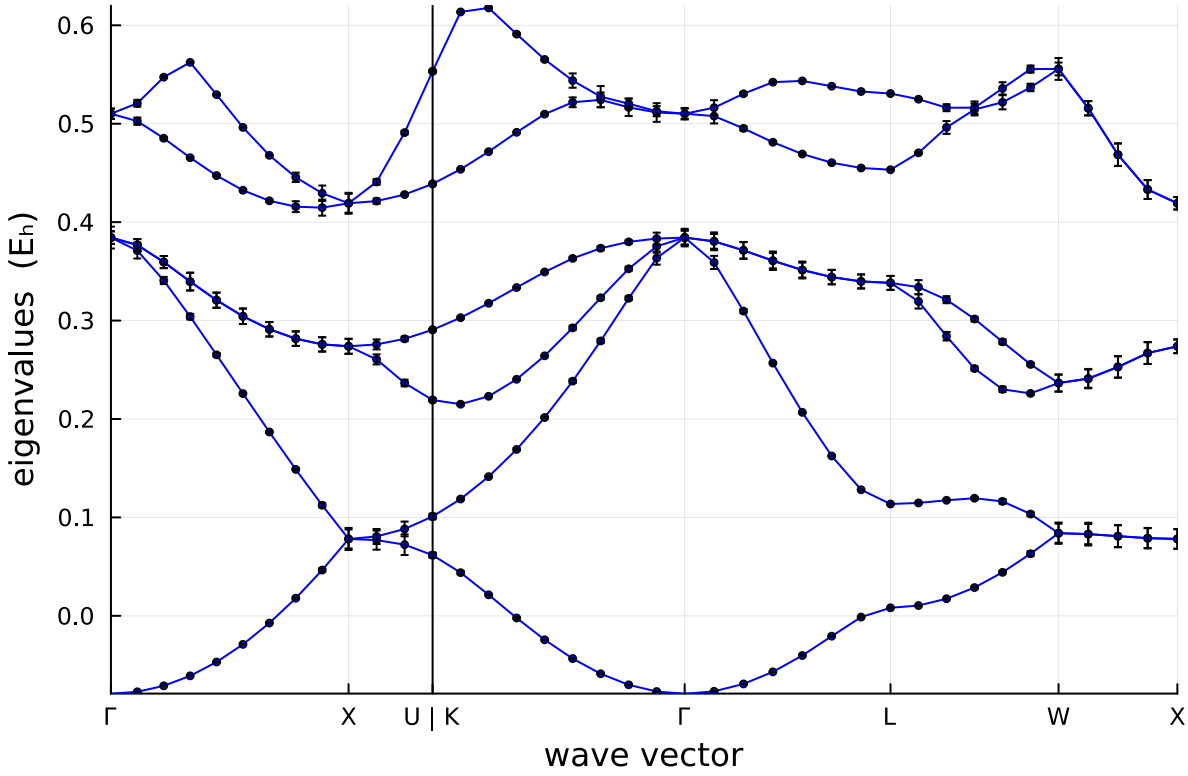



```
begin
    local p = plot(yaxis=:log, xlabel=L"$\mathcal{E}$ values")


    plot!(p, Ecuts, ρ_kato_temple[:, 1], shape=:cross, label="Kato Temple bound")
    # plot!(p, Ecuts, ρ_kato_temple[2], shape=:cross, label="Kato Temple evalue2")

    plot!(p, Ecuts, ρ_bauer_fike[:, 1], shape=:cross, label="Bauer Fike bound")
    # # plot!(p, Ecuts, ρ_bauer_fike[:, 2], shape=:cross, label="Bauer Fike evalue2")

    plot!(p, Ecuts, λ1, shape=:cross, label=L"|\lambda_1-\lambda_{1,\mathcal{E}=80}|")
    # # plot!(p, Ecuts, λ2, shape=:cross, label=L"|\lambda_2-
    \lambda_{2,\mathcal{E}=80}|")

    title!(p, "Kato-Temple and Bauer Fike for the first eigenvalue")

end
```

Selection deleted

In this case the Kato-Temple bound is tighter than its Bauer-Fike bound which makes Kato-Temple more useful in our task. Moreover, having a bound for algorithmic and arithmetic error we can conclude depending on the value $\mathcal{E}$ where discretization error dominates in our approximate solution.

Solution (c):




```
begin
    local Ecut = 7
    local Fcut = (sqrt(Ev_estimated) + sqrt(Ecut)) ^ 2
    local band_data = compute_bands_auto(model; n_bands=6, tol=1e-2, Ecut=Ecut)
    local basis_large = basis_change_Ecut(band_data.basis, Fcut)

    local X_large = transfer_blochwave(band_data.X , band_data.basis, basis_large)
    local ham = Hamiltonian(basis_large)

    ρ_bauer_fike4 = []
    ρ_kato_temple4 = []
    ρ_best = []

    for (ik, kpt) in enumerate(band_data.basis.kpoints)
        residual_k = ham[ik] * X_large[ik] - X_large[ik] * Diagonal(band_data.λ[ik])
        r_norms = norm.(eachcol(residual_k))
        kt_bound = get_KT_bound(band_data.λ[ik], r_norms)

        push!(ρ_bauer_fike4, r_norms)
        push!(ρ_kato_temple4, kt_bound)

        push!(ρ_best, min.(r_norms, kt_bound))
    end


    band_data4_bf=merge(band_data,(;λerror=ρ_best))
    DFTK.plot_band_data(kpath, band_data4_bf)

end

```

Selection deleted

Analyzing the plots from **(c)** and task 3 we observe that the error bars derived from the Kato-Temple estimate are tighter compared to those obtained only from the Bauer-Fike estimate. At the same time, the error bars tend to decrease with increasing cutoff, which would align with the expectations of smaller discretization errors leading to more accurate results. The main limitation is that we can apply the Kato-Temple theorem only when the targeted eigenvalue is isolated which is not always the case.

Task 5: A Schur-based estimate for the gap

Based on the definition of the basis cutoffs $\mathcal{E}$ and $\mathcal{F}$ and the respective projectors $P_k^{\mathcal{E}}$, $P_k^{\mathcal{F}}$ into these bases as well as $Q_k^{\mathcal{E}}$ and $Q_k^{\mathcal{F}}$ as the projectors to the complements of the bases, we can identify the following orthogonal decompositions of the Hilbert space $L^2_{\text{per}}(\Omega)$:

$$\begin{aligned} id &= (P_k^{\mathcal{E}} + Q_k^{\mathcal{E}}) \\ &= (P_k^{\mathcal{E}} + P_k^{\mathcal{F}} Q_k^{\mathcal{E}} + Q_k^{\mathcal{F}} Q_k^{\mathcal{E}}) \\ &= (P_k^{\mathcal{E}} + P_k^{\mathcal{R}} + Q_k^{\mathcal{F}}) \end{aligned}$$

where we defined $P_k^{\mathcal{R}} \equiv P_k^{\mathcal{F}} Q_k^{\mathcal{E}}$. Using the notations

$$\begin{aligned} H_k^{\mathcal{E}\mathcal{F}} &= P_k^{\mathcal{E}} H_k P_k^{\mathcal{F}} & H_k^{\mathcal{R}\mathcal{E}} &= P_k^{\mathcal{R}} H_k P_k^{\mathcal{E}} \\ H_k^{\mathcal{E}\mathcal{E}^\perp} &= P_k^{\mathcal{E}} H_k Q_k^{\mathcal{E}} & H_k^{\mathcal{E}\mathcal{F}^\perp} &= P_k^{\mathcal{E}} H_k Q_k^{\mathcal{F}} \end{aligned}$$

and so on, we can decompose the Hamiltonian fiber into blocks:

$$H_k = \begin{pmatrix} H_k^{\mathcal{E}\mathcal{E}} & H_k^{\mathcal{E}\mathcal{E}^\perp} \\ H_k^{\mathcal{E}^\perp\mathcal{E}} & H_k^{\mathcal{E}^\perp\mathcal{E}^\perp} \end{pmatrix} = \begin{pmatrix} H_k^{\mathcal{E}\mathcal{E}} & H_k^{\mathcal{E}\mathcal{R}} & H_k^{\mathcal{E}\mathcal{F}^\perp} \\ H_k^{\mathcal{R}\mathcal{E}} & H_k^{\mathcal{R}\mathcal{R}} & H_k^{\mathcal{R}\mathcal{F}^\perp} \\ H_k^{\mathcal{F}^\perp\mathcal{E}} & H_k^{\mathcal{F}^\perp\mathcal{R}} & H_k^{\mathcal{F}^\perp\mathcal{F}^\perp} \end{pmatrix}.$$

We note that our iterative diagonalisation as part of the band structure calculation indeed gives us access to a few eigenpairs $(\widetilde{\lambda}_{kn}, \widetilde{X}_{kn})$ of $H_k^{\mathcal{E}\mathcal{E}}$. Moreover due to Courant-Fisher $\lambda_{kn} \leq \widetilde{\lambda}_{kn}$, i.e. an upper bound to λ_{kn} is readily at hand. Suppose now we additionally had a lower bound μ_{kn}, i.e.

$$\mu_{kn} \leq \lambda_{kn} \leq \widetilde{\lambda}_{kn}$$

then

Selection deleted $$\delta_{kn} \geq \min\left(\mu_{kn} - \widetilde{\lambda}_{k,n-1},\, \mu_{k,n+1} - \widetilde{\lambda}_{k,n}\right) \qquad (3).$$

Finding such a suitable μ_{kn} provably is the goal of this task.

(a) Show that for Cohen-Bergstresser Hamiltonians and an appropriate choice of $\mathcal{F}$ depending on $\mathcal{E}$ (Task 2 (c)), the following structure of the Hamiltonian is obtained:

$$H_k = \begin{pmatrix} H_k^{\mathcal{E}\mathcal{E}} & V_k^{\mathcal{E}\mathcal{R}} & 0 \\ V_k^{\mathcal{R}\mathcal{E}} & H_k^{\mathcal{R}\mathcal{R}} & V_k^{\mathcal{R}\mathcal{F}^\perp} \\ 0 & V_k^{\mathcal{F}^\perp\mathcal{R}} & H_k^{\mathcal{F}^\perp\mathcal{F}^\perp} \end{pmatrix},$$

where the use of V instead of H indicates that the kinetic term does not contribute to the respective block and 0 indicates an all-zero block.

Solution (a):

Using the results from task **2 (b)**, namely how $\mathcal{F}$ is defined, we can conclude that indeed

$$H_k^{\mathcal{E}\mathcal{F}^\perp} = H_k^{\mathcal{E}\mathcal{F}^\perp} = 0$$

Furthermore,

$$\left(T_k^{\mathcal{R}\mathcal{E}}\right)_{GG'} = \langle e_G | T_k e_{G'} \rangle = \delta_{GG'} \frac{1}{2}|G+k|^2 = 0$$

since

$$\mathcal{E} < \frac{1}{2}|G+k|^2 < \mathcal{F} \quad \text{and} \quad \frac{1}{2}|G'+k|^2 < \mathcal{E}.$$

This means that the kinetic terms

$$T_k^{\mathcal{R}\mathcal{E}} = T_k^{\mathcal{R}\mathcal{E}} = 0$$

Analogically,

$$\left(T_k^{\mathcal{F}^\perp\mathcal{R}}\right)_{GG'} = \langle e_G | T_k e_{G'} \rangle = \delta_{GG'} \frac{1}{2}|G+k|^2 = 0$$

where

$$\mathcal{F} < \frac{1}{2}|G+k|^2 \quad \text{and} \quad \mathcal{E} < \frac{1}{2}|G'+k|^2 < \mathcal{F}.$$

Selection deleted

Therefore,

$$T_k^{\mathcal{F}^\perp\mathcal{R}} = T_k^{\mathcal{R}\mathcal{F}^\perp} = 0$$

We focus on the splitting of H_k into four blocks. An important realisation for our purpose is now that if $H_k - \mu_{kn}$ has exactly $n-1$ negative eigenvalues, then we necessarily have $\mu_{kn} < \lambda_{kn}$. In the following we will thus develop conditions on a parameter μ, such that $H_k - \mu$ has a provable number of negative eigenvalues.

A first step is the Haynsworth inertia additivity formula **[Hay1968]**. For a matrix

$$H_k - \mu = \begin{pmatrix} H_k^{\mathcal{E}\mathcal{E}} - \mu & V_k^{\mathcal{E}\mathcal{E}^\perp} \\ V_k^{\mathcal{E}^\perp\mathcal{E}} & H_k^{\mathcal{E}^\perp\mathcal{E}^\perp} - \mu \end{pmatrix}.$$

this theorem states that

$$N(H_k - \mu) = N(H_k^{\mathcal{E}\mathcal{E}} - \mu) + N(S_\mu)$$

where $N(\mathcal{A})$ denotes the number of negative eigenvalues of the operator $\mathcal{A}$ and S_μ is the Schur complement

$$S_\mu = \left(H_k^{\mathcal{E}^\perp\mathcal{E}^\perp} - \mu\right) - V_k^{\mathcal{E}^\perp\mathcal{E}}\left(H_k^{\mathcal{E}\mathcal{E}} - \mu\right)^{-1} V_k^{\mathcal{E}\mathcal{E}^\perp}.$$

[Hay1968]:

E. V. Haynsworth. Linear Algebra Appl. **1**, 73 (1968)

Selection deleted

(b) Using this statement explain why a $\mu \in (\widetilde{\lambda}_{k,n-1}, \widetilde{\lambda}_{kn})$ such that $S_\mu \geq 0$ is a guaranteed lower bound to λ_{kn}.

(c) We proceed to obtain conditions that ensure $S_\mu \geq 0$. Let $X = L^2_{\text{per}}(\Omega) \backslash \text{span}\left(\mathbb{B}_k^{\mathcal{E}}\right)$. Recall that $S_\mu \geq 0$ exactly if

$$\langle x, S_\mu x \rangle \geq 0 \qquad \forall x \in X$$

Prove the following statements:

- First $\forall x \in X$

  $$\left\langle x \,\middle|\, (H_k^{\mathcal{E}\mathcal{E}} - \mu)x \right\rangle \geq \mathcal{E} - \|V_k^{\mathcal{E}^\perp \mathcal{E}^\perp}\|_{\text{op}} - \mu,$$

  where $\|\cdot\|_{\text{op}}$ is the standard Hilbert operator norm

  $$\|\mathcal{A}\|_{\text{op}} = \sup_{0 \neq \varphi \in L^2_{\text{per}}(\Omega)} \frac{\langle \varphi, \mathcal{A}\varphi \rangle}{\langle \varphi, \varphi \rangle}$$

- Second, given a full eigendecomposition $H_k^{\mathcal{E}\mathcal{E}} = \widetilde{X}_k \widetilde{\Lambda}_k \widetilde{X}_k^H$ show that for $\mu \in I_n$ with the open interval

  $$I_n = \left(\frac{1}{2}\left(\widetilde{\lambda}_{k,n-1} + \widetilde{\lambda}_{kn}\right), \widetilde{\lambda}_{kn} \right)$$

  we have $\forall x \in X$:

  $$\begin{aligned} \left\langle x \,\middle|\, V_k^{\mathcal{E}^\perp \mathcal{E}} \left(H_k^{\mathcal{E}\mathcal{E}} - \mu\right)^{-1} V_k^{\mathcal{E}\mathcal{E}^\perp} x \right\rangle &\leq \left\| \left(V_k^{\mathcal{R}\mathcal{E}} \widetilde{X}_k\right) \left(\widetilde{\Lambda}_k - \mu\right)^{-1} \left(V_k^{\mathcal{R}\mathcal{E}} \widetilde{X}_k\right)^H \right\|_{\text{op}} \\ &\leq \frac{\|V_k\|_{\text{op}}^2}{\widetilde{\lambda}_{kn} - \mu} \end{aligned}$$

- Combining both show that

$$S_\mu \geq \mathcal{E} - \mu - l_1^V - \frac{(l_1^V)^2}{\widetilde{\lambda}_{kn} - \mu} \quad \text{where} \quad l_1^V = \sum_{|G| < \sqrt{2\mathcal{E}_V}} |\hat{V}(G)|. \tag{4}$$

Selection deleted

You may want to use Young's inequality, that is

$$\|V_k\|_{\text{op}} \leq \sum_{G \in \mathbb{L}^*} |\hat{V}(G)|.$$

Solution (c: 1):

First of all

$$H_k^{\mathcal{E}^\perp\mathcal{E}^\perp} = Q_k^{\mathcal{E}} H_k Q_k^{\mathcal{E}} = Q_k^{\mathcal{E}} T_k Q_k^{\mathcal{E}} + Q_k^{\mathcal{E}} V Q_k^{\mathcal{E}}$$

where the elements of the kinetic term are defined as the following

$$(T_k^{\mathcal{E}^\perp\mathcal{E}^\perp})_{GG'} = \langle e_G \,|\, T_k e_{G'} \rangle = \frac{1}{2}\delta_{GG'}|G+k|^2$$

It means that $T_k^{\mathcal{E}^\perp\mathcal{E}^\perp}$ is diagonal and we have:

$$\left\langle x \,\middle|\, T_k^{\mathcal{E}^\perp\mathcal{E}^\perp} x \right\rangle = \frac{1}{2}\delta_{GG'}|G+k|^2 \langle x \,|\, x\rangle \geq \mathcal{E}\langle x \,|\, x\rangle.$$

On the other hand,

$$\left|\left\langle x \,\middle|\, V_k^{\mathcal{E}^\perp\mathcal{E}^\perp} x \right\rangle\right| \leq \|x\| \|V_k^{\mathcal{E}^\perp\mathcal{E}^\perp} x\|$$

so that

$$-\|x\| \|V_k^{\mathcal{E}^\perp\mathcal{E}^\perp} x\| \leq \left\langle x \,\middle|\, V_k^{\mathcal{E}^\perp\mathcal{E}^\perp} x \right\rangle \leq \|x\| \|V_k^{\mathcal{E}^\perp\mathcal{E}^\perp} x\|$$

Therefore,

$$\left\langle x \,\middle|\, V_k^{\mathcal{E}^\perp\mathcal{E}^\perp} x \right\rangle \geq -\|x\| \|V_k^{\mathcal{E}^\perp\mathcal{E}^\perp} x\| \geq -\|x\| \|V_k^{\mathcal{E}^\perp\mathcal{E}^\perp}\|_{\mathrm{op}} \|x\| = -\|V_k^{\mathcal{E}^\perp\mathcal{E}^\perp}\|_{\mathrm{op}} \langle x \,|\, x\rangle$$

In the end, putting it all together and using Cauchy–Schwarz inequality, we have

$$\left\langle x \,\middle|\, (H_k^{\mathcal{E}^\perp\mathcal{E}^\perp} - \mu) x \right\rangle \geq \left(\mathcal{E} - \|V_k^{\mathcal{E}^\perp\mathcal{E}^\perp}\|_{\mathrm{op}} - \mu\right) \langle x \,|\, x\rangle \geq \left(\mathcal{E} - \|V_k^{\mathcal{E}^\perp\mathcal{E}^\perp}\|_{\mathrm{op}} - \mu\right)$$

since vector x are normalized.

Selection deleted

Solution (c: 2):

Using a full eigendecomposition $H_k^{\mathcal{EE}} = \widetilde{X}_k \widetilde{\Lambda}_k \widetilde{X}_k^H$ we can express $\left(H_k^{\mathcal{EE}} - \mu\right)^{-1}$ as following

$$\left(H_k^{\mathcal{EE}} - \mu\right)^{-1} = \widetilde{X}_k \Big(\widetilde{\Lambda}_k - \mu\Big)^{-1} \widetilde{X}_k^H,$$

and applying the same logic as above the inner product can be bounded above by the operator norm:

$$\left\langle x \,\middle|\, V_k^{\mathcal{E}^\perp \mathcal{E}} \left(H_k^{\mathcal{EE}} - \mu\right)^{-1} V_k^{\mathcal{EE}^\perp} x \right\rangle \leq \left\| V_k^{\mathcal{E}^\perp \mathcal{E}} \widetilde{X}_k \Big(\widetilde{\Lambda}_k - \mu\Big)^{-1} \widetilde{X}_k^H V_k^{\mathcal{EE}^\perp} \right\|_{\text{op}}$$

Since we work with Cohen-Bergstresser Hamiltonians and $\widetilde{X}_k$ are unitary, applying the sub-multiplicative property of the operator norm we get:

$$\begin{aligned} \left\| V_k^{\mathcal{E}^\perp \mathcal{E}} \widetilde{X}_k \Big(\widetilde{\Lambda}_k - \mu\Big)^{-1} \widetilde{X}_k^H V_k^{\mathcal{EE}^\perp} \right\|_{\text{op}} &= \left\| \left(V_k^{\mathcal{RE}} \widetilde{X}_k\right) \Big(\widetilde{\Lambda}_k - \mu\Big)^{-1} \left(V_k^{\mathcal{RE}} \widetilde{X}_k\right)^H \right\|_{\text{op}} \leq \\ &\leq \left\| V_k^{\mathcal{RE}} \right\|_{\text{op}} \left\| \Big(\widetilde{\Lambda}_k - \mu\Big)^{-1} \right\|_{\text{op}} \left\| V_k^{\mathcal{RE}} \right\|_{\text{op}} \end{aligned}$$

Since $V_k^{\mathcal{RE}}$ is part of the Hamiltonian and the operator norm of the $H_k^{\mathcal{EE}}$ is bounded by its spectral radius:

$$\left\| V_k^{\mathcal{RE}} \right\|_{\text{op}} \leq \left\| H_k^{\mathcal{EE}} \right\|_{\text{op}}$$

Finally, considering the definition of the interval I_n, we can conclude that $\mu < \widetilde{\lambda}_{kn}$. Therefore,

$$\left\| \left(V_k^{\mathcal{RE}} \widetilde{X}_k\right) \Big(\widetilde{\Lambda}_k - \mu\Big)^{-1} \left(V_k^{\mathcal{RE}} \widetilde{X}_k\right)^H \right\|_{\text{op}} \leq \frac{\|V_k\|_{\text{op}}^2}{\widetilde{\lambda}_{kn} - \mu}$$

Selection deleted

Solution (c: 3):

We will start with the Young's inequality for the operator norm:

$$\|V_k\|_{\text{op}} \leq \sum_{|G|<\sqrt{2\mathcal{E}_V}} |\hat{V}(G)| = l_1^V.$$

Then for the first inequality:

$$\left\langle x \,\middle|\, (H_k^{\mathcal{E}^\perp\mathcal{E}^\perp} - \mu)x \right\rangle \geq \mathcal{E} - \|V_k^{\mathcal{E}^\perp\mathcal{E}^\perp}\|_{\text{op}} - \mu \geq \mathcal{E} - l_1^V - \mu$$

For the second inequality:

$$\left\langle x \,\middle|\, V_k^{\mathcal{E}^\perp\mathcal{E}} \left(H_k^{\mathcal{E}\mathcal{E}} - \mu\right)^{-1} V_k^{\mathcal{E}\mathcal{E}^\perp} x \right\rangle \leq \frac{\|V_k\|_{\text{op}}^2}{\widetilde{\lambda}_{kn} - \mu} \leq \frac{(l_1^V)^2}{\widetilde{\lambda}_{kn} - \mu}$$

Finally,

$$\begin{aligned}
\langle x, S_\mu x\rangle &= \left\langle x, \left(\left(H_k^{\mathcal{E}^\perp\mathcal{E}^\perp} - \mu\right) - V_k^{\mathcal{E}^\perp\mathcal{E}} \left(H_k^{\mathcal{E}\mathcal{E}} - \mu\right)^{-1} V_k^{\mathcal{E}\mathcal{E}^\perp}\right) x \right\rangle = \\
&= \left\langle x, \left(H_k^{\mathcal{E}^\perp\mathcal{E}^\perp} - \mu\right) x \right\rangle - \left\langle x, V_k^{\mathcal{E}^\perp\mathcal{E}} \left(H_k^{\mathcal{E}\mathcal{E}} - \mu\right)^{-1} V_k^{\mathcal{E}\mathcal{E}^\perp} x \right\rangle. \\
&\geq \mathcal{E} - l_1^V - \mu - \frac{(l_1^V)^2}{\widetilde{\lambda}_{kn} - \mu}
\end{aligned}$$

Selection deleted

The strategy to determine μ is thus as follows:

1. Compute l_1^V. For this we need a Fourier representation of V. A real space representation is available as `Vreal = DFTK.total_local_potential(ham)` where `ham` is a `DFTK.Hamiltonian`, which can be transformed into Fourier space using a fast-fourier transform `fft(basis, kpoint, Vreal)` where `basis` is the `ham.basis` and `kpoint` is the `Kpoint` object matching the currently processed k. Note that this assumes that $\mathcal{E}_V < \mathcal{E}$.
2. Using the lower bound (4) find the largest $\mu \in I_n$ for which it can be ensured that $S_\mu \geq 0$. Since we are not using the exact S_μ here, but only a lower bound, this step can fail (i.e. both possible solutions μ are outside of I_n). In this case we are unable to obtain a lower bound for $\widetilde{\lambda}_{kn}$ and thus unable to obtain a Kato-Temple estimate.
3. If 2. goes through we set $\mu_{kn} = \mu$ and thus have guaranteed bounds $\mu_{kn} \leq \lambda_{kn} \leq \widetilde{\lambda}_{kn}$.

(d) Focusing on the first two eigenpairs of the Γ-point of the Cohen-Bergstresser model, plot both the guaranteed lower bound μ_{kn} from above procedure as well as the non-guarandeed bound of Task 4 as you increase $\mathcal{E}$. Take the computation of $\widetilde{\lambda}_{nk}$ for $\mathcal{E} = 80$ as the reference. Which bound is sharper ? While it should be emphasised that better guaranteed bounds than our development are possible (see for example **[HLC2020]**), can you comment on the advantages and disadvantages of guaranteed error bounds ?

[HLC2020]:

> M. F. Herbst, A. Levitt and E. Cancès. Faraday Discuss., **224**, 227 (2020). DOI 10.1039/D0FD00048E

Selection deleted

```
MethodError: no method matching fft(::DFTK.PlaneWaveBasis{Float64, Float64, DFTK.CPU, Array{StaticArraysCore.SVector{3, Int64}, 3}, Array{StaticArraysCore.SVector{3, Float64}, 3}, Vector{StaticArraysCore.SVector{3, Int64}}}, ::DFTK.Kpoint{Float64, Vector{StaticArraysCore.SVector{3, Int64}}}, ::Array{ComplexF64, 4})
Closest candidates are:
fft(::DFTK.PlaneWaveBasis, ::DFTK.Kpoint, !Matched::AbstractArray{T, 3} where T; kwargs...)
@ DFTK C:\Users\User\.julia\packages\DFTK\HG5Ae\src\fft.jl:114
fft(!Matched::AbstractArray{<:Real}, ::Any)
@ AbstractFFTs C:\Users\User\.julia\packages\AbstractFFTs\4iQz5\src\definitions.jl:214
fft(!Matched::AbstractArray{<:Complex{<:Union{Integer, Rational}}}, ::Any)
@ AbstractFFTs C:\Users\User\.julia\packages\AbstractFFTs\4iQz5\src\definitions.jl:216
...
```

1. **top-level scope** @ *Local: 8* [inlined]

```
begin
    local Ecut = 5
    local basis = PlaneWaveBasis(model; Ecut=Ecut, kgrid=(1, 1, 1))
    local ham = Hamiltonian(basis)
    local Vreal = ComplexF64.(DFTK.total_local_potential(ham))

    kpoint = basis.kpoints[1]
    Vfourier = fft(ham.basis, kpoint, Vreal;) #kpoint,

    l1_V = sum(abs.(Vfourier))

    # eigres = diagonalize_all_kblocks(DFTK.lobpcg_hyper, ham, n_bands=6)
    # λkn = eigres.λ[1][1:2]

end
```

Task 6: Band structure with guaranteed bounds

Based on the results so far compute a band structure with guaranteed error bars for $\mathcal{E} = 7$ a k-point density of 15 and the 6 first bands (i.e. the 6 lowest eigenpairs). For each eigenvalue:

- Estimate the **algorithm** and **arithmetic error** by re-computing the in-basis residual $P_k^{\mathcal{E}} r_{kn}$ using `Double64`. For this you need to build a basis and a Hamiltonian that employes `Double64` as the working precision. To change DFTK's internal working precision follow the Arbitrary floating-point types documentation page. From a `basis_double64` using `Double64` precision a corresponding Hamiltonian is obtained by `Hamiltonian(basis_double64)` as shown before. *Hint:* Before computing products `Hamiltonian(basis_double64) * X`, ensure that `X` has been converted to `Double64` as well.
- Estimate the **discretisation error** using both the Bauer-Fike estimates of Tasks 2 & 3 as well as the Kato-Temple estimates of Task 5. In the band structure annotate the tightest bound that is available to you.

Selection deleted

```
1 Enter cell code...
```

Selection deleted

UndefVarError: `basisD64` not defined

1. **top-level scope** @ *Local: 11*

```
begin

    λ_conv_double64=[]
    X_conv_double64=[]
    res_norm_double64=[]
    ham_list_double64=[]
    ρ_bauer_fike_double64 = Vector{Vector{Float64}}()
    println(ρ_bauer_fike_double64)


    ham_double64 = Hamiltonian(basisD64)
    eigres_double64 = diagonalize_all_kblocks(DFTK.lobpcg_hyper, ham_double64, 6)

    #large_base3=basis_change_Ecut(band_data_double64.basis,20)
    #ham_large3=Hamiltonian(large_base3)
    #eigres_large_3 = diagonalize_all_kblocks(DFTK.lobpcg_hyper, ham_large3, 6)


    for (ik, kpt) in enumerate(bands_double64.basis.kpoints)




        hamk = ham_double64[ik]
        #hamk_large= ham_large3[ik]

        λk   = eigres_double64.λ[ik]
        Xk   = eigres_double64.X[ik]

        #λk_large   = eigres_large_3.λ[ik]
        #Xk_large   = eigres_large_3.X[ik]

        residual_k = hamk * Xk - Xk * Diagonal(λk)
        column_norms = [norm(residual_k[:,i]) for i in 1:6]

        print(size(residual_k))
        println(norm.(residual_k))
        println(typeof(norm.(residual_k)))
        println(size(norm.(residual_k)))

        push!(ham_list_double64,hamk)
        push!(ρ_bauer_fike_double64,column_norms)
        push!(λ_conv_double64,λk)
        push!(X_conv_double64,Xk)
        push!(res_norm_double64,norm(residual_k))
    end
    println(ρ_bauer_fike_double64)

end
```

Selection deleted

```
Vector{Float64}[]
```

Adaptive diagonalisation techniques

The main purpose of *a posteriori* error bounds as we have developed in the previous tasks is to understand the error in the current computational setup. A good error bound is sharp, cheap to compute and ideally guaranteed, i.e. the actual error is always smaller. One can thus think of the error bars as safety checks. If one is not yet satisfied with the current quality of results, one can always choose a more accurate numerical setup and re-compute. For example if the estimated discretisation error is too large, one just increases $\mathcal{E}$ until one is satisfied.

As simple as this procedure is, it has one notable drawback: It leads to a nested set of loops, where for each attempted $\mathcal{E}$ (outer loop), we iteratively diagonalise the H_k using LOBPCG (inner loop). A natural question is thus, whether one can fuse these two loops, i.e. adapt $\mathcal{E}$ *while the LOBPCG is converging*, such that at each step the algorithm error and discretization error are balanced.

To make this idea more clear, let us recall a basic LOBPCG implementation, which in this case only performs a diagonalisation of the Hamiltonian fiber at the gamma point of a passed model:

Selection deleted

nonadaptive_lobpcg (generic function with 1 method)

```julia
function nonadaptive_lobpcg(model::Model{T}, Ecut, n_bands;
                            maxiter=100, tol=1e-6, verbose=false) where {T}
    kgrid = (1, 1, 1)  # Γ point only
    basis = PlaneWaveBasis(model; Ecut, kgrid)
    ham   = Hamiltonian(basis)
    hamk  = ham[1]                  # Select Γ point
    prec  = PreconditionerTPA(hamk)  # Initialise preconditioner
    X     = DFTK.random_orbitals(hamk.basis, hamk.kpoint, n_bands)

    converged = false
    λ = NaN
    residual_norms = NaN
    residual_history = []

    P = zero(X)
    R = zero(X)
    for i in 1:maxiter
        if i > 1
            Z = hcat(X, P, R)
        else
            Z = X
        end
        Z = Matrix(qr(Z).Q)  # QR-based orthogonalisation

        # Rayleigh-Ritz
        HZ = hamk * Z
        λ, Y = eigen(Hermitian(Z' * HZ))
        λ = λ[1:n_bands]
        Y = Y[:, 1:n_bands]
        new_X = Z * Y

        # Compute residuals and convergence check
        R = HZ * Y - new_X * Diagonal(λ)
        residual_norms = norm.(eachcol(R))
        push!(residual_history, residual_norms)
        verbose && @printf "%3i %8.4g %8.4g\n" i λ[end] residual_norms[end]
        if maximum(residual_norms) < tol
            converged = true
            X .= new_X
            break
        end

        # Precondition and update
        DFTK.precondprep!(prec, X)
        ldiv!(prec, R)
        P .= X - new_X
        X .= new_X

        # Additional step:
        # Move to larger basis ?
    end

    (; λ, X, basis, ham, converged, residual_norms, residual_history)
```

Selection deleted

```
end
```

This implementation can be easily compared to the `DFTK.lobpcg_hyper` function of DFTK. For example:

```
[-6.4615e-14, -2.65676e-13, -8.01026e-14, -6.45428e-13, -1.68643e-13, -4.84579e-12]
```

```
let
    Ecut     = 10
    n_bands  = 6
    basis    = PlaneWaveBasis(model; Ecut=10, kgrid=(1, 1, 1))
    ham      = Hamiltonian(basis)
    res_dftk = diagonalize_all_kblocks(DFTK.lobpcg_hyper, ham, n_bands)
    res_dftk_λ_Γ = res_dftk.λ[1]  # Extract eigenvalues at Γ point

    # Notice: This function always only computes one k-Point
    res_nonadaptive = nonadaptive_lobpcg(model, Ecut, n_bands)

    res_nonadaptive.λ - res_dftk_λ_Γ
end
```

In order to make this routine discretisation-adaptive we need to add an additional step as indicated in the above source code: After we have checked for convergence and as we are updating `X`, `P` and `R` to their new values, we might additionally want to increase the discretisation basis to employ a new, increased cutoff value $\mathcal{E} + \Delta$ to discretise the Hamiltonian. Using

```
transfer_blochwave(X, basis_small, basis_small.kpoints[1],
                      basis_large, basis_large.kpoints[1])
```

we can then transfer the state vectors `X`, `P` and `R` to this basis.

This means while the iterative eigensolver is converging we also improve the discretisation basis by successively increasing $\mathcal{E}$. Each LOBPCG iteration may thus run using a different discretisation basis.

Selection deleted

Task 7: Developing a discretisation-adaptive LOBPCG

Assume the LOBPCG currently runs at cutoff $\mathcal{F}$. Due to the properties of the Cohen-Bergstresser Hamiltonian (Task 2 (b)) there is a smaller cutoff $\mathcal{E} < \mathcal{F}$ such that (Task 5 (a))

$$Q_k^{\mathcal{F}} H P_k^{\mathcal{E}} = 0.$$

(a) Prove that if $\|Q_k^{\mathcal{E}} \widetilde{X}_{kn}\| = 0$, then

$$|\lambda_{kn} - \widetilde{\lambda}_{kn}| \leq \|P_k^{\mathcal{F}} r_{kn}\|,$$

i.e. that the total error is driven to zero as the iterative diagonalisation employing a cutoff $\mathcal{F}$ is converging. Based on this argue why a large value for $\|Q_k^{\mathcal{E}} \widetilde{X}_{kn}\|$ relative to $\|P_k^{\mathcal{F}} r_{kn}\|$ provides a reasonable indicator to decide when to refine the discretisation.

(b) Following the idea of (a) code up an adaptive LOBPCG by monitoring the ratio

$$\frac{\|Q_k^{\mathcal{E}} \widetilde{X}_{kn}\|}{\|P_k^{\mathcal{F}} r_{kn}\|} \qquad (5).$$

Note, that one way to compute $Q_k^{\mathcal{E}} \widetilde{X}_{kn}$ is as $\widetilde{X}_{kn} - P_k^{\mathcal{E}} \widetilde{X}_{kn}$ where $P_k^{\mathcal{E}}$ can be obtained by transferring from the large basis to the small basis and back to the large basis, i.e.

```
X_small = transfer_blochwave(X_large,
                             basis_large, basis_large.kpoints[1],
                             basis_small, basis_small.kpoints[1])
P_X     = transfer_blochwave(X_small,
                             basis_small, basis_small.kpoints[1],
                             basis_large, basis_large.kpoints[1])
```

Whenever (5) becomes too large, you should switch to a finer discretisation basis, e.g. by switching from a cutoff $\mathcal{E}$ to $\mathcal{E} + \Delta$. Experiment with this setup to find what are good ratios (5) to indicate switching and good values for Δ. Some strategy for exploration:

- Assume we eventually want to target a cutoff $\mathcal{F}^{\text{final}} = 80$. Save the convergence history for running the full LOBPCG at exactly this cutoff. This is your reference.
- With your adaptive methods you should try to loose as little as possible in the rate of convergence compared to the reference convergence profile, but try to use as small values of $\mathcal{F}$ for as many iterations as you can.
- To develop your approach, first only compute a single eigenvalue (`n_bands = 1`) and then start considering more than one. Note, that you will need to adapt (5) and take apropriate maxima / minima over all computed eigenpairs to decide when to switch to the next bigger basis.
- Be creative and experiment. In this Task there is no "best" solution.

Selection deleted

Solution (a):

Using the definition of $\mathcal{F}$ from task 2 (b), triangular inequality, and assuming $\widetilde{X}_{kn}$ to be normalized we have

$$\begin{aligned}|\lambda_{kn} - \widetilde{\lambda}_{kn}| = |\lambda_{kn} - \widetilde{\lambda}_{kn}| \left\|\widetilde{X}_{kn}\right\| \leq \\ \leq \|r_{kn}\| + \|H_k\widetilde{X}_{kn} - \lambda_{kn}\widetilde{X}_{kn}\| \approx \\ \approx \|r_{kn}\| = \|P_k^{\mathcal{F}} r_{kn}\|,\end{aligned}$$

where

$$r_{kn} = H_k\widetilde{X}_{kn} - \widetilde{\lambda}_{kn}\widetilde{X}_{kn}.$$

If the ratio (5) is large it indicates that the approximate eigenvector $\widetilde{X}_{kn}$ has significant components outside the subspace spanned by $P_k^{\mathcal{E}}$ meaning that the chosen cutoff $\mathcal{E}$ may be insufficient to capture the eigenvector.

Solution (b):

Selection deleted

```
adaptive_lobpcg (generic function with 1 method)
```

```julia
function adaptive_lobpcg(model::Model{T}, Ecut, n_bands;
                         maxiter=100, tol=1e-6, verbose=false, threshold=10)
where {T}
    kgrid = (1, 1, 1)  # Γ point only
    basis = PlaneWaveBasis(model; Ecut, kgrid)
    ham   = Hamiltonian(basis)
    hamk  = ham[1]                  # Select Γ point
    prec  = PreconditionerTPA(hamk)  # Initialise preconditioner
    X     = DFTK.random_orbitals(hamk.basis, hamk.kpoint, n_bands)

    converged = false
    λ = NaN
    residual_norms = NaN
    residual_history = []
    Ecut_new = Ecut


    basis_prev = basis

    P = zero(X)
    R = zero(X)
    for i in 1:maxiter
        if i > 1
            Z = hcat(X, P, R)
        else
            Z = X
        end
        Z = Matrix(qr(Z).Q)  # QR-based orthogonalisation

        # Rayleigh-Ritz
        HZ = hamk * Z
        λ, Y = eigen(Hermitian(Z' * HZ))
        λ = λ[1:n_bands]
        Y = Y[:, 1:n_bands]
        new_X = Z * Y

        # Compute residuals and convergence check
        R = HZ * Y - new_X * Diagonal(λ)
        residual_norms = norm.(eachcol(R))
        push!(residual_history, residual_norms)
        verbose && @printf "%3i %8.4g %8.4g\n" i λ[end] residual_norms[end]
        if maximum(residual_norms) < tol
            converged = true
            X .= new_X
            break
        end

        # Precondition and update
        DFTK.precondprep!(prec, X)
        ldiv!(prec, R)
        P .= X - new_X
        X .= new_X

```

```
        # Additional step:
        basis_small  = basis_change_Ecut(basis_prev, Ecut_new - 3)
        X_E = transfer_blochwave_kpt(X,
                          basis_prev, basis_prev.kpoints[1],
                          basis_small, basis_small.kpoints[1])
        P_X = transfer_blochwave_kpt(X_E,
                          basis_small, basis_small.kpoints[1],
                          basis_prev, basis_prev.kpoints[1])


        # Computing the ratio
        ratio = norm(X - P_X) / norm(residual_norms)

        # Moving to a larger basis
        if ratio > threshold

            Ecut_new = Ecut_new + 2
            println("i = ", i, ", Ecut = ", Ecut_new)
            basis_new = PlaneWaveBasis(model; Ecut=Ecut_new, kgrid)

            X   = transfer_blochwave_kpt(X, basis_prev, basis_prev.kpoints[1],
                      basis_new, basis_new.kpoints[1])
            P   = transfer_blochwave_kpt(P, basis_prev, basis_prev.kpoints[1],
                      basis_new, basis_new.kpoints[1])
            R   = transfer_blochwave_kpt(R, basis_prev, basis_prev.kpoints[1],
                      basis_new, basis_new.kpoints[1])

            hamk  = Hamiltonian(basis_new)[1]
            prec  = PreconditionerTPA(hamk)
            basis_prev = basis_new

        end

    end

    (; λ, X, basis, ham, converged, residual_norms, residual_history)
end
```

[-0.079228, 0.384299, 0.384299, 0.384299, 0.510143, 0.510143]

```
begin
    local Ecut     = 5
    local n_bands  = 6

    res_adaptive = adaptive_lobpcg(model, Ecut, n_bands, threshold=0.2)
    res_adaptive.λ
end
```

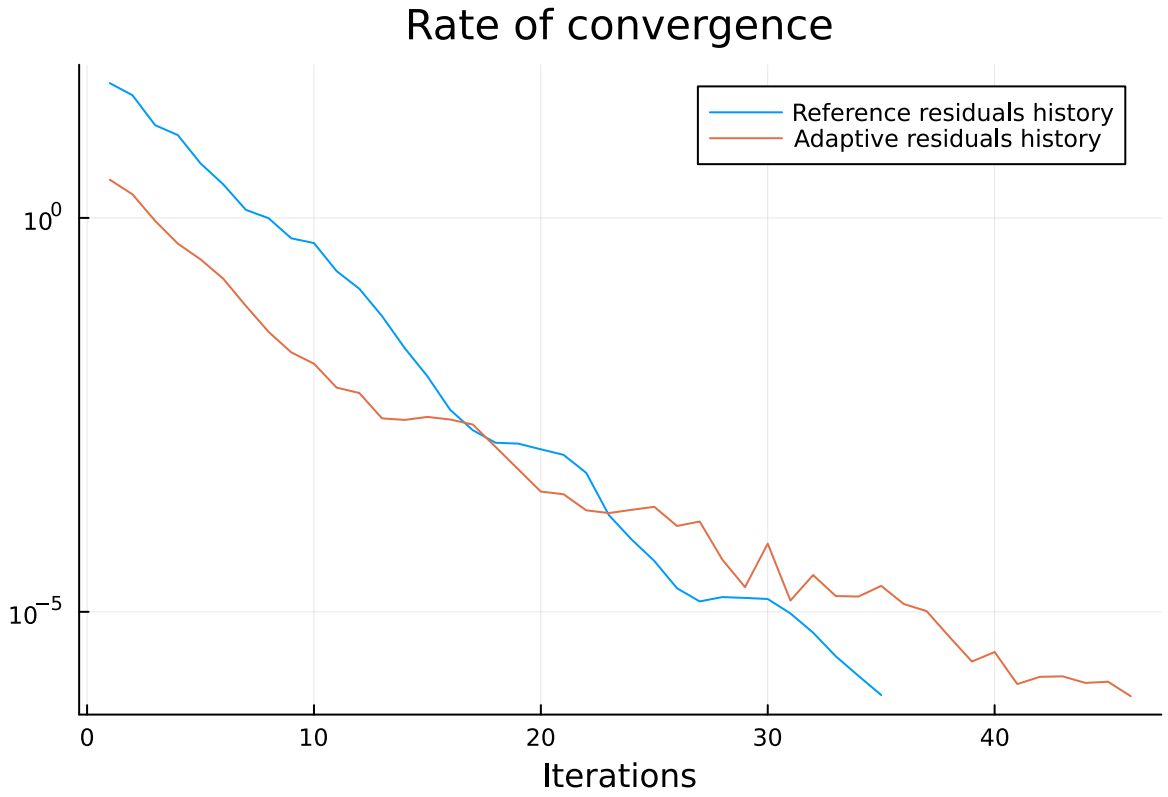
```
i = 1, Ecut = 7
i = 6, Ecut = 9
i = 9, Ecut = 11
i = 11, Ecut = 13
i = 13, Ecut = 15
i = 20, Ecut = 17
i = 26, Ecut = 19
i = 29, Ecut = 21
i = 31, Ecut = 23
i = 36, Ecut = 25
i = 39, Ecut = 27
i = 41, Ecut = 29
```

[-0.079228, 0.384299, 0.384299, 0.384299, 0.510143, 0.510143]

```
begin
    local Ecut     = 80
    local n_bands  = 6

    res_nonadaptive = nonadaptive_lobpcg(model, Ecut, n_bands)
    res_nonadaptive.λ
end
```

[-0.0791827, 0.389126, 0.389126, 0.389126, 0.512338, 0.512338]

```
begin
    local Ecut     = 3
    local n_bands  = 6

    res_nonadaptive5 = nonadaptive_lobpcg(model, Ecut, n_bands)
    res_nonadaptive5.λ
end
```




```
begin
    local p = plot(yaxis=:log, xlabel="Iterations")
    plot!(1:size(res_nonadaptive.residual_history)[1], norm.
    (res_nonadaptive.residual_history), label="Reference residuals history")

    plot!(1:size(res_adaptive.residual_history)[1], norm.
    (res_adaptive.residual_history), label="Adaptive residuals history")
    title!(p, "Rate of convergence")
end
```

Experimenting with various thresholds for the ratio given in equation (5) and values of Δ we can see that the rate of convergence using the adaptive method, with an appropriately set threshold and Δ, can effectively match the reference convergence profile. Moreover, the developed approach has better computational efficiency by using smaller cutoffs during the early steps in the algorithm.

GTH pseudopotentials

To close off this project we will now consider slightly more realistic potentials than the Cohen-Bergstresser model. We will look at the local part of the Goedecker-Teter-Hutter pseudopotentials[GTH96]. Neglecting some minor details involving the structure of the system, these potentials can be understood in Fourier space as

$$\hat{V}(p) = P(p)\exp(-p^2/\sigma^2),$$

i.e. a strongly decaying Gaussian times a polynomial prefactor. Instead of just a few non-zero Fourier coefficients like in the Cohen-Bergtresser case, we thus have the weaker setting, that the Fourier cofficients only strongly decay as $|G| \to \infty$. Unlike our development in Task 2 (b), we thus cannot find an elevated cutoff $\mathcal{F} > \mathcal{E}$, such that the residual can be computed exactly. However, due to the strong decay the truncated residual $P_k^{\mathcal{F}} r_{kn}$ of Tasks 2 becomes a better and better approximation to r_{kn} the larger $\mathcal{E}$ and $\mathcal{F}$ are taken. Thus the ideas of Tasks 2 can still be employed to compute *approximate* errors for this GTH potential — though without any guarantees that the actual error is smaller than these estimates.

[GTH96]:
S. Goedecker, M. Teter, and J. Hutter Phys. Rev. B **54**, 1703 (1996). DOI 10.1103/PhysRevB.54.1703

Task 8: An error indicator for GTH pseudopotentials

A setup for the GTH pseudopotential model, discretised for a provided cutoff $\mathcal{E}$ and using only the Γ-point is given by the function

```
make_gth_basis (generic function with 1 method)
```

```
function make_gth_basis(Ecut)
    Si = ElementPsp(:Si, psp=load_psp("hgh/lda/si-q4"))
    atoms = [Si, Si]
    positions = [ones(3)/8, -ones(3)/8]
    lattice = 5 .* [[0 1 1.]; [1 0 1.]; [1 1 0.]]
    model = Model(lattice, atoms, positions; terms=[Kinetic(), AtomicLocal()])
    PlaneWaveBasis(model; Ecut, kgrid=(1, 1, 1))
end
```

Assume a secondary cutoff $\mathcal{F} > \mathcal{E}$, which is large enough that the error bounds developed in Task 2 are suffiently good to be useful. In other words we will assume that the residual computed in $\mathbb{B}_k^{\mathcal{F}}$, i.e.

$$P_k^{\mathcal{F}} r_{kn} = H_k^{\mathcal{F}\mathcal{F}} \widetilde{X}_{kn} - \widetilde{\lambda}_{kn} \widetilde{X}_{kn},$$

is a good estimate of the true residual.

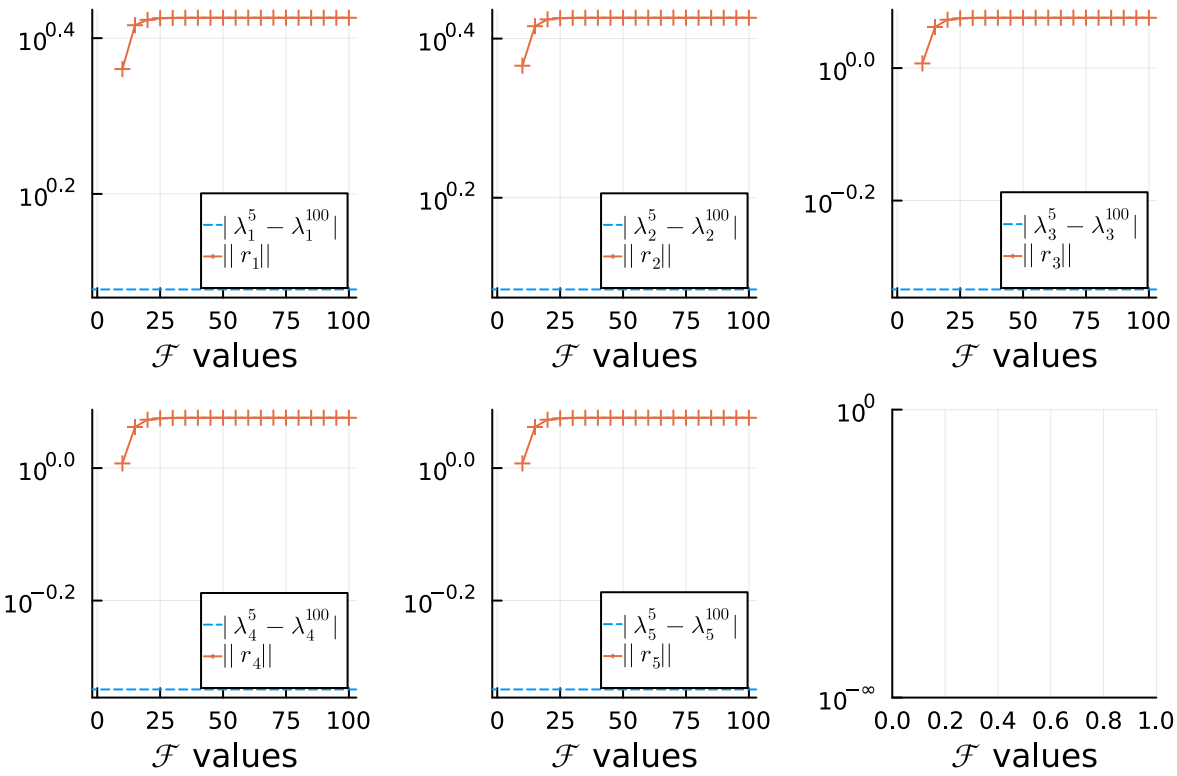
Focusing on the first 5 eigenvalues of the Γ point of the GTH Hamiltonian, investigate the Bauer-Fike error bounds based on this approximate residual. Investigate in particular:

(a) Take $\mathcal{E} = 100$ as a reference and plot the error bound for $\mathcal{E} = 5$ and for varying $\mathcal{F}$ between 10 and 100. Does the bound always seem to hold ? Does it converge with $\mathcal{F}$?

(b) For a few offsets Δ between 5 and 30 set $\mathcal{F} = \mathcal{E} + \Delta$ and consider the 2nd and 3rd eigenvalue. Plot both the obtained error estimate as well as the error of the eigenvalue as you vary $\mathcal{E}$ between 5 and 50. Again take $\mathcal{E} = 100$ as a reference. What offset Δ would you recommend based on this investigation ?

Solution (a):

```
begin
    n_bands_8a = 5
    local ik = 1

    Fcuts = 10:5:100
    ρ_BF = zeros(length(Fcuts), n_bands_8a)

    basis_100   = make_gth_basis(100)
    ham_100     = Hamiltonian(basis_100)
    eigres_100  = diagonalize_all_kblocks(DFTK.lobpcg_hyper, ham_100, n_bands_8a)
    λ_100       = eigres_100.λ[ik]

    basis_5     = make_gth_basis(5)
    ham_5       = Hamiltonian(basis_5)
    eigres_5    = diagonalize_all_kblocks(DFTK.lobpcg_hyper, ham_5, n_bands_8a)
    λ_5         = eigres_5.λ[ik]

    r_ref = abs.(λ_5 - λ_100)

    for (i, Fcut) in enumerate(Fcuts)
        basis   = make_gth_basis(Fcut)
        ham     = Hamiltonian(basis)
        X_F     = transfer_blochwave(eigres_5.X, basis_5, basis)

        residual_k = ham[ik] * X_F[ik] - X_F[ik] * Diagonal(eigres_5.λ[ik])
        ρ_BF[i, :] = norm.(eachcol(residual_k))

    end

end
```



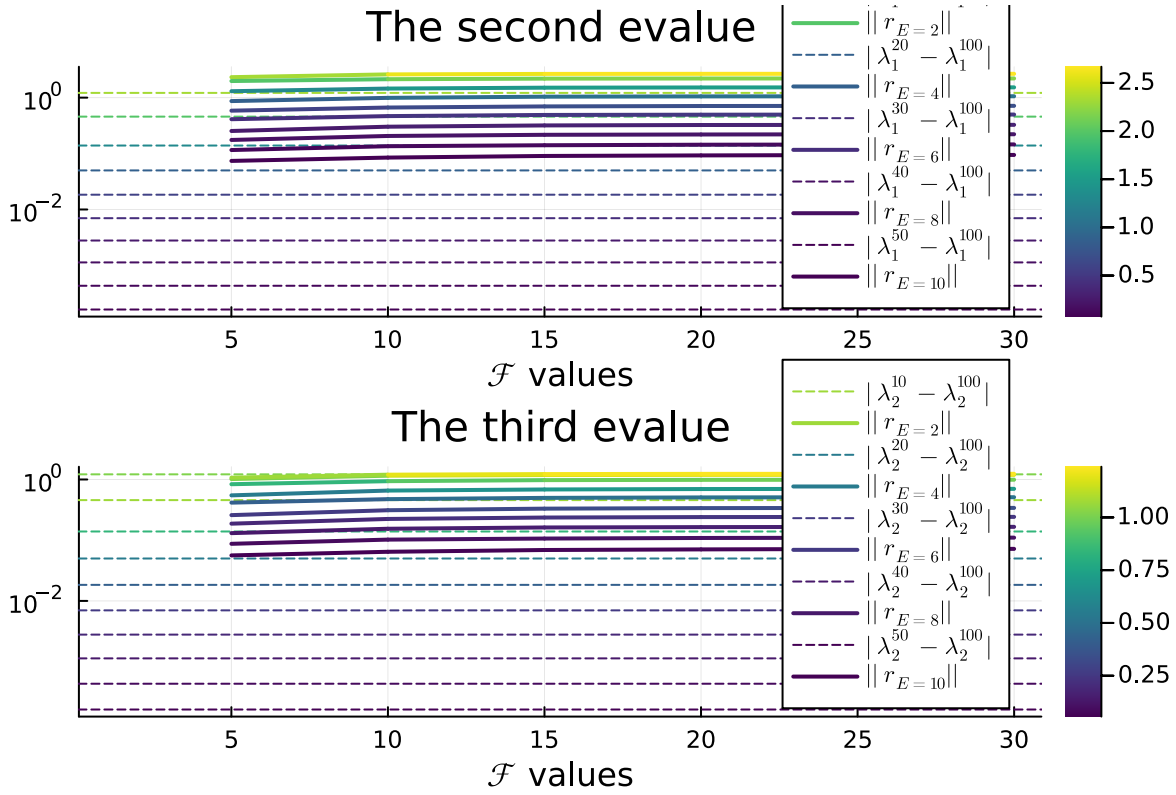

```
begin

    local p = plot(yaxis=:log, layout=(2, 3),
                    empty_at = 6, xlabel=L"$\mathcal{F}$ values")

    for i in 1:n_bands_8a

        hline!([r_ref[i]], subplot=i, line=:dash,
                label=string(latexstring("|λ^5_$(i) - λ^{100}_$(i)|")))
        plot!(p, Fcuts, ρ_BF[:, i], subplot=i, shape=:cross,
                label=string(latexstring("||r_$(i)||")))

    end
    plot!(legend=:bottomright)
end
```

```
No strict ticks found

No strict ticks found
```

Increasing $\mathcal{F}$ we can see that all bounds are converging and always are bigger than the reference value.

Solution (b):

```
begin
    Delta_range = 5:5:30
    E_range= 5:5:50
    local n_bands = 3
    e_BF    = zeros(length(E_range), length(Delta_range), 2)
    e_ref   = zeros(length(E_range), 2)

    for (i, E) in enumerate(E_range)
        basis_E     = make_gth_basis(E)
        ham_E       = Hamiltonian(basis_E)
        eigres_E    = diagonalize_all_kblocks(DFTK.lobpcg_hyper, ham_E, n_bands)

        e_ref[i, :] = abs.(eigres_E.λ[ik][2:3] - λ_100[2:3])

        for (j, Δ) in enumerate(Delta_range)
            F = E + Δ
            basis   = make_gth_basis(F)
            ham     = Hamiltonian(basis)
            X_F     = transfer_blochwave(eigres_E.X, basis_E, basis)

            res_k = ham[ik] * X_F[ik] - X_F[ik] * Diagonal(eigres_E.λ[ik])
            e_BF[i, j, :] = norm.(eachcol(res_k))[2:3]
        end
    end
end
```




```
begin
    gradient = [:blue, :red]
    local p = plot(yaxis=:log, layout=(2, 1), xlabel=L"$\mathcal{F}$ values")

    for i in 1:2

        for (j, E) in enumerate(E_range)

            if E % 10 == 0
                label_h = string(latexstring("|λ^{$(E)}_$(i) - λ^{100}_$(i)|"))
                label_r = string(latexstring("||r_{E = $(j)}||"))
            else
                label_h = ""
                label_r = ""
            end

            hline!([e_ref[j]], subplot=i, line=:dash,
                    line_z=e_BF[j, : , i],
                    color=cgrad(:viridis),
                    label=label_h
            )

            plot!(p, Delta_range, e_BF[j, : , i],
                    subplot=i, linewidth = 2,
                    line_z=e_BF[j, : , i],
                    color=cgrad(:viridis),
                    label=label_r
            )
        end
    end

    plot!(legend=:bottomright)
```

```
    plot!(title="The second evalue", subplot=1)
    plot!(title="The third evalue", subplot=2)

end
```