

ON THE COMPUTATION OF MAXIMALLY LOCALIZED WANNIER FUNCTIONS

VERSION 1.1

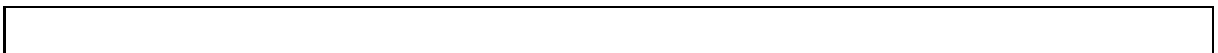
DIPLOMARBEIT

VON

A N D R E A S K L Ö C K N E R

BETREUT VON

P R O F E S S O R D R . W I L L Y D Ö R F L E R



Erklärung

Die selbständige und eigenhändige Anfertigung dieser Arbeit versichere ich an Eides statt. Alle verwendeten Hilfsmittel und Quellen sind im Anhang angegeben.

Karlsruhe, den 4. Oktober 2004

Andreas Klöckner

Anschrift:

Andreas Klöckner
Im Türmle 11
78224 Singen

e-mail:

`andreas@tiker.net`

Contents

1	Introduction	5
1.1	What and why	5
1.2	Acknowledgements	6
1.3	Notation	6
1.4	Directory of symbols	7
1.5	Version history	8
2	Eigenproblems with periodic coefficients	9
2.1	Motivation	9
2.2	Elementary definitions	10
2.3	The Floquet transform	10
2.4	Some consequences	15
2.5	Bands and gaps	17
2.6	Smoothness of the dispersion relation	18
3	Wannier functions	21
3.1	Definition	21
3.2	Localization of Wannier functions: Basics	22
3.3	Wannier functions as a basis set	22
3.4	Localization methods	23
3.5	The location operator in \mathbf{k} -space	24
4	Localization in \mathbf{k}-space	27
4.1	The spread functional	27
4.2	A mesh in \mathbf{k} -space	29
4.3	Finite difference formulae in \mathbf{k} -space	29
4.4	The discretized spread functional	30
4.5	A Problem and its Solution	31
4.6	Decomposition of the new spread functional	33
4.7	The gradient of the spread functional	34
4.7.1	A straightforward approach to the gradient	35
4.7.2	Small changes to $U^{\mathbf{k}}$	36
4.7.3	A first gradient of Ω	39
4.7.4	Marzari and Vanderbilt's gradient of Ω	41
4.8	Minimizing the spread	42
4.8.1	The starting strategy	43
4.8.2	The initial inner products $M^{\mathbf{k},\mathbf{b},(0)}$	44
4.8.3	Updating the inner product matrices	45
4.8.4	Other implementation notes	45
5	Implementation	47
5.1	PyLinear	48
5.2	PyAngle	50
5.3	FemPy	51
5.3.1	The user interface	51

5.3.2	FemPy's inner workings	52
5.3.3	Numerical experiments	54
5.3.4	Further work	58
5.4	PyWannier	58
5.4.1	Weak formulation of the eigenproblem	58
5.4.2	Discretization of the eigenproblem	59
5.4.3	The way to maximally localized Wannier functions	62
5.4.4	Future work	63
6	Results	65
6.1	Constant permittivity in one dimension	65
6.2	The non-constant case in one dimension	65
6.3	Constant permittivity in two dimensions	69
6.4	The non-constant case in two dimensions	71
6.5	Maximally localized Wannier functions	74
7	Conclusions and Future Work	79
A	Auxiliary results	81

Chapter 1

Introduction

1.1 What and why

This thesis is about photonic crystals and some tools used to simulate them. But what exactly are photonic crystals, and why would you want to spend time learning about them? In a nutshell, many people believe that photonic crystals could be the “next semiconductors”. When Eli Yablonovitch and his coworkers at Bell Communications Research in New Jersey drilled their first holes into a block of dielectric material in 1991, they were trying to do with electromagnetic waves what is commonly being done with electrons in silicon chips. Photons travel, literally, at the speed of light, while electrons have a lower rate of signal propagation. So, Yablonovitch and his team hoped to accelerate things by using light instead of electric current. Light is already being used for transport of information, and photonic crystals could be the first step to the processing of information (i.e. computation) using light.

But before this can happen, there are many hurdles to overcome. For example, light escapes very easily. Unlike electrons that can only go where you put a cable, light happily spreads just about anywhere, even in vacuum. Photonic crystals already solve this problem, as we will see in our simulations. The next step would be building photonic devices like filters, switches, memories and logic gates—areas where lots of research is currently being done.

Unfortunately, photonic crystals are devilishly hard to fabricate. The structures need to have roughly the same scale as the wavelength with which they are going to be used. So, microwave crystals (on the millimeter scale, such as Yablonovitch’s sample) are relatively easy to make by hand. But for visible light (around 300-600 nanometers), manufacturing such crystals poses severe problems. As Greg Parker and Martin Charlton put it in their August 2000 *PhysicsWorld* feature article, “If the scale was 1000 times smaller, we could build the structure atom-by-atom using a chemical reaction; and if it was 1000 times larger, we could build the structure mechanically.”

This is why simulation is so important. If you are going to go through the significant effort of building a structure, you’d like to know in advance that it will work the way you have planned. But the tried and true workhorses of electromagnetics simulation, the FDTD (finite difference time domain) method and the plane wave method, both perform suboptimally for large-scale simulations, because both do not adapt well to the piecewise continuous permittivity layouts commonly found in photonic crystals. The finite element method has been used with some success ([AK99], [Dob99]) and fares slightly better, but it also runs into problems for large supercell simulations. Clearly, it would be advantageous if we could find a better approximation basis than any of the basis sets provided by these methods.

To this end, we can gain insight from the analogy between “real” crystals and photonic crystals. Maximally localized Wannier functions [MV97] have been used quite successfully by physicists as basis sets in large-scale ab-initio simulations of crystal structures. Busch and his team [BMGM⁺03] as well as Whittaker and Croucher [WC03] were the first to carry this approach over to photonic crystals.

This thesis aims to be a readable and coherent introduction to the theory and the computational means involved in the construction of maximally localized Wannier functions. While it was my goal to be as mathematically rigorous as possible, this is not pursued at all costs. For example, a few results which point deeper into distribution theory are simply stated and not proven. In a similar balance between readability and complexity, we restrict ourselves to two (instead of the full three) dimensions. Many of the ideas presented in the text were also tried out computationally, using a custom finite element code

which was created solely for this purpose. The implementation is described and its results are presented. Finally, we will evaluate what was achieved and provide directions and pointers for further work.

The cover page shows the dispersion relation of a homogeneous two-dimensional medium. It may be found within this work, with more annotations, as Figure 6.5a).

1.2 Acknowledgements

I would like to take this opportunity to thank all the people without whom this thesis would have never happened. First and foremost, that includes my advisor Prof. Dr. Willy Dörfler, who directed my attention to this exciting topic and spent many days supporting me in my efforts. Especially through the more difficult parts of this work, his help was invaluable.

My parents also deserve a huge thank you—their encouragement and support all the way through my studies were what made this possible in the first place. Dirk Wäldin, Christopher Poulton, Christian Koos, Matthias Schillinger, Sergei Mingaleev and Simon Gemmrich endured my questions and shared their insights in many enlightening discussions.

Besides the tools mentioned in the implementation chapter, the preparation of this document involved the programs TEX_{MACS} (which I can only recommend), Grace, Gnuplot, gl2ps, TEX , VTK and MayaVi, all of which have been released under open-source or Free Software licenses. Without these tools, my life as a writer would have been significantly harder. I owe their creators a great deal.

Last, but not least, I would like to thank my girlfriend Josie, who took part very intimately in the highs and lows of my work and put up with my moods when things didn't go the way they were supposed to. Without her perseverance and constant encouragement, I would have never left square one. Also, she proofread the text and fixed my broken English. If despite her efforts any errors remain, they are most certainly mine.

Thank you all!

1.3 Notation

Vectors are distinguished from scalars by bold type, e.g. \mathbf{b} vs. b . The corresponding non-bold character denotes the Euclidean (i.e. 2-)norm of that vector, so that $b = |\mathbf{b}|_2$. Discretized entities are distinguished from their analytic counterparts by a superscript “ Δ ”. For example, a discretized gradient operator would be written “ ∇^Δ ”.

Given a matrix A , the symbol A^H denotes the complex-hermitian transpose of A , and A^T means the real transpose of A . Given a function, scalar or matrix A , a superscript asterisk A^* denotes the complex conjugate of A .

1.4 Directory of symbols

<i>Symbol</i>	<i>Meaning</i>
$\Delta f(t)$	The expression $f(t + \Delta t) - f(t)$, where Δt should be clear from context.
$\nabla^2 f(\mathbf{r})$	The Laplace operator $\nabla \cdot \nabla f(\mathbf{r})$.
$\delta(\cdot)$	The Dirac Delta distribution in $\mathbf{0}$.
$\delta_{n,m}$	The Kronecker symbol.
d	The dimensionality of the problem. Throughout this thesis, $d = 2$.
$L \subset \mathbb{R}^d$	The direct lattice.
$\hat{L} \subset \mathbb{R}^d$	The reciprocal lattice.
$\mathcal{B}(L') \subset \mathbb{R}^d$	A basis of a lattice L' . (mostly $L' \in \{L, \hat{L}\}$)
$\mathbf{r} \in \mathbb{R}^d$	A real space vector.
$\mathbf{R} \in L$	A direct lattice vector.
$\mathbf{k} \in \mathbb{R}^d$	A reciprocal space vector.
$\mathbf{K} \in \hat{L}$	A reciprocal lattice vector.
\mathcal{K}	The Brillouin zone mesh, as in [MP76]. See Section 4.2.
S	The stencil of nearest-neighbor vectors in \mathcal{K} .
$P \subset \mathbb{R}^d$	A real-space primitive cell of L , for example the Wigner-Seitz cell.
$B \subset \mathbb{R}^d$	The first Brillouin zone of the crystal under consideration.
λ	The real-space Lebesgue measure on \mathbb{R}^d .
$\partial P(\mathbf{R}) \subset \partial P$	The \mathbf{R} -boundary of P . See Section 2.2.
$\psi_{n,\mathbf{k}}(\cdot)$	The n th Bloch function with crystal momentum \mathbf{k} .
$u_{n,\mathbf{k}}(\cdot)$	A periodic Bloch function $u_n(\mathbf{k}, \mathbf{r}) := e^{-i\mathbf{k} \cdot \mathbf{r}} \psi_n(\mathbf{k}, \mathbf{r})$.
$w_n(\cdot)$	The n th (generalized) Wannier function at $\mathbf{0}$.
$w_{n,\mathbf{R}}(\cdot)$	The n th (generalized) Wannier function at \mathbf{R} , $w_{n,\mathbf{R}}(\mathbf{x}) := w_n(\mathbf{x} - \mathbf{R})$.
$\varepsilon(\mathbf{r})$	The L -periodic permittivity function.
$\langle \cdot, \cdot \rangle_{\nabla \Omega}$	The inner product on the Ω -gradient space. See Section 4.7.
$\langle u, f(\mathbf{r})v \rangle_{\mathbb{R}^d}$	The inner product $\int_{\mathbb{R}^d} u(\mathbf{r})\varepsilon(\mathbf{r})f^*(\mathbf{r})v^*(\mathbf{r})d\mathbf{r}$.
$\langle u, f(\mathbf{r})v \rangle_P$	The inner product $\int_P u(\mathbf{r})\varepsilon(\mathbf{r})f^*(\mathbf{r})v^*(\mathbf{r})d\mathbf{r}$.
$L_\varepsilon^2(\Omega)$	See Definition 2.1.
$\mathcal{U}f$	The Floquet transform. See Theorem 2.2.

1.5 Version history

Oct 4, 2004	Preliminary version.
Oct 5, 2004	Final version. Changes vs. preliminary in <code>errata.tm</code> .
Oct 18, 2004	Online version.
	Added a missing π in Figure 6.1.
	Split off big images.
	Removed an extraneous “b)” in Figure 6.14.
Oct 20, 2004	Added a few missing indices to the ρ_i of Section 5.3.3.
	Changed a “quantitatively” to “qualitatively” in Section 6.3.
Nov 4, 2004	Changed a leftover \mathcal{P} to the correct γ in Section 6.3.
Nov 9, 2004	Removed a leftover empty inner product in Section 4.5.
Nov 18, 2004	Replaced “differentiable operator” by “differential operator” in Sec. 2.3.
Nov 21, 2004	Fixed a bug in a script that made Grace figures locale-dependent.
Dec 16, 2004	Removed an extraneous prime in Section 2.2, added one in Section 2.3.
Mar 7, 2005	Corrected $tW \rightarrow tW$ in Section 4.7.
May 23, 2005	Fixed the sign of $\operatorname{Re} \operatorname{tr} [tW^k R^{k,b}]$ in the last few formulae of Section 4.7. (The end result remains unaffected.)
July 21, 2005	Eliminated references to a “bug” in Marzari’s method. (Chapters 4, ??)
	Added more insight on the gradient of $\operatorname{Re} \operatorname{tr}[AB]$. (Section 4.7)
	Added a clarification with regard to ∇ and ∇^2 to Section 2.1.
	Reorganized and clarified Section 4.8.
	“Version 1.1” to clarify that there have been significant changes.

Chapter 2

Eigenproblems with periodic coefficients

In this chapter, we will set the mathematical stage for the remainder of this thesis.

2.1 Motivation

At the most basic level, photonic crystals can be understood as infinite three-dimensional media of periodically varying electric permittivity, and our main concern in this thesis shall be the computation of the natural *electric* ($\mathbf{E}(\mathbf{r}, t)$) and *magnetic* ($\mathbf{H}(\mathbf{r}, t)$) field modes associated with these crystals. Maxwell's Equations are an apt mathematical model of our situation—quantum effects need not be considered since we are typically working on a macroscopic scale. We will constrain our search for solutions to Maxwell's Equations in a few ways: First, we will only be looking for time-harmonic fields. Second, we restrict the given (*relative*) *permittivity* $\varepsilon(\mathbf{r})$ to be a scalar and independent of the third space coordinate, effectively making our material *linear and isotropic* and our problem two-dimensional. Finally, the (*relative*) *magnetic permeability* μ is assumed to be 1 everywhere.

The first simplification allows us to separate the time variable: $\mathbf{E}(\mathbf{r}, t) = e^{i\omega t}\mathbf{E}(\mathbf{r})$ and $\mathbf{H}(\mathbf{r}, t) = e^{i\omega t}\mathbf{H}(\mathbf{r})$. This leads us to an eigenvalue problem of the form

$$\begin{aligned} -\nabla \times \mathbf{E}(\mathbf{r}) &= \frac{i\omega}{c}\mathbf{H}(\mathbf{r}), \\ \nabla \times \mathbf{H}(\mathbf{r}) &= \frac{i\omega}{c}\varepsilon(\mathbf{r})\mathbf{E}(\mathbf{r}), \\ \nabla \cdot \mathbf{E}(\mathbf{r}) = 0, \quad \nabla \cdot \mathbf{H}(\mathbf{r}) &= 0. \end{aligned}$$

for $\mathbf{r} \in \mathbb{R}^3$. A closer look at our second simplification above reveals that for reasons of symmetry we would also expect \mathbf{E} and \mathbf{H} to not depend on the third space coordinate either, so we can assume $\partial_3\mathbf{E} = \mathbf{0}$ and $\partial_3\mathbf{H} = \mathbf{0}$. If we consider the vector $(E_1, E_2, E_3, H_1, H_2, H_3)$, it is easy to see that in our case the system above decomposes into a direct sum of two simpler ones, namely for $(0, 0, \psi := E_3, H_1, H_2, 0)$, we get

$$-\nabla^2\psi(\mathbf{r}) = \frac{\omega^2}{c^2}\varepsilon(\mathbf{r})\psi(\mathbf{r}) \tag{2.1}$$

(a form called TM for *Transverse Magnetic*). Likewise, for $(E_1, E_2, 0, 0, 0, \psi := H_3)$ we get

$$-\nabla \cdot \left(\frac{1}{\varepsilon(\mathbf{r})}\nabla\psi(\mathbf{r}) \right) = \frac{\omega^2}{c^2}\psi(\mathbf{r}) \tag{2.2}$$

(a form called TE for *Transverse Electric*). In both (2.1) and (2.2), it is understood that ∇^2 (and ∇ , respectively) only refer to the first two spatial dimensions, such that

$$\nabla^2 = \partial^2/\partial x_1^2 + \partial^2/\partial x_2^2 \quad \text{and} \quad \nabla = (\partial^2/\partial x_1^2, \partial^2/\partial x_2^2)^T.$$

In the setting of photonic crystals, the TE case tends to be slightly harder since the permittivity ε often has jump discontinuities, requiring the use of distribution theory to properly define the divergence operator in (2.2). Therefore, we will concentrate on the TM problem (2.1).

2.2 Elementary definitions

In this section, we will briefly introduce a few basic terms in reference to the three-dimensional theory of crystalline solids. Since we will be dealing with two-dimensional structures in the remainder, the obvious reductions of these terms apply.

Given some linearly independent vectors $\{\mathbf{R}_1, \mathbf{R}_2, \mathbf{R}_3\}$, we define the associated *Bravais Lattice* to be

$$L := \{n_1\mathbf{R}_1 + n_2\mathbf{R}_2 + n_3\mathbf{R}_3 : n_i \in \mathbb{Z}(i = 1, 2, 3)\}$$

and call $\mathcal{B}(L) := \{\mathbf{R}_1, \mathbf{R}_2, \mathbf{R}_3\}$ its *lattice basis*. Note that the origin $\mathbf{0}$ is naturally contained in any Bravais lattice. A function f is said to be *L-periodic* if for any $\mathbf{R} \in L$ and any $\mathbf{r} \in \mathbb{R}^3$, the identity $f(\mathbf{r}) = f(\mathbf{r} + \mathbf{R})$ holds. A *primitive unit cell* is an open set $P \subset \mathbb{R}^3$ such that

$$\overline{\bigcup_{\mathbf{R} \in L} \{P + \mathbf{R}\}} = \mathbb{R}^3,$$

where the union is disjoint. Whenever a reference is made to the letter P from now on, it is meant that the statement is valid for any choice of primitive unit cell. The *Wigner-Seitz cell* P_W is one particular choice of such a cell:

$$P_W := \{\mathbf{r} \in \mathbb{R}^3 : |\mathbf{r}| < |\mathbf{r} + \mathbf{R}| \text{ for } \mathbf{R} \in L \setminus \{\mathbf{0}\}\}.$$

Verbally, the Wigner-Seitz cell is the set of all $\mathbf{r} \in \mathbb{R}^3$ which are closer to the origin than to any other lattice point.

The *reciprocal lattice* \hat{L} is a theoretical construct that proves to be of great utility in solid state theory. It naturally arises in many situations, for example when calculating crystal diffraction or studying *L*-periodic functions. It is defined as the set of vectors $\mathbf{K} \in \mathbb{R}^3$ for which $e^{i\mathbf{K} \cdot \mathbf{R}} = 1$ (or equivalently $\mathbf{K} \cdot \mathbf{R} = 2\pi n$ for some $n \in \mathbb{Z}$), for all $\mathbf{R} \in L$. It turns out that the reciprocal lattice is a Bravais lattice itself, and its basis can be written

$$\begin{aligned} \mathcal{B}(\hat{L}) &= \{\mathbf{K}_1, \mathbf{K}_2, \mathbf{K}_3\}, \\ \text{with } \mathbf{K}_i &\text{ determined by } \mathbf{K}_i \cdot \mathbf{R}_j = 2\pi\delta_{i,j} \quad (i, j \in \{1, 2, 3\}). \end{aligned}$$

The nine equations given by the above condition are sufficient to specify the nine unknown coordinates of \mathbf{K}_i uniquely.

The Wigner-Seitz cell of the reciprocal lattice is called the *First Brillouin Zone* B (or just *Brillouin Zone*). Typically, vectors in the Brillouin zone (or, more generally, in reciprocal space) are named \mathbf{k} .

For a unit cell P , the symbol $\partial P(\mathbf{R})$ for $\mathbf{R} \in L$, called the *\mathbf{R} -boundary of the unit cell*, is

$$\partial P(\mathbf{R}) := \{\mathbf{r} \in \partial P : \mathbf{r} + \mathbf{R} \in \partial P\}.$$

To clarify this notation, we note that, as is apparent from intuition,

$$\bigcup_{\mathbf{R} \in L \setminus \{-1, 1\}} [\partial P(\mathbf{R}) \cup (\partial P(\mathbf{R}) + \mathbf{R})] = \partial P.$$

The union is over a set of representatives from L such that for a pair $\{-\mathbf{R}, \mathbf{R}\}$, only one is contained in the set. The union is actually disjoint except for corner points of the primitive unit cell.

Finally, let $\mathbf{R}_P(\mathbf{r})$ be the $\mathbf{R} \in L$ such that $\mathbf{r} - \mathbf{R}_P(\mathbf{r}) \in P$. Note that this notation is not well-defined for boundary points unless a provision is made to include one ‘‘half’’ of the boundary in P . This shortcoming shall not matter to us, because our main use of the symbol $\mathbf{R}_P(\mathbf{r})$ is within integrals over \mathbf{r} , to which the set $\partial P + L$ of all such boundary points constitutes a zero set.

A lot more detail about many of these definitions, as well as motivation and theoretical background information can be found in the introductory book [AM76].

2.3 The Floquet transform

Obviously, a problem like Equation (2.1) is computationally and analytically hard to tackle, not just because it lives on the entire real plane. But since our permittivity function ε is *L*-periodic, we might

hope that solving an equation resembling (2.1) on just one primitive unit cell P with appropriate boundary conditions might be enough, and the solution on all the remaining translations of the primitive unit cell might just be a copy of this solution, in some sense. We will see that this hope is not for naught.

Most of the theory surrounding the eigenvalue problem (2.1) with periodic coefficients has been known for decades, Chapter XIII.16 of Reed and Simon's book [RS78] contains an excellent summary. Large parts of this section are patterned after their work. However, we need to make a small adaptation to take into account the fact that the investigation there concentrates on Schrödinger operators of the form $-\nabla^2 + V$ with a potential V , while our TM operator has the slightly different shape $-\nabla^2/\varepsilon$. First, we need some notation.

Definition 2.1 Let $\varepsilon : P \rightarrow [\varepsilon_-, \varepsilon_+]$ with $0 < \varepsilon_- \leq \varepsilon_+ < \infty$ be piecewise continuously differentiable. Then let

$$\langle \varphi, \psi \rangle_{\mathbb{R}^d} := \int_{\mathbb{R}^d} \varphi(\mathbf{r}) \psi^*(\mathbf{r}) \varepsilon(\mathbf{r} - \mathbf{R}_P(\mathbf{r})) d\mathbf{r},$$

and call the resulting Hilbert space $L_\varepsilon^2(\mathbb{R}^d)$. Define $\langle \cdot, \cdot \rangle_P$ and $L_\varepsilon^2(P)$ analogously, but just on the primitive unit cell P .

Note that the mapping

$$\begin{aligned} \Phi_\Omega : L^2(\Omega) &\rightarrow L_\varepsilon^2(\Omega) \\ \psi &\mapsto \Phi_\Omega \psi := \psi / \sqrt{\varepsilon} \end{aligned}$$

is a Hilbert space isometry between $L^2(\Omega)$ and $L_\varepsilon^2(\Omega)$, because

$$\langle \Phi_\Omega \varphi, \Phi_\Omega \psi \rangle_{L_\varepsilon^2(\Omega)} = \langle \varphi / \sqrt{\varepsilon}, \psi / \sqrt{\varepsilon} \rangle_{L_\varepsilon^2(\Omega)} = \int_\Omega \frac{\varphi(\mathbf{r})}{\sqrt{\varepsilon(\mathbf{r})}} \varepsilon(\mathbf{r}) \frac{\psi^*(\mathbf{r})}{\sqrt{\varepsilon(\mathbf{r})}} d\mathbf{r} = \langle \varphi, \psi \rangle_{L^2(\Omega)}.$$

Be mindful that Φ_Ω 's definition, and thus the equivalence of the norms, hinges critically on ε being bounded from below by a positive constant.

By our motivation, we are looking for some map that reduces a function on the whole space down to one on a primitive unit cell. The *Floquet transform*, defined below, assumes this role.

Theorem 2.2 Define a transform \mathcal{U} on $\mathcal{S}(\mathbb{R}^d)$ by

$$(\mathcal{U}f)_\mathbf{k}(\mathbf{r}) := \sum_{\mathbf{R} \in L} e^{i\mathbf{k} \cdot \mathbf{R}} f(\mathbf{r} - \mathbf{R}).$$

Then \mathcal{U} 's domain may be extended to all of $L_\varepsilon^2(\mathbb{R}^d)$, and it becomes a unitary operator

$$\mathcal{U} : L_\varepsilon^2(\mathbb{R}^d) \rightarrow L^2(B \times L_\varepsilon^2(P)),$$

where $L^2(B \times L_\varepsilon^2(P))$ has the inner product

$$\langle \varphi, \psi \rangle_{L^2(B \times L_\varepsilon^2(P))} = \frac{1}{\lambda(B)} \int_B \langle \varphi_\mathbf{k}, \psi_\mathbf{k} \rangle_P d\mathbf{k}.$$

Proof Let U be as in Theorem XIII.97 of [RS78]. Then it is clear that $\mathcal{U} = \Phi_P \circ U \circ \Phi_{\mathbb{R}^d}^{-1}$ properly defines \mathcal{U} on all of $L_\varepsilon^2(\mathbb{R}^d)$ —where it is understood that the final Φ_P must be defined as

$$\begin{aligned} \Phi_P : L^2(B \times L^2(P)) &\rightarrow L^2(B \times L_\varepsilon^2(P)) \\ \psi &\mapsto (\Phi_P \psi), \quad (\Phi_P \psi)_\mathbf{k} := \left(\underbrace{\Phi_P}_{\text{on } L^2(P)} \psi_\mathbf{k} \right)_\mathbf{k}. \end{aligned}$$

Since each of the operators is an isometry, so is their composition \mathcal{U} . The form of the inner product is a consequence of the space's role as a “constant fiber direct integral” with the measure defined by Reed and Simon.

Note: $\mathcal{S}(\mathbb{R}^d)$ is a space of rapidly decreasing functions used mostly in the construction of the Fourier Transform. See [Rud91], Chapter 6 for a definition. \square

It is very important to understand the shape of \mathcal{U} 's target space. Each $\psi \in L^2(B \times L_\varepsilon^2(P))$ actually consists of many functions $\psi_{\mathbf{k}}$, one for each Brillouin zone vector $\mathbf{k} \in B$. Each of these $\psi_{\mathbf{k}} \in L_\varepsilon^2(P)$ is a function $\psi_{\mathbf{k}} : P \rightarrow \mathbb{C}$ for which $\langle \psi_{\mathbf{k}}, \psi_{\mathbf{k}} \rangle_P < \infty$ holds. This is almost the same as being square-integrable, with the small change of the $\varepsilon(\mathbf{r})$ inserted into the inner product (cf. Definition 2.1). Each ψ can itself be understood as a function with values in $L_\varepsilon^2(P)$. Now we demand that this function $\psi : B \rightarrow L_\varepsilon^2(P)$ be square-integrable itself, in the sense of the inner product defined above. It is probably helpful to note that this inner product is really nothing more than a Brillouin zone average of the inner products on the “small spaces” for each \mathbf{k} .

Now is a good time to take a short break from developing more theory and to consider our present situation. Instead of solving the eigenvalue problem (2.1) on $L_\varepsilon^2(\mathbb{R}^d)$, we will transform it by \mathcal{U} and solve it on $L^2(B \times L_\varepsilon^2(P))$. If the eigenproblems on each of the $L_\varepsilon^2(P)$ turn out to be independent of each other, then we have just made a big step forward. In that case, we may simply solve many small problems (one for each $\mathbf{k} \in B$) on easily-controlled bounded domains P instead of one big problem on an uncontrollably huge domain \mathbb{R}^d .

So, contrary to the hope uttered above, it will not be enough to solve just one small problem on *one* primitive unit cell P and then stamp that solution all over \mathbb{R}^d . Instead, we have to deal with *infinitely many* small problems, solve each individually, and then we can find the whole-space solution (if it exists) by \mathcal{U}^{-1} . This is not as simple as we had dared to hope, but it is still much easier than our initial problem.

Above, we hinted that the component functions $\psi_{\mathbf{k}}$ of a Floquet-transformed function ψ may have to obey special conditions at the boundary of P . These conditions are already determined by the definition of the Floquet transform. Assume $\mathbf{R}' \in \mathcal{B}(L)$ and $\mathbf{r} \in \partial P(\mathbf{R}')$.

$$\begin{aligned} (\mathcal{U}f)_{\mathbf{k}}(\mathbf{r} + \mathbf{R}') &= \sum_{\mathbf{R} \in L} e^{i\mathbf{k} \cdot \mathbf{R}} f(\mathbf{r} + \mathbf{R}' - \mathbf{R}) \\ (\text{let } \mathbf{R}'' := \mathbf{R} - \mathbf{R}') &= \sum_{\mathbf{R}'' \in L} e^{i\mathbf{k} \cdot (\mathbf{R}'' + \mathbf{R}')} f(\mathbf{r} - \mathbf{R}'') \\ &= e^{i\mathbf{k} \cdot \mathbf{R}'} (\mathcal{U}f)_{\mathbf{k}}(\mathbf{r}). \end{aligned}$$

This means that the functions $\psi_{\mathbf{k}} \in L_\varepsilon^2(P)$ have to satisfy the boundary condition $\psi_{\mathbf{k}}(\mathbf{r} + \mathbf{R}) = e^{i\mathbf{k} \cdot \mathbf{R}} \psi_{\mathbf{k}}(\mathbf{r})$. In fact, these boundary conditions are not yet strong enough. Just like with “regular” periodic boundary conditions, we will need a similar condition on the first derivative to ensure the self-adjointness of our differential operator.

If you think about it, these boundary conditions are quite natural. Recall that a plane wave can be described as $e^{i\mathbf{k} \cdot \mathbf{r}}$ with a *wave vector* \mathbf{k} . The wave vector's direction gives the propagation direction of the wave, while its length can be seen as something akin to the wave's frequency. Our boundary conditions basically dictate that the eigenfunctions associated with the Brillouin zone vector \mathbf{k} behave a little bit like plane waves with a wave vector \mathbf{k} . From this interpretation it may be understandable that \mathbf{k} is also often called the *crystal momentum*.

Right now, we can also understand why the \mathbf{k} remain constrained to the Brillouin zone and don't need to go beyond. Given $\mathbf{k} \in B$ and $\mathbf{K} \in \hat{L}$, the boundary conditions given by $\mathbf{k} + \mathbf{K}$ are exactly the same as the conditions for \mathbf{k} alone. By the defining property of the reciprocal lattice \hat{L} ,

$$e^{i(\mathbf{k} + \mathbf{K}) \cdot \mathbf{R}} = e^{i\mathbf{k} \cdot \mathbf{R}} \underbrace{e^{i\mathbf{K} \cdot \mathbf{R}}}_{=1} = e^{i\mathbf{k} \cdot \mathbf{R}}.$$

Now, before we use the Floquet transform on our differential operator, we should be clear about what exactly we are doing. The Floquet transform is defined on L^2 -like spaces which do not lend themselves well to the definition of differential operators. However, for example, $C_0^\infty(\mathbb{R})$ is dense in $L_\varepsilon^2(\mathbb{R})$, and there are no problems defining differential operators on C_0^∞ . (Section 7.8 of [Sch81] proves a slightly stronger result for just $L^2(\mathbb{R})$.) Therefore, we will consider all such derivatives defined on a dense subset of L_ε^2 , or simply *densely defined* in L_ε^2 . For this definition to make sense, the differential operator also needs to be *closable*, so that different sequences in the dense subset which converge to the same limit point always have the same limit under the operator. Here, we will simply assert this for our operators and again refer to the literature for details.

The following theorem gives us the Floquet decomposition of the differential operator in Equation (2.1).

Theorem 2.3 *Let ε be as in Definition 2.1. Then the whole-space TM operator from (2.1) is decomposed by the Floquet transform into*

$$\mathcal{U} \left(-\frac{\nabla^2}{\varepsilon} \right) \mathcal{U}^{-1} = \frac{1}{\lambda(B)} \int_B^\oplus H(\mathbf{k}) d\mathbf{k},$$

with $H(\mathbf{k}) := -\nabla^2/\varepsilon$ on $L_\varepsilon^2(P)$ under the boundary conditions

$$\varphi(\mathbf{r} + \mathbf{R}) = e^{i\mathbf{k} \cdot \mathbf{R}} \varphi(\mathbf{r}) \quad \text{and} \quad \nabla \varphi(\mathbf{r} + \mathbf{R}) \cdot \mathbf{n} = e^{i\mathbf{k} \cdot \mathbf{R}} \nabla \varphi(\mathbf{r}) \cdot \mathbf{n} \quad (2.3)$$

for $\mathbf{r} \in \partial P(\mathbf{R})$ and any $\mathbf{R} \in L$ and a unit-length vector \mathbf{n} normal to ∂P in \mathbf{r} .

This is analogous to part (b) of Theorem XIII.97 in [RS78], with \mathcal{U} exchanged for U . The boundary conditions in Equation (2.3) are called *Floquet boundary conditions*.

This last theorem gives us what we asked for: we can solve the eigenvalue problem for each \mathbf{k} independently. Next, we would like to find out more about the spectrum and the eigenfunctions of $H(\mathbf{k})$. As an intermediate result, we need the following Lemma.

Lemma 2.4 *The operator $H(\mathbf{k})$ on $L_\varepsilon^2(P)$ is self-adjoint and has a compact resolvent.*

Proof $H(\mathbf{k})$ is Hermitian: For $\varphi, \psi \in D(H(\mathbf{k}))$, by Green's Second Identity

$$\begin{aligned} \left\langle -\frac{\nabla^2 \varphi}{\varepsilon}, \psi \right\rangle_P &= - \int_P \frac{\nabla^2 \varphi(\mathbf{r})}{\varepsilon(\mathbf{r})} \varepsilon(\mathbf{r}) \psi^*(\mathbf{r}) d\mathbf{r} = - \int_P \nabla^2 \varphi(\mathbf{r}) \psi^*(\mathbf{r}) d\mathbf{r} \\ &= \left\langle \varphi, -\frac{\nabla^2 \psi}{\varepsilon} \right\rangle_P + \int_{\partial P} [\varphi(\mathbf{r}) \nabla \psi^*(\mathbf{r}) - \nabla \varphi(\mathbf{r}) \psi^*(\mathbf{r})] \cdot \mathbf{n} dS. \end{aligned}$$

Carrying on with our task, the latter term, which is somewhat in the way, can be eliminated because it equals zero:

$$\begin{aligned} & \int_{\partial P} [\varphi(\mathbf{r}) \nabla \psi^*(\mathbf{r}) - \nabla \varphi(\mathbf{r}) \psi^*(\mathbf{r})] \cdot \mathbf{n} dS \\ &= \sum_{\mathbf{R} \in L/\{-1,1\}} \int_{\partial P(\mathbf{R})} [\varphi(\mathbf{r}) \nabla \psi^*(\mathbf{r}) - \nabla \varphi(\mathbf{r}) \psi^*(\mathbf{r})] \cdot \mathbf{n} dS \\ & \quad + \sum_{\mathbf{R} \in L/\{-1,1\}} \int_{\partial P(\mathbf{R})} [\varphi(\mathbf{r} + \mathbf{R}) \nabla \psi^*(\mathbf{r} + \mathbf{R}) - \nabla \varphi(\mathbf{r} + \mathbf{R}) \psi^*(\mathbf{r} + \mathbf{R})] \cdot (-\mathbf{n}) dS \\ &= \sum_{\mathbf{R} \in L/\{-1,1\}} \int_{\partial P(\mathbf{R})} [\varphi(\mathbf{r}) \nabla \psi^*(\mathbf{r}) - \nabla \varphi(\mathbf{r}) \psi^*(\mathbf{r})] \cdot \mathbf{n} dS \\ & \quad - \sum_{\mathbf{R} \in L/\{-1,1\}} \int_{\partial P(\mathbf{R})} [e^{i\mathbf{k} \cdot \mathbf{R}} \varphi(\mathbf{r}) e^{-i\mathbf{k} \cdot \mathbf{R}} \nabla \psi^*(\mathbf{r}) - e^{i\mathbf{k} \cdot \mathbf{R}} \nabla \varphi(\mathbf{r}) e^{-i\mathbf{k} \cdot \mathbf{R}} \psi^*(\mathbf{r})] \cdot \mathbf{n} dS \\ &= 0. \end{aligned}$$

We assert that the domains of $H(\mathbf{k})$ and its adjoint coincide analogously to the case of the Laplacian on L^2 , making $H(\mathbf{k})$ self-adjoint.

To show the second part of the claim, consider the operator $H^{(0)}(\mathbf{k}) := -\nabla^2$ on $L^2(P)$ with the above boundary conditions. We can write down a complete (in $D(H^{(0)}(\mathbf{k}))$) set of eigenfunctions of $H^{(0)}(\mathbf{k})$ explicitly, namely:

$$\psi_{n,\mathbf{k}}^{(0)}(\mathbf{r}) := e^{i(\mathbf{k} + 2\pi \mathbf{n}(n)) \cdot \hat{\mathbf{r}}} \in L^2(P),$$

where $\mathbf{k} \in B$, $\mathbf{n} : \mathbb{N} \rightarrow \mathbb{Z}^d$ is an enumeration of all integer tuples and $\hat{\mathbf{r}}$ is the coordinate vector of \mathbf{r} in the basis $\mathcal{B}(L)$. Without loss of generality $\mathbf{r} = \hat{\mathbf{r}}$. Then the eigenvalues are $\lambda_{n,\mathbf{k}} := |\mathbf{k} + 2\pi \mathbf{n}(n)|^2$ and thus $\lambda_{n,\mathbf{k}} \rightarrow \infty$ as $n \rightarrow \infty$. Since $H^{(0)}(\mathbf{k})$ is elliptic and self-adjoint, we can apply Theorem A.3 b) to see that $H^{(0)}(\mathbf{k})$ has a compact resolvent. When comparing this argument with the discussion in [RS78], beware that their $\boldsymbol{\theta}$ is our \mathbf{k} and their \mathbf{k} is our $\mathbf{n}(n)$.

Now let

$$M(A, b) := \{\psi \in D(A) : \|\psi\| \leq 1, \|A\psi\| \leq b\}$$

and consider $M(H(\mathbf{k}), b) \subset L^2_\varepsilon(P)$ and $M(H^{(0)}(\mathbf{k}), b) \subset L^2(P)$. From our argument above and through Theorem A.3 c) we know that $M(H^{(0)}(\mathbf{k}), b)$ is compact in $L^2(P)$ for any b . Since

$$\left\langle -\frac{\nabla^2}{\varepsilon}\psi, \psi \right\rangle_P = \langle -\nabla^2\psi, \psi \rangle_{L^2(P)},$$

and $D(-\nabla^2/\varepsilon) = D(-\nabla^2)$, we are looking at the marginally different sets

$$\begin{aligned} A(b) &:= M(-\nabla^2/\varepsilon, b) = \{\psi \in D(-\nabla^2) : \|\psi\|_{L^2_\varepsilon(P)} \leq 1, \|\nabla^2\psi\|_{L^2(P)} \leq b\} \\ B(b) &:= M(-\nabla^2, b) = \{\psi \in D(-\nabla^2) : \|\psi\|_{L^2(P)} \leq 1, \|\nabla^2\psi\|_{L^2(P)} \leq b\} \end{aligned}$$

$B(b)$ is known to be compact for any b . Fix any b . The case $A(b) = \emptyset$ is trivial. Assume $A(b) \neq \emptyset$. We want to show that $A(b)$ is sequentially compact. So, let $\{x_n\}_{n \in \mathbb{N}} \subset A(b)$. Without loss of generality $\varepsilon_+ \geq 1$. Then we have $\{1/\varepsilon_+ x_n\}_{n \in \mathbb{N}} \subset B(1/\varepsilon_+ b)$, and we can pick a convergent subsequence $1/\varepsilon_+ x_{n_m} \rightarrow 1/\varepsilon_+ x$ in $B(b)$. Now $x_{n_m} \rightarrow x$ also holds in L^2_ε because of norm equivalence. Finally, by the definitions of the sets, we have $x \in A(b)$, so $A(b)$ is sequentially compact as well.

To be able to use Theorem A.3 c) to prove $H(\mathbf{k})$ to have compact resolvent, we note that $H(\mathbf{k})$ inherits its ellipticity, a.k.a. “bounded-from-below-ness” from $H^{(0)}(\mathbf{k})$ in a straightforward manner since ε is also bounded from below. \square

Using the above Lemma, we obtain the following important and informative result:

Theorem 2.5 *Each $H(\mathbf{k})$ has a complete set of eigenfunctions $\psi_{n,\mathbf{k}}(\mathbf{r})$ with eigenvalues $E_{n,\mathbf{k}}$. Extend $\psi_{n,\mathbf{k}}(\mathbf{r})$ to all of \mathbb{R}^d by using the boundary condition (2.3). For $\varphi \in \mathcal{S}(\mathbb{R}^d)$, let*

$$\tilde{\varphi}_m(\mathbf{k}) := \langle \varphi, \psi_{m,\mathbf{k}} \rangle_{\mathbb{R}^d}.$$

Then

a) we have a Parseval-like identity:

$$\int_{\mathbb{R}^d} |\varphi(\mathbf{r})|^2 d\mathbf{r} = \sum_n \frac{1}{\lambda(B)} \int_B |\tilde{\varphi}_n(\mathbf{k})|^2 d\mathbf{k},$$

b) the $\tilde{\varphi}_m(\mathbf{k})$ are the coefficients of an expansion of φ in Bloch functions:

$$\varphi(\mathbf{r}) = \sum_n \frac{1}{\lambda(B)} \int_B \tilde{\varphi}_n(\mathbf{k}) \psi_{n,\mathbf{k}}(\mathbf{r}) d\mathbf{k},$$

c) it is possible to extend the domain of “ \cdot ” to $L^2_\varepsilon(\mathbb{R}^d)$ continuously, and $-\nabla^2/\varepsilon$ obeys

$$-\left(\frac{\widetilde{\nabla^2}}{\varepsilon}\varphi\right)_n(\mathbf{k}) = E_{n,\mathbf{k}} \tilde{\varphi}_n(\mathbf{k})$$

for all $\varphi \in D(H(\mathbf{k}))$,

d) and “ \cdot ” maps $L^2_\varepsilon(\mathbb{R}^d)$ onto $\bigoplus_n L^2(B)$.

This is analogous to the first part of the proof of Theorem XIII.98 in [RS78], substituting Lemma 2.4 for the first part of its proof.

Altogether, we found out that each $H(\mathbf{k})$ has a purely discrete spectrum and its eigenvectors form a complete orthonormal basis $\{\psi_{n,\mathbf{k}}\}$ of $L^2_\varepsilon(P)$ with the appropriate boundary conditions. The functions $\psi_{n,\mathbf{k}}$ are called *Bloch functions* or *Bloch modes*, and apparently they even form something like an orthonormal basis in all of $L^2_\varepsilon(\mathbb{R}^d)$, as any function from that space can be expanded into Bloch modes without loss of information. Compared to our initial situation, we know quite a bit more about our problem, even enough to begin dealing with it computationally. In the following sections, we will attempt to cover some finer points of the theory before we move on.

Let me close this section with some bibliographic remarks. Kuchment's various works contain large amounts of information on the subject of this section. His book [Kuc93] is a technically thorough treatment of PDEs with periodic coefficients, while the book chapter [Kuc01] is an effective overview of the mathematics surrounding photonic crystals. However, for this particular purpose, [RS78] proves to be the reference with the most useful level of detail. Many introductory physics books (including [AM76]) present justifications of why the whole-space problem may be reduced to one primitive unit cell with Floquet boundary conditions, often under the name of "Bloch's Theorem". Few of these justifications can be considered proofs in the mathematical sense of the word.

2.4 Some consequences

In this section, we will develop some extensions to the theory of the previous section that shall help us in later chapters. Let us begin by stating a few Dirac-type relationships for the Floquet transform. For example, we will freely use the fact that, symbolically,

$$\frac{1}{\lambda(B)} \sum_{\mathbf{R} \in L} e^{i\mathbf{k} \cdot \mathbf{R}} = \delta(\mathbf{k}). \quad (2.4)$$

We will not prove this with analytical precision, nor is a reference for this relationship known. It is however pretty easy to make it *seem* plausible. From Theorem 2.2 we will use the fact that \mathcal{U} is an isometry. Let $\text{supp}(\varphi) \subset P$ and consider

$$\begin{aligned} \langle 1, \varphi \rangle_{\mathbb{R}^d} &= \int_P 1 \varepsilon(\mathbf{r}) \varphi^*(\mathbf{r}) d\mathbf{r}, \\ \langle \mathcal{U}1, \mathcal{U}\varphi \rangle_{L^2(B \times L^2_\varepsilon(P))} &= \frac{1}{\lambda(B)} \int_B \int_P (\mathcal{U}1)_{\mathbf{k}}(\mathbf{r}) \varepsilon(\mathbf{r}) (\mathcal{U}\varphi)_{\mathbf{k}}^*(\mathbf{r}) d\mathbf{r} d\mathbf{k} \\ &= \frac{1}{\lambda(B)} \int_B \int_P \sum_{\mathbf{R} \in L} e^{i\mathbf{k} \cdot \mathbf{R}} \varepsilon(\mathbf{r}) \sum_{\mathbf{R}' \in L} e^{-i\mathbf{k} \cdot \mathbf{R}'} \varphi^*(\mathbf{r} - \mathbf{R}') d\mathbf{r} d\mathbf{k} \\ &= \frac{1}{\lambda(B)} \int_B \sum_{\mathbf{R} \in L} e^{i\mathbf{k} \cdot \mathbf{R}} d\mathbf{k} \int_P \varepsilon(\mathbf{r}) \varphi^*(\mathbf{r}) d\mathbf{r} \\ \Rightarrow \frac{1}{\lambda(B)} \int_B \sum_{\mathbf{R} \in L} e^{i\mathbf{k} \cdot \mathbf{R}} d\mathbf{k} &= 1. \end{aligned}$$

(This argument is flawed because $1 \notin L^2_\varepsilon(\mathbb{R}^d)$ and we did not justify the switching of the sum and the integral.) By formally using the geometric series expansion, it is easy to show that

$$\sum_{\mathbf{R} \in L} e^{i\mathbf{k} \cdot \mathbf{R}} = 0 \quad \text{for } \mathbf{k} \neq 0$$

if we kindly ignore that the series does not converge at all since $|e^{i\mathbf{k} \cdot \mathbf{R}}| = 1$. As a result, Equation (2.4) takes the form of a postulate rather than a proven fact. As a close analog, consider

$$\frac{1}{\lambda(B)} \int_B e^{i\mathbf{k} \cdot \mathbf{R}} d\mathbf{k} = \delta_{\mathbf{R}, \mathbf{0}}, \quad (2.5)$$

which can be proven exactly. The following theorems are consequences of Equation (2.4):

Theorem 2.6 *The Bloch functions $\psi_{n, \mathbf{k}}$, extended to all of \mathbb{R}^d by means of the boundary conditions (2.3), are \mathbf{k} - and n -orthogonal, i.e.*

$$\langle \psi_{n, \mathbf{k}}, \psi_{m, \mathbf{k}'} \rangle_{\mathbb{R}^d} = \lambda(B) \delta(\mathbf{k} - \mathbf{k}') \delta_{n, m}.$$

Proof Consider

$$\begin{aligned}
\langle \psi_{n,\mathbf{k}}, \psi_{m,\mathbf{k}'} \rangle_{\mathbb{R}^d} &= \int_{\mathbb{R}^d} \psi_{n,\mathbf{k}}(\mathbf{r}) \psi_{m,\mathbf{k}'}^*(\mathbf{r}) d\mathbf{r} \\
&= \sum_{\mathbf{R}} \int_P \psi_{n,\mathbf{k}}(\mathbf{r} - \mathbf{R}) \psi_{m,\mathbf{k}'}^*(\mathbf{r} - \mathbf{R}) d\mathbf{r} \\
&= \sum_{\mathbf{R}} \int_P e^{-i\mathbf{k} \cdot \mathbf{R}} \psi_{n,\mathbf{k}}(\mathbf{r}) e^{i\mathbf{k}' \cdot \mathbf{R}} \psi_{m,\mathbf{k}'}^*(\mathbf{r}) d\mathbf{r} \\
&= \sum_{\mathbf{R}} \underbrace{e^{i(\mathbf{k}' - \mathbf{k}) \cdot \mathbf{R}}}_{=\delta(\mathbf{k} - \mathbf{k}')\lambda(B)} \langle \psi_{n,\mathbf{k}}, \psi_{m,\mathbf{k}'} \rangle_P \\
&= \lambda(B) \delta(\mathbf{k} - \mathbf{k}') \delta_{n,m},
\end{aligned}$$

which establishes the claim. (Lebesgue's dominated convergence theorem justifies the splitting step.) \square

Another, more important theorem gives us an inversion formula for the Floquet transform:

Theorem 2.7 For $f \in L^2_{\varepsilon}(\mathbb{R}^d)$ and its Floquet transform $\mathcal{U}f \in L^2(B \times L^2_{\varepsilon}(P))$, we have

$$f(\mathbf{r}) = \frac{1}{\lambda(B)} \int_B (\mathcal{U}f)_{\mathbf{k}}(\mathbf{r}) d\mathbf{k}.$$

Equivalently, for any $f \in L^2(B \times L^2_{\varepsilon}(P))$,

$$(\mathcal{U}^{-1}f)(\mathbf{r}) = \frac{1}{\lambda(B)} \int_B f_{\mathbf{k}}(\mathbf{r}) d\mathbf{k}.$$

Proof Consider

$$\begin{aligned}
\frac{1}{\lambda(B)} \int_B (\mathcal{U}f)_{\mathbf{k}}(\mathbf{r}) d\mathbf{r} &= \frac{1}{\lambda(B)} \int_B \sum_{\mathbf{R} \in L} e^{i\mathbf{k} \cdot \mathbf{R}} f(\mathbf{r} - \mathbf{R}) d\mathbf{k} \\
&= \frac{1}{\lambda(B)} \sum_{\mathbf{R} \in L} f(\mathbf{r} - \mathbf{R}) \int_B e^{i\mathbf{k} \cdot \mathbf{R}} d\mathbf{k} \\
&= \sum_{\mathbf{R} \in L} f(\mathbf{r} - \mathbf{R}) \delta_{\mathbf{R},0} = f(\mathbf{r}),
\end{aligned}$$

as claimed. In the current setting, we will forgo justifying the exchange of the sum and the integral and leave this for a later writeup of a more theoretical scope, where all the distribution-theoretic subtleties are worked out in full detail. In fact, situations like this one will arise a few times throughout the rest of this thesis, and we will refer to the explanation given here in each of these cases. \square

The Floquet transform decomposes $-\nabla^2/\varepsilon$ into a direct integral of *identical* differential operators on *varying* domains (the domains vary because different Floquet boundary conditions are valid on each). It is also possible and quite easy to achieve the reverse—a decomposition into *varying* differential operators on *identical* domains. Consider the operator \mathcal{P} , defined as

$$\begin{aligned}
\mathcal{P} : L^2(B \times L^2_{\varepsilon}(P)) &\rightarrow L^2(B \times L^2_{\varepsilon}(P)) \\
\psi &\mapsto \mathcal{P}\psi, \quad (\mathcal{P}\psi)_{\mathbf{k}}(\mathbf{r}) := e^{-i\mathbf{k} \cdot \mathbf{r}} \psi_{\mathbf{k}}(\mathbf{r}).
\end{aligned}$$

\mathcal{P} is an isometry:

$$\begin{aligned}
\langle \mathcal{P}\varphi, \mathcal{P}\psi \rangle_{L^2(B \times L^2_{\varepsilon}(P))} &= \frac{1}{\lambda(B)} \int_B \langle \mathcal{P}\varphi, \mathcal{P}\psi \rangle_P d\mathbf{k} \\
&= \frac{1}{\lambda(B)} \int_B \int_P e^{-i\mathbf{k} \cdot \mathbf{r}} \varphi(\mathbf{r}) \varepsilon(\mathbf{r}) e^{i\mathbf{k} \cdot \mathbf{r}} \psi^*(\mathbf{r}) d\mathbf{r} \\
&= \langle \varphi, \psi \rangle_{L^2(B \times L^2_{\varepsilon}(P))}.
\end{aligned}$$

\mathcal{P} transforms the Floquet boundary conditions into periodic ones:

$$(\mathcal{P}\psi)_{\mathbf{k}}(\mathbf{r} + \mathbf{R}) = e^{-i\mathbf{k}\cdot(\mathbf{r}+\mathbf{R})}\psi_{\mathbf{k}}(\mathbf{r} + \mathbf{R}) = e^{-i\mathbf{k}\cdot(\mathbf{r}+\mathbf{R})}e^{i\mathbf{k}\cdot\mathbf{R}}\psi_{\mathbf{k}}(\mathbf{r}) = e^{-i\mathbf{k}\cdot\mathbf{r}}\psi_{\mathbf{k}}(\mathbf{r}) = (\mathcal{P}\psi)_{\mathbf{k}}(\mathbf{r})$$

for any $\mathbf{R} \in L$ and any $\mathbf{r} \in \partial P(\mathbf{R})$. Similarly, the condition on the gradient becomes

$$\begin{aligned} \nabla(\mathcal{P}\psi)_{\mathbf{k}}(\mathbf{r} + \mathbf{R}) &= \nabla(e^{-i\mathbf{k}\cdot(\mathbf{r}+\mathbf{R})}\psi_{\mathbf{k}}(\mathbf{r} + \mathbf{R})) \\ &= \psi_{\mathbf{k}}(\mathbf{r} + \mathbf{R})\nabla e^{-i\mathbf{k}\cdot(\mathbf{r}+\mathbf{R})} + e^{-i\mathbf{k}\cdot(\mathbf{r}+\mathbf{R})}\nabla\psi_{\mathbf{k}}(\mathbf{r} + \mathbf{R}) \\ &= e^{i\mathbf{k}\cdot\mathbf{R}}\psi_{\mathbf{k}}(\mathbf{r})(-i\mathbf{k}e^{-i\mathbf{k}\cdot(\mathbf{r}+\mathbf{R})}) + e^{-i\mathbf{k}\cdot(\mathbf{r}+\mathbf{R})}e^{i\mathbf{k}\cdot\mathbf{R}}\nabla\psi_{\mathbf{k}}(\mathbf{r}) \\ &= -i\mathbf{k}e^{-i\mathbf{k}\cdot\mathbf{r}}\psi_{\mathbf{k}}(\mathbf{r}) + e^{-i\mathbf{k}\cdot\mathbf{r}}\nabla\psi_{\mathbf{k}}(\mathbf{r}) \\ &= \nabla(\mathcal{P}\psi)_{\mathbf{k}}(\mathbf{r}), \end{aligned}$$

where we left out the inner product with the boundary unit normal for simplicity. The differential operator $H(\mathbf{k})$ becomes

$$\begin{aligned} H_p(\mathbf{k}) := \mathcal{P}H(\mathbf{k})\mathcal{P}^{-1}u &= e^{-i\mathbf{k}\cdot\mathbf{r}}\left(-\frac{\nabla^2}{\varepsilon}\right)e^{i\mathbf{k}\cdot\mathbf{r}}u \\ &= -e^{-i\mathbf{k}\cdot\mathbf{r}}\frac{1}{\varepsilon}\left[e^{i\mathbf{k}\cdot\mathbf{r}}\nabla^2u + 2\nabla e^{i\mathbf{k}\cdot\mathbf{r}} \cdot \nabla u + u\nabla^2 e^{i\mathbf{k}\cdot\mathbf{r}}\right] \\ &= -\frac{1}{\varepsilon}\left[\nabla^2u + 2i\mathbf{k} \cdot \nabla u - k^2u\right] \\ &= -\frac{1}{\varepsilon}\left[\nabla^2 + 2i\mathbf{k} \cdot \nabla - k^2\right]u. \end{aligned}$$

Observe how the differential operator now explicitly depends on \mathbf{k} , while the boundary conditions are independent of \mathbf{k} . The letter \mathcal{P} was chosen for this isometry since it turns the Bloch modes $\psi_{n,\mathbf{k}}$ into *periodic* functions $u_{n,\mathbf{k}}$. For the remainder of this thesis, we will maintain the letters ψ and u respectively for these two kinds of functions, so that

$$u_{n,\mathbf{k}}(\mathbf{r}) := e^{-i\mathbf{k}\cdot\mathbf{r}}\psi_{n,\mathbf{k}}(\mathbf{r}).$$

The $u_{n,\mathbf{k}}$ form orthonormal bases of $L^2_\varepsilon(P)$ with periodic boundary conditions; this property carries over from the $\psi_{n,\mathbf{k}}$ by means of the isometry \mathcal{P} .

2.5 Bands and gaps

In this section, we will take a closer look at the joint spectrum of the $H(\mathbf{k})$. Each of the $H(\mathbf{k})$ has been shown to have a fully discrete spectrum. In Theorem 2.5, we defined a quantity $E_{n,\mathbf{k}}$ to specify $H(\mathbf{k})$'s n th-largest eigenvalue by magnitude. Naturally, we're curious how $E_{n,\mathbf{k}}$ behaves if viewed as a function of \mathbf{k} (and n is kept constant). To make this point of view explicit, we will write $E_n(\mathbf{k})$ instead of $E_{n,\mathbf{k}}$ for the rest of this chapter.

First, let us examine the physical meaning and properties of $E_n(\mathbf{k}) = \omega_n^2(\mathbf{k})/c^2$. The letter E was chosen for a reason— $E_n(\mathbf{k})$ is the n th energy level at which an electromagnetic wave can propagate through our crystal while obeying the Floquet conditions (2.3) for \mathbf{k} . Since there is a merely discrete spectrum for each \mathbf{k} , only light of certain energy levels is allowed to propagate through the crystal. If there is no energy level corresponding to an incident wave with parameters \mathbf{k} and ω , it will not propagate in an unmodified manner.

The graph $\{(\mathbf{k}, E_n(\mathbf{k})) : \mathbf{k} \in B, n = 1, 2, \dots\}$ of $E_n(\mathbf{k})$ is called the *dispersion relation*. Since $E_m(\mathbf{k})$ is an observable physical quantity, we strongly expect it to change continuously, if not differentially, in \mathbf{k} . (but cf. the next section) So, we can visualize the dispersion relation as a stack of sheets over the Brillouin zone. These sheets are called *bands*. It turns out that the bands behave in most particular ways even in simple cases, as we will see in Chapter 6. I encourage you to take a look at, for example, Figures 6.5, 6.6 and 6.10 right now to see some of the phenomena which arise. Sometimes, the bands touch or penetrate each other. Wherever $E_n(\mathbf{k}) = E_m(\mathbf{k})$ for a given \mathbf{k} and $n \neq m$, the two bands (and, more specifically, the energy levels) are called *degenerate*. Often, the $E_m(\mathbf{k})$ lose their otherwise excellent smoothness properties in or around these degeneracies. If two bands cross each other or share certain energy levels, they are called *entangled*.

Sometimes we can even have a situation where there is an interval where *no* eigenvalue $E_n(\mathbf{k})$ is found, for any \mathbf{k} . An interval like this is called a *band gap*. It is exactly the emergence of such gaps that makes photonic crystals so important. The gap allows the crystal to act as an insulator of light, just like regular air is an insulator for electric current. Many researchers believe that the presence of a proper insulator of light is a stepping stone on the way to fully-photonic integrated circuits. This is why photonic crystals are often called *photonic band gap (PBG) materials*.

2.6 Smoothness of the dispersion relation

In this section, we will argue that non-degenerate eigenvalues actually depend on \mathbf{k} in a continuously differentiable manner. Suppose $E_n(\mathbf{k}_0)$ is a non-degenerate eigenvalue of $H(\mathbf{k}_0)$.

Define a function

$$\begin{aligned} F : L_\varepsilon^2(P) \times \mathbb{R}_0^+ \times B &\rightarrow L_\varepsilon^2(P) \times \mathbb{R}_0^+ \\ (u, \lambda, \mathbf{k}) &\mapsto \begin{pmatrix} H_p(\mathbf{k})u - \lambda u \\ \|u\|^2 - 1 \end{pmatrix} \end{aligned}$$

where we chose $H_p(\mathbf{k})$, the explicitly varying differential operator, over $H(\mathbf{k})$ in order to work on a constant domain. Whenever $F(u, \lambda, \mathbf{k}) = \mathbf{0}$, (u, λ) is an eigenpair of $H_p(\mathbf{k})$. We are going to sketch the use of the implicit function theorem A.4 on F in order to find a locally continuously differentiable implicit function $\mathbf{k} \mapsto (u, \lambda)$. Consequently, in our application, D_x differentiates by (u, λ) , and D_y differentiates by \mathbf{k} , where D_x and D_y are as defined in Theorem A.4. Now, formally

$$(D_x F(u, \lambda, \mathbf{k}))(v, \mu) = \begin{pmatrix} H_p(\mathbf{k})v - \lambda v - \mu u \\ 2 \langle v, u \rangle_P \end{pmatrix}.$$

To apply Theorem A.4, we need $D := D_x F(u_{n, \mathbf{k}_0}, E_n(\mathbf{k}_0), \mathbf{k}_0)$ to be an isomorphism. D clearly is linear. We will show that D is bijective, and we assert that spaces can be found such that D and D^{-1} are continuous; Sobolev spaces are suitable in this regard. To show that D is injective, we have to prove that it only has the trivial null space, by considering the equations

$$\begin{aligned} H_p(\mathbf{k}_0)v - E_n(\mathbf{k}_0)v &= \mu u_{n, \mathbf{k}_0}, \\ \langle v, u_{n, \mathbf{k}_0} \rangle_P &= 0. \end{aligned}$$

The eigenspace $\mathcal{E}_{n, \mathbf{k}_0}$ of $H_p(\mathbf{k}_0)$ only consists of the span of u_{n, \mathbf{k}_0} , since we specifically excluded degeneracies. Also, $v \in \mathcal{E}_{n, \mathbf{k}_0}^\perp$, by the second equation. $H_p(\mathbf{k}_0)$ takes on “diagonal” form w.r.t. its eigenvectors, so $\mathcal{E}_{n, \mathbf{k}_0}$ is an orthogonally invariant subspace of $H_p(\mathbf{k}_0)$, and also of $H_p(\mathbf{k}_0) - E_n(\mathbf{k}_0) \text{Id}$, since any subspace is an orthogonally invariant subspace of the identity. What this means is that if v is orthogonal to u_{n, \mathbf{k}_0} , so is $H_p(\mathbf{k}_0)v - E_n(\mathbf{k}_0)v$. Hence $\mu = 0$ and $H_p(\mathbf{k}_0)v = E_n(\mathbf{k}_0)v$, and so $v \in \mathcal{E}_{n, \mathbf{k}_0} \cap \mathcal{E}_{n, \mathbf{k}_0}^\perp = \{0\}$. Thus, D is injective.

To show that it is surjective, consider $H_p(\mathbf{k}_0)|_{\mathcal{E}_{n, \mathbf{k}_0}^\perp} \cdot E_n(\mathbf{k}_0) \in \rho(H_p(\mathbf{k}_0)|_{\mathcal{E}_{n, \mathbf{k}_0}^\perp})$, so $M := [H_p(\mathbf{k}_0) - E_n(\mathbf{k}_0) \text{Id}]|_{\mathcal{E}_{n, \mathbf{k}_0}^\perp}$ is invertible. Let $y \in L_\varepsilon^2(P)$ and $\alpha \in \mathbb{R}$. We can split

$$y = \langle y, u_{n, \mathbf{k}_0} \rangle_P u_{n, \mathbf{k}_0} + y_\perp.$$

Then, let $v := M^{-1}y_\perp + \alpha/2u_{n, \mathbf{k}_0}$ and $\mu = -\langle y, u_{n, \mathbf{k}_0} \rangle_P$.

$$\begin{aligned} &D_x F(u_{n, \mathbf{k}_0}, E_n(\mathbf{k}_0), \mathbf{k}_0)(v, \mu) \\ &= \begin{pmatrix} H_p(\mathbf{k}_0)v - E_n(\mathbf{k}_0)v - \mu u_{n, \mathbf{k}_0} \\ 2 \langle v, u_{n, \mathbf{k}_0} \rangle_P \end{pmatrix} \\ &= \begin{pmatrix} (H_p(\mathbf{k}_0) - E_n(\mathbf{k}_0) \text{Id})(M^{-1}y_\perp + \alpha/2u_{n, \mathbf{k}_0}) + \langle y, u_{n, \mathbf{k}_0} \rangle_P u_{n, \mathbf{k}_0} \\ 2 \langle M^{-1}y_\perp + \alpha/2u_{n, \mathbf{k}_0}, u_{n, \mathbf{k}_0} \rangle_P \end{pmatrix} \\ &= \begin{pmatrix} (H_p(\mathbf{k}_0) - E_n(\mathbf{k}_0) \text{Id})(M^{-1}y_\perp) + (H_p(\mathbf{k}_0) - E_n(\mathbf{k}_0) \text{Id})\alpha/2u_{n, \mathbf{k}_0} + \langle y, u_{n, \mathbf{k}_0} \rangle_P u_{n, \mathbf{k}_0} \\ \alpha \end{pmatrix} \\ &= \begin{pmatrix} MM^{-1}y_\perp + \alpha/2(E_n(\mathbf{k}_0)u_{n, \mathbf{k}_0} - E_n(\mathbf{k}_0)u_{n, \mathbf{k}_0}) + \langle y, u_{n, \mathbf{k}_0} \rangle_P u_{n, \mathbf{k}_0} \\ \alpha \end{pmatrix} \\ &= \begin{pmatrix} y_\perp + \langle y, u_{n, \mathbf{k}_0} \rangle_P u_{n, \mathbf{k}_0} \\ \alpha \end{pmatrix} = \begin{pmatrix} y \\ \alpha \end{pmatrix}. \end{aligned}$$

Since y and α were picked arbitrarily, we have shown that D is also surjective. Theorem A.4 now grants us a continuously differentiable function on a neighborhood of \mathbf{k}_0 that yields an eigenpair of the same band. So, we can see that speaking of *bands* as functions as suggested in the previous section is not entirely unwarranted.

The dispersion relation can also be viewed as the set of zeros of an entire function of order d in \mathbb{C}^{d+1} by Theorem 4.4.2 of [Kuc93].

Chapter 3

Wannier functions

3.1 Definition

At this point, we have applied the Floquet transform to our whole-space problem, and we found a number of eigenfunctions $\psi_{n,\mathbf{k}} \in L^2_\varepsilon(P)$ which depend on an extra parameter \mathbf{k} , since they live in the image space of the transform. It is natural to ask what we get when we try to use the inverse Floquet transform on these functions to convert them “back” into functions on the whole space and make them independent of \mathbf{k} . The resulting functions are called *Wannier functions* and are the subject of this chapter.

Definition 3.1 *The n th Wannier function $w_{n,\mathbf{0}}$ centered at $\mathbf{0}$ is defined as*

$$w_{n,\mathbf{0}}(\mathbf{r}) := \mathcal{U}^{-1}(\psi_n) \in L^2_\varepsilon(\mathbb{R}^d).$$

More generally, the n th Wannier function $w_{n,\mathbf{R}}$ centered at \mathbf{R} is defined as

$$w_{n,\mathbf{R}}(\mathbf{r}) := w_{n,\mathbf{0}}(\mathbf{r} - \mathbf{R}).$$

Using Theorem 2.7, the function $w_{n,\mathbf{R}}$ can also be defined directly:

$$w_{n,\mathbf{R}}(\mathbf{r}) = \frac{1}{\lambda(B)} \int_B \psi_{n,\mathbf{k}}(\mathbf{r} - \mathbf{R}) d\mathbf{k} = \frac{1}{\lambda(B)} \int_B e^{-i\mathbf{k} \cdot \mathbf{R}} \psi_{n,\mathbf{k}}(\mathbf{r}) d\mathbf{k}.$$

By their direct definition, Wannier functions can be viewed as an “average” electromagnetic field at a certain position. They find their most common use as a localized basis set for the expansion of wave functions in periodic systems. We will investigate the use of Wannier functions as basis sets in Section 3.3. In many ways, the “Wannier” (or “real-space”) view is complementary to the “ \mathbf{k} -space” (or “reciprocal-space”) view developed in the previous chapter.

Wannier functions were first introduced by Gregory H. Wannier in 1937. In his work on the large exciton [Wan37], he found himself in need of localized states to describe the systems he was considering. The direct definition of Wannier functions also arises almost naturally in conjunction with the LCAO approximation of entire-crystal wave functions in solid state theory, where these wave functions are approximated as *Linear Combinations of Atomic Orbitals*. (see [AM76], Chapter 10)

By their definition, Wannier functions can be expanded in Bloch functions according to Theorem 2.5 as

$$\widetilde{(w_{n,\mathbf{R}})_\nu} = \delta_{n,\nu} e^{-i\mathbf{k} \cdot \mathbf{R}}. \quad (3.1)$$

Blount [[Blo62], Section 5] states that Wannier functions can also be defined by variational means. They are the functions ψ to minimize the integral

$$I := \langle H(\mathbf{k})\psi, \psi \rangle_{\mathbb{R}^d}$$

under the conditions $\|\psi\| = 1$ and $\langle T_{\mathbf{R}}\psi, \psi \rangle_{\mathbb{R}^d} = 0$. $T_{\mathbf{R}}$ is a translation by a lattice vector $\mathbf{R} \in L$.

3.2 Localization of Wannier functions: Basics

Unfortunately, Wannier functions have a serious drawback if computed naïvely by means of Definition 3.1—their behavior is quite erratic, and they are not well-localized. This stems from the fact that every Bloch function has its own indeterminate phase factor. The functions $e^{i\varphi_n(\mathbf{k})}\psi_{n,\mathbf{k}}$ for any $\varphi_n(\mathbf{k})$ are equivalent to just $\psi_{n,\mathbf{k}}$ from the point of view of Section 2.3. This non-uniqueness can be circumvented to some extent by choosing the phases of $\psi_{n,\mathbf{k}}$ such that for a given point \mathbf{r}

$$\arg \psi_{n,\mathbf{k}}(\mathbf{r}) = \arg \psi_{n,\mathbf{k}'}(\mathbf{r}) \quad \text{for } \mathbf{k}, \mathbf{k}' \in B. \quad (3.2)$$

Equation (3.2) works fine as a localizer for Wannier functions as long as the n th band is energetically separate from all other bands. But it fails for two or more energetically entangled bands even if appropriately generalized. Our next best move to achieve localization is to generalize the notion of a Wannier function. We lift the constraint that the n th Wannier function be computed exclusively from n th band Bloch modes, and postulate that is ambiguous in the presence of degeneracies anyway. Assume, for simplicity, that the bands $1, \dots, J$ are entangled. We introduce mixing matrices $U^{\mathbf{k}} \in \mathbb{C}^{J \times J}$ for each $\mathbf{k} \in B$ and new “mixed” or “generalized” Bloch modes by

$$\psi_{n,\mathbf{k},\text{gen}} := \sum_{m=1}^J U_{n,m}^{\mathbf{k}} \psi_{m,\mathbf{k}}.$$

The Wannier functions computed by means of Definition 3.1 from the $\psi_{n,\mathbf{k},\text{gen}}$ are called *generalized Wannier functions*. We require that the $U^{\mathbf{k}}$ are unitary to keep the $\psi_{n,\mathbf{k},\text{gen}}$ orthonormal. By adjusting the $U^{\mathbf{k}}$, it is possible to fix the phases of the Bloch functions sufficiently to make the Wannier functions localized even in the presence of entanglement. Much of the rest of this thesis is dedicated to developing a minimization procedure to find the right set of $U^{\mathbf{k}}$. Marzari and Vanderbilt, authors of the seminal paper [MV97] from 1997, whose approach we are following here, were the first to compute maximally localized Wannier functions in this generalized sense. The idea of localizing Wannier functions by means of choosing Bloch mode phases is older, however. It dates back to at least Blount’s paper [Blo62] from 1962, and was probably mentioned before then.

Marzari and Vanderbilt also make the conjecture that maximally localized Wannier functions should have no more than a constant overall phase and should be purely real otherwise.

While we have generously used the term before now, what exactly do we mean by “localized”? We choose it to mean “having minimal second moment”. The second moment of the n th Wannier function would be

$$\Omega_n := \langle r^2 w_{n,\mathbf{0}}, w_{n,\mathbf{0}} \rangle_{\mathbb{R}^d} - |\langle \mathbf{r} w_{n,\mathbf{0}}, w_{n,\mathbf{0}} \rangle_{\mathbb{R}^d}|^2. \quad (3.3)$$

Since our situation involves not just one band, but a set of J bands, our aim is to have

$$\Omega := \sum_{n=1}^J \Omega_n \rightarrow \min.$$

The vector $\langle \mathbf{r} w_{n,\mathbf{0}}, w_{n,\mathbf{0}} \rangle_{\mathbb{R}^d}$ has a meaning beyond its role in the second moment: it can be understood as the n th *Wannier center*.

3.3 Wannier functions as a basis set

The idea to use maximally-localized Wannier functions in the sense of Marzari and Vanderbilt as basis sets for photonic crystal computations was developed independently in [BMGM⁺03] and [WC03]. To successfully compute the inner products involved in a Ritz-Galerkin treatment of such computations, it is of critical importance that good localization is achieved, because otherwise the spatial integration cutoff will lead to severe imprecision. Except for this difficulty, Wannier functions have many properties that make them well-suited for computation, such as being orthonormal:

Theorem 3.2 *The Wannier functions are n - and \mathbf{R} -orthogonal, i.e.*

$$\langle w_{n,\mathbf{R}}, w_{m,\mathbf{R}'} \rangle_{\mathbb{R}^d} = \delta_{m,n} \delta_{\mathbf{R},\mathbf{R}'}$$

Proof Without loss of generality, assume $\mathbf{R}' = \mathbf{0}$.

$$\begin{aligned}
\langle w_{n,\mathbf{R}}, w_{m,\mathbf{0}} \rangle_{\mathbb{R}^d} &= \frac{1}{\lambda(B)^2} \left\langle \int_B e^{-i\mathbf{k}\cdot\mathbf{R}} \psi_{n,\mathbf{k}}(\mathbf{r}) d\mathbf{k}, \int_B \psi_{m,\mathbf{k}'}(\mathbf{r}) d\mathbf{k}' \right\rangle_{\mathbb{R}^d} \\
&= \frac{1}{\lambda(B)^2} \int_B \int_B e^{-i\mathbf{k}\cdot\mathbf{R}} \langle \psi_{n,\mathbf{k}}, \psi_{m,\mathbf{k}'} \rangle_{\mathbb{R}^d} d\mathbf{k} d\mathbf{k}' \\
&= \frac{1}{\lambda(B)^2} \int_B \int_B e^{-i\mathbf{k}\cdot\mathbf{R}} \lambda(B) \delta(\mathbf{k} - \mathbf{k}') \delta_{n,m} d\mathbf{k} d\mathbf{k}' \\
&= \delta_{m,n} \frac{1}{\lambda(B)} \int_B e^{-i\mathbf{k}\cdot\mathbf{R}} d\mathbf{k} = \delta_{m,n} \delta_{\mathbf{R},\mathbf{0}},
\end{aligned}$$

as claimed. \square

Theorem 3.3 *The Wannier functions form an orthonormal basis of $L^2_\varepsilon(\mathbb{R}^d)$.*

Proof Suppose we have an expansion

$$\varphi(\mathbf{r}) = \sum_n \frac{1}{\lambda(B)} \int_B \tilde{\varphi}_n(\mathbf{k}) \psi_{n,\mathbf{k}}(\mathbf{r}) d\mathbf{k}$$

of a function $\varphi \in L^2_\varepsilon(\mathbb{R}^d)$ in Bloch modes according to Theorem 2.5. Next, remember that $\tilde{\varphi}_n$ is only defined on B and could be thought of as \tilde{L} -periodic. Periodic functions, however, can be conveniently analyzed using a Fourier series expansion. So, let us consider $\tilde{\varphi}$'s lattice Fourier coefficients

$$\alpha_{n,\mathbf{R}} := \frac{1}{\lambda(B)} \int_{\mathbf{k}} e^{i\mathbf{k}\cdot\mathbf{R}} \tilde{\varphi}_n(\mathbf{k}) d\mathbf{k}.$$

Let us now use the $\alpha_{n,\mathbf{R}}$ as coefficients in an expansion into Wannier functions and see what happens.

$$\begin{aligned}
&\sum_n \sum_{\mathbf{R}} \alpha_{n,\mathbf{R}} w_{n,\mathbf{R}}(\mathbf{r}) \\
&= \frac{1}{\lambda(B)^2} \sum_n \sum_{\mathbf{R}} \int_B e^{i\mathbf{k}\cdot\mathbf{R}} \tilde{\varphi}_n(\mathbf{k}) d\mathbf{k} \int_B e^{-i\mathbf{k}'\cdot\mathbf{R}} \psi_{n,\mathbf{k}'}(\mathbf{r}) d\mathbf{k}' \\
&\stackrel{(*)}{=} \frac{1}{\lambda(B)^2} \sum_n \int_B \int_B \sum_{\mathbf{R}} e^{i(\mathbf{k}-\mathbf{k}')\cdot\mathbf{R}} \tilde{\varphi}_n(\mathbf{k}) \psi_{n,\mathbf{k}'}(\mathbf{r}) d\mathbf{k} d\mathbf{k}' \\
&= \frac{1}{\lambda(B)^2} \sum_n \int_B \int_B \underbrace{\sum_{\mathbf{R}} e^{i(\mathbf{k}-\mathbf{k}')\cdot\mathbf{R}}}_{=\delta(\mathbf{k}-\mathbf{k}')\lambda(B)} \tilde{\varphi}_n(\mathbf{k}) \psi_{n,\mathbf{k}'}(\mathbf{r}) d\mathbf{k} d\mathbf{k}' \\
&= \frac{1}{\lambda(B)} \sum_n \int_B \tilde{\varphi}_n(\mathbf{k}) \psi_{n,\mathbf{k}}(\mathbf{r}) d\mathbf{k} \\
&= \varphi(\mathbf{r})
\end{aligned}$$

As in the proof of Theorem 2.7, we will not justify the exchange of the sum and the integral in the step marked (*). We will also not worry about the convergence of the Fourier series. So, we have found a way to expand any function $\varphi \in L^2_\varepsilon(\mathbb{R}^d)$ into Wannier functions, as claimed. \square

3.4 Localization methods

At present, we are missing the following pieces in order to be able to minimize the spread functional Ω defined above:

- a computable, discretized term for Ω ,
- a gradient of this discretization,

- and a method to use this information to find a minimum in Ω .

Over the rest of this section and the next chapter, we will develop these tools.

It is quite natural to ask whether we could get by without the gradient information, and thus save a lot of work. The answer is: not really. Due to the high number of dimensions of the search space, a gradient-less minimization approach is probably bound to fail (or at least be too expensive to be feasible). Even for a measly 4×4 Brillouin zone mesh with as little as 5 bands per \mathbf{k} -point, the resulting search space has about $4^2 \times 5 = 80$ real (not complex) dimensions. (To see this, consider the skew-hermitian matrices as a local parametrization of the manifold of unitary matrices, as in Section 4.7.) For more realistic mesh sizes and band counts, any minimization algorithm that does not use gradient information will likely take too much computational effort, since the information otherwise contained in the gradient needs to be obtained by regular function evaluations, which is prohibitive in so many dimensions.

Initial attempts made in this work to use line-searches along pseudo-random directions proved futile. It seems that most directions provide little change at all, and given an already relatively good starting guess as the one identified in Section 4.8.1, even those directions which do allow change usually only permit an increase in Wannier function spread.

As a result, we will need to apply a gradient-bound nonlinear minimum search. Two algorithms were tested: Simple gradient descent with Brent's method as a line search and a slight modification of the Polak-Ribière variant of the nonlinear CG minimum search, again with Brent's method as the line search. As expected, the CG method provided significant benefits, often cutting the time required to reach a given threshold more than in half. At the end of the next chapter, we will have developed a variant of Marzari and Vanderbilt's method of minimizing Wannier function spread.

Gygi, Fattbert and Schwegler [GFS03] have developed an entirely different approach to minimizing Wannier function spread: They try to achieve localization not by mixing at the Bloch mode level, but instead at the Wannier function level. Effectively, their algorithm finds the mixture of all the $w_{n,\mathbf{R}}$ such that the resulting spread is minimal. In their case, the minimum search can be reduced to the problem of finding a unitary matrix U such that

$$\sum_n \sum_{\mu, \nu: \mu \neq \nu} [U^H A_n U]_{\mu, \nu}^2 \rightarrow \min$$

for a number of matrices A_n . This is a generalization of what the Jacobi matrix diagonalization method is supposed to do, namely the simultaneous diagonalization of all the matrices A_n . An appropriate algorithm already exists (see [BGBM93] and the extension in [CS96]) and makes for a somewhat simpler method than the one developed here. Unfortunately, I was unable to squeeze an implementation and a thorough analysis of this new method into the schedule of my thesis work.

3.5 The location operator in \mathbf{k} -space

In preparation of Chapter 4, we will now find computable expressions for the (otherwise almost incomputable) scalar products over all of \mathbb{R}^d arising in Equation (3.3). The discussion builds heavily upon material from [Blo62].

Lemma 3.4 *[[Blo62], Appendix A] Let $f \in L_\varepsilon^2(\mathbb{R}^d)$ and its expansion in Bloch functions $\tilde{f}_n \in L^2(B)$ as in Theorem 2.5 and also $\tilde{f}_n \in C^1(B)$. Further, let the $\psi_{n,\mathbf{k}}$ be continuously differentiable in \mathbf{k} . Then*

$$\mathbf{r}f(\mathbf{r}) = \frac{1}{\lambda(B)} \int_B \sum_\nu \left[\psi_{\nu,\mathbf{k}}(\mathbf{r}) i \nabla_{\mathbf{k}} \tilde{f}_\nu(\mathbf{k}) + \psi_{\nu,\mathbf{k}}(\mathbf{r}) \sum_{\nu'} \tilde{f}_{\nu'}(\mathbf{k}) \langle i \nabla_{\mathbf{k}} u_{\nu',\mathbf{k}}, u_{\nu,\mathbf{k}} \rangle_P \right] d\mathbf{k}.$$

Proof Since the $u_{n,\mathbf{k}}(\mathbf{r})$ for $n \in \mathbb{N}$ form an ONB of $L_\varepsilon^2(P)$ with periodic boundary conditions and (componentwise) $i \nabla_{\mathbf{k}} u_{n,\mathbf{k}} \in L_\varepsilon^2(P)$ with periodic BCs, we can expand

$$i \nabla_{\mathbf{k}} u_{n,\mathbf{k}} = \sum_{n'} u_{n',\mathbf{k}} \langle i \nabla_{\mathbf{k}} u_{n,\mathbf{k}}, u_{n',\mathbf{k}} \rangle_P.$$

Consider the product gradient

$$\begin{aligned} \nabla_{\mathbf{k}} \underbrace{(e^{i\mathbf{k}\cdot\mathbf{r}} u_{n,\mathbf{k}}(\mathbf{r}) \tilde{f}_n(\mathbf{k}))}_{=\psi_{n,\mathbf{k}}(\mathbf{r})} &= i\mathbf{r}\psi_{n,\mathbf{k}}(\mathbf{r})\tilde{f}_n(\mathbf{k}) + e^{i\mathbf{k}\cdot\mathbf{r}}\nabla_{\mathbf{k}}(u_{n,\mathbf{k}}(\mathbf{r})\tilde{f}_n(\mathbf{k})) \\ \Leftrightarrow \mathbf{r}\psi_{n,\mathbf{k}}(\mathbf{r})\tilde{f}_n(\mathbf{k}) &= -i\nabla_{\mathbf{k}}(\psi_{n,\mathbf{k}}(\mathbf{r})\tilde{f}_n(\mathbf{k})) + ie^{i\mathbf{k}\cdot\mathbf{r}}\nabla_{\mathbf{k}}(u_{n,\mathbf{k}}(\mathbf{r})\tilde{f}_n(\mathbf{k})). \end{aligned}$$

Now, finally, calculate

$$\begin{aligned} \mathbf{r}f(\mathbf{r}) &= \frac{1}{\lambda(B)} \int_B \sum_n \mathbf{r}\psi_{n,\mathbf{k}}(\mathbf{r})\tilde{f}_n(\mathbf{k})d\mathbf{k} \\ &= \frac{1}{\lambda(B)} \int_B \sum_n [-i\nabla_{\mathbf{k}}(e^{i\mathbf{k}\cdot\mathbf{r}} u_{n,\mathbf{k}}(\mathbf{r})\tilde{f}_n(\mathbf{k})) + e^{i\mathbf{k}\cdot\mathbf{r}}i\nabla_{\mathbf{k}}(u_{n,\mathbf{k}}(\mathbf{r})\tilde{f}_n(\mathbf{k}))]d\mathbf{k} \\ &\stackrel{(*)}{=} \frac{1}{\lambda(B)} \int_B \sum_n e^{i\mathbf{k}\cdot\mathbf{r}} \left[u_{n,\mathbf{k}}(\mathbf{r})i\nabla_{\mathbf{k}}\tilde{f}_n(\mathbf{k}) + \tilde{f}_n(\mathbf{k}) \sum_{n'} u_{n',\mathbf{k}} \langle i\nabla_{\mathbf{k}}u_{n,\mathbf{k}}, u_{n',\mathbf{k}} \rangle_P \right] d\mathbf{k} \\ &= \frac{1}{\lambda(B)} \int_B \sum_n \left[\psi_{n,\mathbf{k}}(\mathbf{r})i\nabla_{\mathbf{k}}\tilde{f}_n(\mathbf{k}) + \tilde{f}_n(\mathbf{k}) \sum_{n'} \psi_{n',\mathbf{k}} \langle i\nabla_{\mathbf{k}}u_{n,\mathbf{k}}, u_{n',\mathbf{k}} \rangle_P \right] d\mathbf{k} \\ &= \frac{1}{\lambda(B)} \int_B \sum_n \left[\psi_{n,\mathbf{k}}(\mathbf{r})i\nabla_{\mathbf{k}}\tilde{f}_n(\mathbf{k}) + \psi_{n,\mathbf{k}} \sum_{n'} \tilde{f}_{n'}(\mathbf{k}) \langle i\nabla_{\mathbf{k}}u_{n',\mathbf{k}}, u_{n,\mathbf{k}} \rangle_P \right] d\mathbf{k}, \end{aligned}$$

where the step marked (*) uses Corollary A.2 to eliminate the first term, which is possible since $\psi_{n,\mathbf{k}}(\mathbf{r})$ is \hat{L} -periodic in \mathbf{k} . Also take note of the slightly subtle switching of $n \leftrightarrow n'$ in the last step. \square

It should be noted that the $\psi_{n,\mathbf{k}}$ are usually not differentiable in \mathbf{k} where several eigenvalues are degenerate (see Section 6.1 for an elementary example). For our current purposes, we will continue in the illusion that this is not so. If the extra rigor of taking these discontinuities into account is desired, we need to reconsider the above proof. All the derivatives may be reinterpreted in the distributional sense without significantly changing the end result, but the conditions of Corollary A.2 are violated. The usual approach to this is to temporarily punch holes into the area of integration around these defects in the integrand and let the radius of these holes go to zero. We assume that there are no sets of discontinuity of $\psi_{n,\mathbf{k}}$ with positive Lebesgue measure in B . So if S is a connected set of points where $\psi_{n,\mathbf{k}}$ is discontinuous, the surface integral of Theorem A.1 becomes

$$\int_S (\psi_{n,\mathbf{k}}(\mathbf{r})^+ - \psi_{n,\mathbf{k}}(\mathbf{r})^-) dS$$

as the radius of the hole goes to zero, where S is now a “one-sided” surface. In three dimensions, the surface of integration is still two-dimensional, so if S is a point or locally line-like, then the surface integral will still be 0. Similarly, point discontinuities cause no worries in two dimensions.

For the final step, we combine Equation (3.1) and Lemma 3.4 to obtain

$$\mathbf{r}w_{n,\mathbf{0}}(\mathbf{r}) = \frac{1}{\lambda(B)} \int_B \sum_{\nu} \psi_{\nu,\mathbf{k}}(\mathbf{r}) \langle i\nabla_{\mathbf{k}}u_{n,\mathbf{k}}, u_{\nu,\mathbf{k}} \rangle_P d\mathbf{k}. \quad (3.4)$$

Theorem 3.5 [[Blo62], Appendix A] *Let $\psi_{n,\mathbf{k}}$ be continuously differentiable in \mathbf{k} . Then*

$$\langle \mathbf{r}w_{n,\mathbf{0}}, w_{m,\mathbf{R}} \rangle_{\mathbb{R}^d} = \frac{1}{\lambda(B)} \int_B e^{i\mathbf{k}\cdot\mathbf{R}} \langle i\nabla_{\mathbf{k}}u_{n,\mathbf{k}}, u_{m,\mathbf{k}} \rangle_P d\mathbf{k}$$

and

$$\langle \mathbf{r}^2 w_{n,\mathbf{0}}, w_{n,\mathbf{0}} \rangle_{\mathbb{R}^d} = \frac{1}{\lambda(B)} \int_B \langle i\nabla_{\mathbf{k}}u_{n,\mathbf{k}}, i\nabla_{\mathbf{k}}u_{n,\mathbf{k}} \rangle_P d\mathbf{k}.$$

Proof All we should have to do now is stick Equation (3.4) into the expectation value calculations. For the first equality, consider

$$\begin{aligned}
& \langle \mathbf{r}w_{n,\mathbf{0}}, w_{m,\mathbf{R}} \rangle_{\mathbb{R}^d} \\
&= \frac{1}{\lambda(B)^2} \left\langle \int_B \sum_{\nu} \psi_{\nu,\mathbf{k}} \langle i\nabla_{\mathbf{k}} u_{n,\mathbf{k}}, u_{\nu,\mathbf{k}} \rangle_P d\mathbf{k}, \int_B e^{-i\mathbf{k}' \cdot \mathbf{R}} \psi_{m,\mathbf{k}'} d\mathbf{k}' \right\rangle_{\mathbb{R}^d} \\
&= \frac{1}{\lambda(B)^2} \int_B \int_B \sum_{\nu} \langle i\nabla_{\mathbf{k}} u_{n,\mathbf{k}}, u_{\nu,\mathbf{k}} \rangle_P e^{i\mathbf{k}' \cdot \mathbf{R}} \langle \psi_{\nu,\mathbf{k}}, \psi_{m,\mathbf{k}'} \rangle_{\mathbb{R}^d} d\mathbf{k} d\mathbf{k}' \\
&\stackrel{(*)}{=} \frac{1}{\lambda(B)} \int_B \int_B \sum_{\nu} \langle i\nabla_{\mathbf{k}} u_{n,\mathbf{k}}, u_{\nu,\mathbf{k}} \rangle_P e^{i\mathbf{k}' \cdot \mathbf{R}} \delta_{\nu,m} \delta(\mathbf{k} - \mathbf{k}') d\mathbf{k} d\mathbf{k}' \\
&= \frac{1}{\lambda(B)} \int_B e^{i\mathbf{k} \cdot \mathbf{R}} \langle i\nabla_{\mathbf{k}} u_{n,\mathbf{k}}, u_{m,\mathbf{k}} \rangle_P d\mathbf{k},
\end{aligned}$$

where the step marked (*) uses Theorem 2.6. And for the second one,

$$\begin{aligned}
& \langle r^2 w_{n,\mathbf{0}}, w_{n,\mathbf{0}} \rangle_{\mathbb{R}^d} \\
&= \left\langle \frac{1}{\lambda(B)} \int_B \sum_{\nu} \psi_{\nu,\mathbf{k}} \langle i\nabla_{\mathbf{k}} u_{n,\mathbf{k}}, u_{\nu,\mathbf{k}} \rangle_P d\mathbf{k}, \frac{1}{\lambda(B)} \int_B \sum_{\nu'} \psi_{\nu',\mathbf{k}'} \langle i\nabla_{\mathbf{k}'} u_{n,\mathbf{k}'}, u_{\nu',\mathbf{k}'} \rangle_P d\mathbf{k}' \right\rangle_{\mathbb{R}^d} \\
&= \frac{1}{\lambda(B)^2} \int_B \int_B \sum_{\nu, \nu'} \langle i\nabla_{\mathbf{k}} u_{n,\mathbf{k}}, u_{\nu,\mathbf{k}} \rangle_P \langle u_{\nu',\mathbf{k}'}, i\nabla_{\mathbf{k}'} u_{n,\mathbf{k}'} \rangle_P \langle \psi_{\nu,\mathbf{k}}, \psi_{\nu',\mathbf{k}'} \rangle_{\mathbb{R}^d} d\mathbf{k} d\mathbf{k}' \\
&= \frac{1}{\lambda(B)} \int_B \int_B \sum_{\nu, \nu'} \langle i\nabla_{\mathbf{k}} u_{n,\mathbf{k}}, u_{\nu,\mathbf{k}} \rangle_P \langle u_{\nu',\mathbf{k}'}, i\nabla_{\mathbf{k}'} u_{n,\mathbf{k}'} \rangle_P \delta_{\nu,\nu'} \delta(\mathbf{k} - \mathbf{k}') d\mathbf{k} d\mathbf{k}' \\
&= \frac{1}{\lambda(B)} \int_B \sum_{\nu} \langle i\nabla_{\mathbf{k}} u_{n,\mathbf{k}}, u_{\nu,\mathbf{k}} \rangle_P \langle u_{\nu,\mathbf{k}}, i\nabla_{\mathbf{k}} u_{n,\mathbf{k}} \rangle_P d\mathbf{k} \\
&= \frac{1}{\lambda(B)} \int_B \left\langle \sum_{\nu} \langle i\nabla_{\mathbf{k}} u_{n,\mathbf{k}}, u_{\nu,\mathbf{k}} \rangle_P u_{\nu,\mathbf{k}}, i\nabla_{\mathbf{k}} u_{n,\mathbf{k}} \right\rangle_P d\mathbf{k} \\
&= \frac{1}{\lambda(B)} \int_B \langle i\nabla_{\mathbf{k}} u_{n,\mathbf{k}}, i\nabla_{\mathbf{k}} u_{n,\mathbf{k}} \rangle_P d\mathbf{k},
\end{aligned}$$

as claimed. □

Chapter 4

Localization in k -space

In this chapter, we will explore the details of Marzari and Vanderbilt’s approach to maximally localized Wannier functions. This chapter constitutes an attempt to give a readable introduction to the method presented in [MV97]. A nonlinear CG algorithm for spread minimization is explicitly presented. This algorithm is likely to be similar to what Marzari and Vanderbilt implement in their code [MVS04].

4.1 The spread functional

We are considering the second-moment functional

$$\Omega := \sum_n [\langle r^2 w_{n,\mathbf{0}}, w_{n,\mathbf{0}} \rangle_{\mathbb{R}^d} - |\langle \mathbf{r} w_{n,\mathbf{0}}, w_{n,\mathbf{0}} \rangle_{\mathbb{R}^d}|^2]$$

in which we add an extra term to get

$$\begin{aligned} \Omega &= \Omega - \sum_n \sum_{\mathbf{R}, m: [\mathbf{R}, m] \neq [\mathbf{0}, n]} [|\langle \mathbf{r} w_{m,\mathbf{R}}, w_{n,\mathbf{0}} \rangle_{\mathbb{R}^d}|^2 - |\langle \mathbf{r} w_{n,\mathbf{0}}, w_{m,\mathbf{R}} \rangle_{\mathbb{R}^d}|^2] \\ &= \sum_n \left[\langle r^2 w_{n,\mathbf{0}}, w_{n,\mathbf{0}} \rangle_{\mathbb{R}^d} - \sum_{\mathbf{R}, m} |\langle \mathbf{r} w_{n,\mathbf{0}}, w_{m,\mathbf{R}} \rangle_{\mathbb{R}^d}|^2 \right] \\ &\quad + \sum_n \sum_{\mathbf{R}, m: [\mathbf{R}, m] \neq [\mathbf{0}, n]} |\langle \mathbf{r} w_{n,\mathbf{0}}, w_{m,\mathbf{R}} \rangle_{\mathbb{R}^d}|^2. \end{aligned}$$

Following Marzari, we name subterms of this decomposition as follows, splitting the last part into diagonal “D” and off-diagonal “OD” parts.

$$\begin{aligned} \Omega_{\text{I}} &:= \sum_n \left[\langle r^2 w_{n,\mathbf{0}}, w_{n,\mathbf{0}} \rangle_{\mathbb{R}^d} - \sum_{\mathbf{R}, m} |\langle \mathbf{r} w_{n,\mathbf{0}}, w_{m,\mathbf{R}} \rangle_{\mathbb{R}^d}|^2 \right], \\ \Omega_{\text{OD}} &:= \sum_{n, m: n \neq m} \sum_{\mathbf{R}} |\langle \mathbf{r} w_{n,\mathbf{0}}, w_{m,\mathbf{R}} \rangle_{\mathbb{R}^d}|^2, \\ \Omega_{\text{D}} &:= \sum_n \sum_{\mathbf{R} \neq \mathbf{0}} |\langle \mathbf{r} w_{n,\mathbf{0}}, w_{n,\mathbf{R}} \rangle_{\mathbb{R}^d}|^2. \end{aligned}$$

In all cases, we are considering a number of J bands, so that both m and n run from $1, \dots, J$. The “I” suffix on Ω_{I} indicates that Ω_{I} is invariant under changes of the unitary mixing matrices, as we will see in the following theorem:

Theorem 4.1 *The functional Ω_{I} does not vary under unitary mixing of the Bloch functions: The mapping*

$$u_{n,\mathbf{k}} \mapsto \sum_{\nu=1}^J [U^{\mathbf{k}}]_{n,\nu} u_{\nu,\mathbf{k}}$$

with (potentially \mathbf{k} -dependent) unitary matrices $U^{\mathbf{k}} \in \mathbb{C}^{n \times n}$ leaves Ω_{I} unchanged.

Proof [This proof was given by my advisor, Prof. Dörfler.] Since $\{w_{m,\mathbf{R}}\}_{m,\mathbf{R}}$ constitutes an ONB of $L^2_\varepsilon(\mathbb{R}^d)$ (cf. Theorem 3.3), we may write

$$\begin{aligned} \sum_{n=1}^J \langle r^2 w_{n,\mathbf{0}}, w_{n,\mathbf{0}} \rangle_{\mathbb{R}^d} &= \sum_{n=1}^J \langle r w_{n,\mathbf{0}}, r w_{n,\mathbf{0}} \rangle_{\mathbb{R}^d} = \sum_{n=1}^J \|r w_{n,\mathbf{0}}\|_{\mathbb{R}^d}^2 \\ &= \sum_{n=1}^J \sum_{m=1}^{\infty} \sum_{\mathbf{R}} |\langle r w_{n,\mathbf{0}}, w_{m,\mathbf{R}} \rangle_{\mathbb{R}^d}|^2 \end{aligned}$$

and thus, using Theorem 3.5,

$$\begin{aligned} \Omega_{\text{I}} &= \sum_{n=1}^J \left[\langle r^2 w_{n,\mathbf{0}}, w_{n,\mathbf{0}} \rangle_{\mathbb{R}^d} - \sum_{m=1}^J \sum_{\mathbf{R}} |\langle r w_{n,\mathbf{0}}, w_{m,\mathbf{R}} \rangle_{\mathbb{R}^d}|^2 \right] \\ &= \sum_{n=1}^J \sum_{m=1}^{\infty} \sum_{\mathbf{R}} |\langle r w_{n,\mathbf{0}}, w_{m,\mathbf{R}} \rangle_{\mathbb{R}^d}|^2 - \sum_{n=1}^J \sum_{m=1}^J \sum_{\mathbf{R}} |\langle r w_{n,\mathbf{0}}, w_{m,\mathbf{R}} \rangle_{\mathbb{R}^d}|^2 \\ &= \sum_{n=1}^J \sum_{m=J+1}^{\infty} \sum_{\mathbf{R}} |\langle r w_{n,\mathbf{0}}, w_{m,\mathbf{R}} \rangle_{\mathbb{R}^d}|^2 \\ &= \sum_{n=1}^J \sum_{m=J+1}^{\infty} \sum_{\mathbf{R}} \frac{1}{\lambda(B)^2} \int_B \int_B e^{i(\mathbf{k}-\mathbf{k}') \cdot \mathbf{R}} \langle i \nabla_{\mathbf{k}} u_{n,\mathbf{k}}, u_{m,\mathbf{k}} \rangle_P \langle i \nabla_{\mathbf{k}'} u_{n,\mathbf{k}'}, u_{m,\mathbf{k}'} \rangle_P^* d\mathbf{k} d\mathbf{k}' \\ &= \sum_{n=1}^J \sum_{m=J+1}^{\infty} \frac{1}{\lambda(B)^2} \int_B \int_B \underbrace{\sum_{\mathbf{R}} e^{i(\mathbf{k}-\mathbf{k}') \cdot \mathbf{R}}}_{=\delta(\mathbf{k}-\mathbf{k}')\lambda(B)} \langle i \nabla_{\mathbf{k}} u_{n,\mathbf{k}}, u_{m,\mathbf{k}} \rangle_P \langle i \nabla_{\mathbf{k}'} u_{n,\mathbf{k}'}, u_{m,\mathbf{k}'} \rangle_P^* d\mathbf{k} d\mathbf{k}' \\ &= \sum_{n=1}^J \sum_{m=J+1}^{\infty} \frac{1}{\lambda(B)} \int_B \langle i \nabla_{\mathbf{k}} u_{n,\mathbf{k}}, u_{m,\mathbf{k}} \rangle_P \langle i \nabla_{\mathbf{k}} u_{n,\mathbf{k}}, u_{m,\mathbf{k}} \rangle_P^* d\mathbf{k}. \end{aligned}$$

Unitary mixing of the first J bands turns the scalar product into

$$\begin{aligned} \langle \nabla u_{n,\mathbf{k}}, u_{m,\mathbf{k}} \rangle_P &\mapsto \left\langle \nabla_{\mathbf{k}} \left[\sum_{\nu} [U^{\mathbf{k}}]_{n,\nu} u_{\nu,\mathbf{k}} \right], u_{m,\mathbf{k}} \right\rangle_P \\ &= \left\langle \sum_{\nu} \nabla_{\mathbf{k}} [U^{\mathbf{k}}]_{n,\nu} u_{\nu,\mathbf{k}} + \sum_{\nu} [U^{\mathbf{k}}]_{n,\nu} \nabla_{\mathbf{k}} u_{\nu,\mathbf{k}}, u_{m,\mathbf{k}} \right\rangle_P \\ &= \sum_{\nu} \nabla_{\mathbf{k}} [U^{\mathbf{k}}]_{n,\nu} \langle u_{\nu,\mathbf{k}}, u_{m,\mathbf{k}} \rangle_P + \sum_{\nu} [U^{\mathbf{k}}]_{n,\nu} \langle \nabla_{\mathbf{k}} u_{\nu,\mathbf{k}}, u_{m,\mathbf{k}} \rangle_P \\ &= \nabla_{\mathbf{k}} [U^{\mathbf{k}}]_{n,m} + \sum_{\nu} [U^{\mathbf{k}}]_{n,\nu} \langle \nabla_{\mathbf{k}} u_{\nu,\mathbf{k}}, u_{m,\mathbf{k}} \rangle_P. \end{aligned}$$

By virtue of Corollary A.2, the first term disappears as soon as it is placed under the Brillouin-zone integral, and we obtain

$$\begin{aligned} \Omega_{\text{I}} &\mapsto \frac{1}{\lambda(B)} \sum_{m=J+1}^{\infty} \sum_{\nu,\nu'} \underbrace{\sum_{n=1}^J [U^{\mathbf{k}}]_{n,\nu} [(U^{\mathbf{k}})^*]_{n,\nu'}}_{=[\text{Id}]_{\nu,\nu'}=\delta_{\nu,\nu'}} \int_B \langle \nabla_{\mathbf{k}} u_{\nu,\mathbf{k}}, u_{m,\mathbf{k}} \rangle_P \langle \nabla_{\mathbf{k}} u_{\nu',\mathbf{k}}, u_{m,\mathbf{k}} \rangle_P^* d\mathbf{k} \\ &= \sum_{m=J+1}^{\infty} \sum_{\nu} \int_B \langle \nabla_{\mathbf{k}} u_{\nu,\mathbf{k}}, u_{m,\mathbf{k}} \rangle_P \langle \nabla_{\mathbf{k}} u_{\nu',\mathbf{k}}, u_{m,\mathbf{k}} \rangle_P^* d\mathbf{k} \\ &= \Omega_{\text{I}}, \end{aligned}$$

as claimed. \square

The fact proven in the previous theorem is not essential to our further studies, but it is helpful within the implementation.

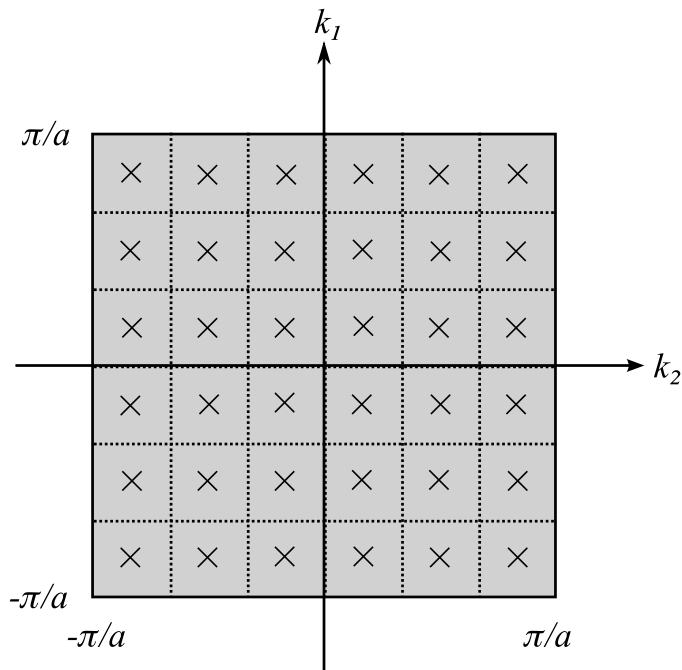


Figure 4.1. A 6×6 Monkhorst-Pack mesh in a simple quadratic Brillouin zone. The mesh consists of the points marked “ \times ”.

4.2 A mesh in k -space

Our next goal will be to obtain an easily computable expression for Ω . As is already apparent from Theorem 3.5, integrals over the Brillouin zone will play a big role. Monkhorst and Pack’s [MP76] approach to computing such integrals is still the most widespread one today. We will only use the grid generation part of their work to find a regularly spaced set \mathcal{K} of points within the Brillouin zone, as illustrated in Figure 4.1. In order to compute the value of a Brillouin zone integral, we will typically replace

$$\frac{1}{\lambda(B)} \int_B f(\mathbf{k}) d\mathbf{k} \quad \text{with} \quad \frac{1}{N} \sum_{\mathbf{k} \in \mathcal{K}} f(\mathbf{k}),$$

where $N := \#\mathcal{K}$. To simplify the notation, a sum over \mathbf{k} shall always be understood to run over \mathcal{K} for the remainder of this chapter.

4.3 Finite difference formulae in k -space

Let $\mathcal{S} := \{\mathbf{b}_1, \dots, \mathbf{b}_n\}$ be the “stencil” of vectors connecting a point in \mathbf{k} -space to its nearest neighbors in the \mathbf{k} -space discretization discussed above. To approximate \mathbf{k} -gradient expressions by finite differences on the mesh \mathcal{K} , we define a discretized gradient

$$\nabla^\Delta f(\mathbf{k}) := \sum_{\mathbf{b} \in \mathcal{S}} w_{\mathbf{b}} \mathbf{b} [f(\mathbf{k} + \mathbf{b}) - f(\mathbf{k})],$$

with weights $w_{\mathbf{b}} \in \mathbb{R}$. Note that, for symmetry reasons, these weights only vary with the Euclidean length of \mathbf{b} and should be viewed as “per-shell” weights. In the simple-cubic lattice considered here, a single shell will always suffice. In general, for the gradient expression to have a consistency order of at least 1, we need to postulate

$$\sum_{\mathbf{b} \in \mathcal{S}} w_{\mathbf{b}} b_\mu b_\nu = \delta_{\mu, \nu} \quad \Leftrightarrow \quad \sum_{\mathbf{b} \in \mathcal{S}} w_{\mathbf{b}} \mathbf{b} \otimes \mathbf{b} = \text{Id},$$

which necessitates several shells for more complex three-dimensional meshes. To see the need for the above condition, consider a linear function $f : B \rightarrow \mathbb{C}$ which maps $\mathbf{k} \mapsto f_0 + \mathbf{g} \cdot \mathbf{k}$:

$$\begin{aligned} g_\nu &\stackrel{!}{=} [\nabla^\Delta f(\mathbf{k})]_\nu = \sum_{\mathbf{b} \in \mathcal{S}} w_b b_\nu [f(\mathbf{k} + \mathbf{b}) - f(\mathbf{k})] \\ &= \sum_{\mathbf{b} \in \mathcal{S}} w_b b_\nu [f_0 + \mathbf{g} \cdot (\mathbf{k} + \mathbf{b}) - f_0 - \mathbf{g} \cdot \mathbf{k}] \\ &= \sum_{\mathbf{b} \in \mathcal{S}} w_b b_\nu [\mathbf{g} \cdot \mathbf{b}] = \sum_{\mathbf{b} \in \mathcal{S}} w_b b_\nu \sum_{\mu} g_\mu b_\mu \\ &= \sum_{\mu} g_\mu \underbrace{\sum_{\mathbf{b} \in \mathcal{S}} w_b b_\nu b_\mu}_{=\delta_{\mu,\nu}} = g_\nu. \end{aligned}$$

So, a consistency order of 1 holds if the weights and directions obey the condition above. As a convention for the rest of this chapter, a sum over \mathbf{b} shall always be understood as running over all $\mathbf{b} \in \mathcal{S}$.

Lemma 4.2 *For the weights and directions w_b and $\mathbf{b} \in \mathcal{S}$ introduced here and an arbitrary vector $\mathbf{a} \in \mathbb{C}^d$, we have*

$$\sum_{\mathbf{b}} w_b |\mathbf{b} \cdot \mathbf{a}|^2 = a^2.$$

Proof By simple calculation:

$$\sum_{\mathbf{b}} w_b |\mathbf{b} \cdot \mathbf{a}|^2 = \sum_{\mathbf{b}} w_b (\mathbf{b} \cdot \mathbf{a})(\mathbf{b} \cdot \mathbf{a})^* = \mathbf{a} \left(\sum_{\mathbf{b}} w_b \mathbf{b} \otimes \mathbf{b} \right) \mathbf{a}^* = a^2,$$

which confirms our claim. \square

As an easy consequence, consider a discretization of $|\nabla f(\mathbf{k})|^2$ for the affine f defined above, where we would expect g^2 :

$$\begin{aligned} |\nabla f(\mathbf{k})|^{\Delta,2} &:= \sum_{\mathbf{b}} w_b |f(\mathbf{k} + \mathbf{b}) - f(\mathbf{k})|^2 \\ &= \sum_{\mathbf{b}} w_b |\mathbf{b} \cdot \mathbf{g}|^2 = g^2. \end{aligned}$$

Despite the tempting looks, in general we have $|\nabla^\Delta f(\mathbf{k})|^2 \neq |\nabla f(\mathbf{k})|^{\Delta,2}$.

$$\begin{aligned} |\nabla^\Delta f(\mathbf{k})|^2 &= \left[\sum_{\mathbf{b}} w_b \mathbf{b} [f(\mathbf{k} + \mathbf{b}) - f(\mathbf{k})] \right] \cdot \left[\sum_{\mathbf{b}'} w_{b'} \mathbf{b}' [f(\mathbf{k} + \mathbf{b}') - f(\mathbf{k})] \right]^* \\ &= \sum_{\mathbf{b}, \mathbf{b}'} w_b \mathbf{b} [f(\mathbf{k} + \mathbf{b}) - f(\mathbf{k})] \cdot w_{b'} \mathbf{b}' [f(\mathbf{k} + \mathbf{b}') - f(\mathbf{k})]^* \\ &= \sum_{\mathbf{b}} w_b \sum_{\substack{\mathbf{b}' \\ \neq \delta_{\mathbf{b}, \mathbf{b}'}}} \underbrace{w_{b'} \mathbf{b} \cdot \mathbf{b}'}_{\neq \delta_{\mathbf{b}, \mathbf{b}'}} [f(\mathbf{k} + \mathbf{b}) - f(\mathbf{k})] [f(\mathbf{k} + \mathbf{b}') - f(\mathbf{k})]^* \\ &\neq \sum_{\mathbf{b}} w_b |f(\mathbf{k} + \mathbf{b}) - f(\mathbf{k})|^2 = |\nabla f(\mathbf{k})|^{\Delta,2}. \end{aligned}$$

4.4 The discretized spread functional

Before we delve into the actual discretization procedure, we define

$$M_{n,m}^{\mathbf{k},\mathbf{b}} := \langle u_{n,\mathbf{k}+\mathbf{b}}, u_{m,\mathbf{k}} \rangle_P. \quad (4.1)$$

Corollary 4.3 M is \hat{L} -periodic, i.e. $M^{\mathbf{k}+\mathbf{K},\mathbf{b}} = M^{\mathbf{k},\mathbf{b}}$ for $\mathbf{k} \in \mathcal{K}$, $\mathbf{K} \in \hat{L}$, $\mathbf{b} \in \mathcal{S}$.

Proof For $\mathbf{K} \in \hat{L}$, we have

$$u_{n,\mathbf{k}+\mathbf{K}}(\mathbf{r}) = e^{-i(\mathbf{k}+\mathbf{K})\cdot\mathbf{r}}\psi_{n,\mathbf{k}+\mathbf{K}}(\mathbf{r}) = e^{-i(\mathbf{k}+\mathbf{K})\cdot\mathbf{r}}\psi_{n,\mathbf{k}}(\mathbf{r}) = e^{-i\mathbf{K}\cdot\mathbf{r}}u_{n,\mathbf{k}}(\mathbf{r}).$$

So,

$$M_{n,m}^{\mathbf{k}+\mathbf{K},\mathbf{b}} = \langle u_{n,\mathbf{k}+\mathbf{K}+\mathbf{b}}, u_{m,\mathbf{k}+\mathbf{K}} \rangle_P = \langle e^{-i\mathbf{K}\cdot\mathbf{r}}u_{n,\mathbf{k}+\mathbf{b}}, e^{-i\mathbf{K}\cdot\mathbf{r}}u_{m,\mathbf{k}} \rangle_P = M_{n,m}^{\mathbf{k},\mathbf{b}},$$

as claimed. \square

Using the results from Section 4.3, the n th Wannier center according to Theorem 3.5

$$\langle \mathbf{r}w_{n,\mathbf{0}}, w_{n,\mathbf{0}} \rangle_{\mathbb{R}^d} = \frac{1}{\lambda(B)} \int_B \langle i\nabla_{\mathbf{k}}u_{n,\mathbf{k}}, u_{n,\mathbf{k}} \rangle_P d\mathbf{k}$$

may be approximated as

$$\begin{aligned} \langle \mathbf{r}w_{n,\mathbf{0}}, w_{n,\mathbf{0}} \rangle_{\mathbb{R}^d}^{\Delta} &:= \frac{i}{N} \sum_{\mathbf{k}} \langle \nabla_{\mathbf{k}}^{\Delta} u_{n,\mathbf{k}}, u_{n,\mathbf{k}} \rangle_P \\ &= \frac{i}{N} \sum_{\mathbf{k}} \left\langle \sum_{\mathbf{b}} w_{\mathbf{b}} \mathbf{b} [u_{n,\mathbf{k}+\mathbf{b}} - u_{n,\mathbf{k}}], u_{n,\mathbf{k}} \right\rangle_P \\ &= \frac{i}{N} \sum_{\mathbf{k},\mathbf{b}} w_{\mathbf{b}} \mathbf{b} [\langle u_{n,\mathbf{k}+\mathbf{b}}, u_{n,\mathbf{k}} \rangle_P - \langle u_{n,\mathbf{k}}, u_{n,\mathbf{k}} \rangle_P] \\ &= \frac{i}{N} \sum_{\mathbf{k},\mathbf{b}} w_{\mathbf{b}} \mathbf{b} [M_{n,n}^{\mathbf{k},\mathbf{b}} - 1]. \end{aligned}$$

We will see shortly that this definition still has some problems, so it is marked as “preliminary” by a “ \sim ” subscript. Following the $|\nabla f(\mathbf{k})|^{\Delta,2}$ pattern from Section 4.3, the r^2 component of the spread functional discretizes to

$$\begin{aligned} \langle r^2 w_{n,\mathbf{0}}, w_{n,\mathbf{0}} \rangle_{\mathbb{R}^d}^{\Delta} &:= \frac{1}{N} \sum_{\mathbf{k}} \langle \nabla u_{n,\mathbf{k}}, \nabla_{\mathbf{k}} u_{n,\mathbf{k}} \rangle_P^{\Delta} \\ &= \frac{1}{N} \sum_{\mathbf{k},\mathbf{b}} w_{\mathbf{b}} \langle u_{n,\mathbf{k}+\mathbf{b}} - u_{n,\mathbf{k}}, u_{n,\mathbf{k}+\mathbf{b}} - u_{n,\mathbf{k}} \rangle_P \\ &= \frac{1}{N} \sum_{\mathbf{k},\mathbf{b}} w_{\mathbf{b}} [\langle u_{n,\mathbf{k}+\mathbf{b}}, u_{n,\mathbf{k}+\mathbf{b}} \rangle_P - 2 \operatorname{Re} \langle u_{n,\mathbf{k}+\mathbf{b}}, u_{n,\mathbf{k}} \rangle_P + \langle u_{n,\mathbf{k}}, u_{n,\mathbf{k}} \rangle_P] \\ &= \frac{1}{N} \sum_{\mathbf{k},\mathbf{b}} w_{\mathbf{b}} [2 - 2 \operatorname{Re} M_{n,n}^{\mathbf{k},\mathbf{b}}]. \end{aligned}$$

4.5 A Problem and its Solution

These two expressions would be all nice and well if it wasn't for one small defect. If we translate all Bloch modes by a constant vector $\mathbf{R} \in L$, we would expect the discretized Wannier centers and spreads to reflect these changes just like their analytic counterparts do, namely:

$$\begin{aligned} \langle \mathbf{r}w_{n,\mathbf{0}}, w_{n,\mathbf{0}} \rangle_{\mathbb{R}^d}^{\Delta} &\mapsto \langle \mathbf{r}w_{n,\mathbf{0}}, w_{n,\mathbf{0}} \rangle_{\mathbb{R}^d}^{\Delta} + \mathbf{R}, \\ \langle r^2 w_{n,\mathbf{0}}, w_{n,\mathbf{0}} \rangle_{\mathbb{R}^d}^{\Delta} &\mapsto \langle r^2 w_{n,\mathbf{0}}, w_{n,\mathbf{0}} \rangle_{\mathbb{R}^d}^{\Delta} + 2\langle \mathbf{r}w_{n,\mathbf{0}}, w_{n,\mathbf{0}} \rangle_{\mathbb{R}^d}^{\Delta} \cdot \mathbf{R} + R^2. \end{aligned}$$

However, for the discretizations marked with “ \sim ”, this is not the case. Since a translation of the Bloch modes by \mathbf{R} amounts to changing $\psi_{n,\mathbf{k}}(\mathbf{r}) \mapsto e^{i\mathbf{k}\cdot\mathbf{R}}\psi_{n,\mathbf{k}}(\mathbf{r})$ for all \mathbf{k} and hence to changing $M_{n,n}^{\mathbf{k},\mathbf{b}} \mapsto e^{i\mathbf{b}\cdot\mathbf{R}}M_{n,n}^{\mathbf{k},\mathbf{b}}$, we currently get

$$\langle \mathbf{r}w_{n,\mathbf{0}}, w_{n,\mathbf{0}} \rangle_{\mathbb{R}^d, \text{trans}}^{\Delta} = \frac{i}{N} \sum_{\mathbf{k},\mathbf{b}} w_{\mathbf{b}} \mathbf{b} [e^{i\mathbf{b}\cdot\mathbf{R}}M_{n,n}^{\mathbf{k},\mathbf{b}} - 1],$$

which does not simplify further. We will change around the terms in the series expansion of $M_{n,n}^{\mathbf{k},\mathbf{b}}$ past the second and third order for these two expressions to make things work out, while aiming for a logarithmic term in order to transform a multiplication by $e^{-i\mathbf{b}\cdot\mathbf{R}}$ into the required addition. In order to do this, we need to understand the series involved. Suppose

$$M_{n,n}^{\mathbf{k},\beta\mathbf{b}} = 1 + i\mu_1\beta + \frac{1}{2}\mu_2\beta^2 + O(\beta^3).$$

We will prove $\mu_1 \in \mathbb{R}$, while μ_2 may be an arbitrary complex number. Consider the Taylor expansion

$$u_{n,\mathbf{k}+\beta\mathbf{b}} = u_{n,\mathbf{k}} + \beta\mathbf{b} \cdot \nabla_{\mathbf{k}} u_{n,\mathbf{k}} + O(\beta^2)$$

where the derivative is understood to be evaluated at $\beta = 0$ and \mathbf{b} is fixed. Then, consider

$$\begin{aligned} 1 &= \langle u_{n,\mathbf{k}+\beta\mathbf{b}}, u_{n,\mathbf{k}+\beta\mathbf{b}} \rangle_P \\ 1 &= \langle u_{n,\mathbf{k}} + \beta\mathbf{b} \cdot \nabla_{\mathbf{k}} u_{n,\mathbf{k}}, u_{n,\mathbf{k}} + \beta\mathbf{b} \cdot \nabla_{\mathbf{k}} u_{n,\mathbf{k}} \rangle_P + O(\beta^2) \\ 1 &= \langle u_{n,\mathbf{k}}, u_{n,\mathbf{k}} \rangle_P + 2\beta \operatorname{Re} \langle \mathbf{b} \cdot \nabla_{\mathbf{k}} u_{n,\mathbf{k}}, u_{n,\mathbf{k}} \rangle_P + O(\beta^2) \\ 0 &= 2\beta \operatorname{Re} \underbrace{\langle \mathbf{b} \cdot \nabla_{\mathbf{k}} u_{n,\mathbf{k}}, u_{n,\mathbf{k}} \rangle_P}_{=i\mu_1} + O(\beta^2), \end{aligned}$$

which proves $\operatorname{Im} \mu_1 = 0$. Considering the expressions we obtained for $\langle \mathbf{r}w_{n,\mathbf{0}}, w_{n,\mathbf{0}} \rangle_{\mathbb{R}^d}^\Delta$ and $\langle r^2w_{n,\mathbf{0}}, w_{n,\mathbf{0}} \rangle_{\mathbb{R}^d}^\Delta$, we have

$$\begin{aligned} M_{n,n}^{\mathbf{k},\beta\mathbf{b}} - 1 &= i\mu_1\beta + O(\beta^2), \\ 2 - 2 \operatorname{Re} M_{n,n}^{\mathbf{k},\beta\mathbf{b}} &= 2 - 2[1 + \frac{1}{2} \operatorname{Re}[\mu_2]\beta^2] = -\operatorname{Re}[\mu_2]\beta^2 + O(\beta^3). \end{aligned}$$

While aiming to maintain the right hand sides of the equations above, we can change the left hand sides to read

$$\begin{aligned} i \operatorname{Im} \ln M_{n,n}^{\mathbf{k},\beta\mathbf{b}} &= i\mu_1\beta + O(\beta^2), \\ 1 - |M_{n,n}^{\mathbf{k},\beta\mathbf{b}}|^2 + \mu_1^2\beta^2 &= 1 - \left(1 + i\mu_1\beta + \frac{1}{2}\mu_2\beta^2\right) \left(1 - i\mu_1\beta + \frac{1}{2}\mu_2^*\beta^2\right) + \mu_1^2\beta^2 \\ &= 1 - \left(1 - i\mu_1\beta + \frac{1}{2}\mu_2^*\beta^2 + i\mu_1\beta + \mu_1^2\beta^2 + \frac{1}{2}\mu_2\beta^2\right) + \mu_1^2\beta^2 + O(\beta^3) \\ &= -\operatorname{Re}[\mu_2]\beta^2 + O(\beta^3). \end{aligned}$$

If we modify the two parts of the spread functional to their final forms

$$\begin{aligned} \langle \mathbf{r}w_{n,\mathbf{0}}, w_{n,\mathbf{0}} \rangle_{\mathbb{R}^d}^\Delta &:= \frac{i}{N} \sum_{\mathbf{k},\mathbf{b}} w_b \mathbf{b} [i \arg M_{n,n}^{\mathbf{k},\mathbf{b}}] = -\frac{1}{N} \sum_{\mathbf{k},\mathbf{b}} w_b \mathbf{b} \arg M_{n,n}^{\mathbf{k},\mathbf{b}}, \\ \langle r^2w_{n,\mathbf{0}}, w_{n,\mathbf{0}} \rangle_{\mathbb{R}^d}^\Delta &:= \frac{1}{N} \sum_{\mathbf{k},\mathbf{b}} w_b \left[1 - |M_{n,n}^{\mathbf{k},\mathbf{b}}|^2 + [\arg M_{n,n}^{\mathbf{k},\mathbf{b}}]^2\right], \end{aligned}$$

we can see that

$$\begin{aligned} \langle \mathbf{r}w_{n,\mathbf{0}}, w_{n,\mathbf{0}} \rangle_{\mathbb{R}^d, \text{trans}}^\Delta &= \frac{1}{N} \sum_{\mathbf{k},\mathbf{b}} w_b \mathbf{b} [\mathbf{b} \cdot \mathbf{R} - \arg M_{n,n}^{\mathbf{k},\mathbf{b}}] \\ &= \langle w_{n,\mathbf{0}}, \mathbf{r}w_{n,\mathbf{0}} \rangle_{\mathbb{R}^d}^\Delta + \mathbf{R}, \end{aligned}$$

and, using Lemma 4.2,

$$\begin{aligned} \langle r^2w_{n,\mathbf{0}}, w_{n,\mathbf{0}} \rangle_{\mathbb{R}^d, \text{trans}}^\Delta &= \frac{1}{N} \sum_{\mathbf{k},\mathbf{b}} w_b \left[1 - |M_{n,n}^{\mathbf{k},\mathbf{b}}|^2 + [\arg M_{n,n}^{\mathbf{k},\mathbf{b}} - \mathbf{b} \cdot \mathbf{R}]^2\right] \\ &= \langle r^2w_{n,\mathbf{0}}, w_{n,\mathbf{0}} \rangle_{\mathbb{R}^d}^\Delta + \frac{1}{N} \sum_{\mathbf{k},\mathbf{b}} w_b [-2\mathbf{b} \cdot \mathbf{R} \arg M_{n,n}^{\mathbf{k},\mathbf{b}} + (\mathbf{b} \cdot \mathbf{R})^2] \\ &= \langle r^2w_{n,\mathbf{0}}, w_{n,\mathbf{0}} \rangle_{\mathbb{R}^d}^\Delta + 2\langle \mathbf{r}w_{n,\mathbf{0}}, w_{n,\mathbf{0}} \rangle_{\mathbb{R}^d}^\Delta \cdot \mathbf{R} + \frac{1}{N} \sum_{\mathbf{k},\mathbf{b}} w_b (\mathbf{b} \cdot \mathbf{R})^2 \\ &= \langle r^2w_{n,\mathbf{0}}, w_{n,\mathbf{0}} \rangle_{\mathbb{R}^d}^\Delta + 2\langle \mathbf{r}w_{n,\mathbf{0}}, w_{n,\mathbf{0}} \rangle_{\mathbb{R}^d}^\Delta \cdot \mathbf{R} + R^2, \end{aligned}$$

so these new expressions fulfill the above requirements.

4.6 Decomposition of the new spread functional

To be able to attack the expressions for Ω_I and Ω_{OD} , we will need an expression for

$$\begin{aligned}
& \sum_{\mathbf{R}} |\langle \mathbf{r}w_{n,0}, w_{m,\mathbf{R}} \rangle_{\mathbb{R}^d}|^2 = \sum_{\mathbf{R}} \langle \mathbf{r}w_{n,0}, w_{m,\mathbf{R}} \rangle_{\mathbb{R}^d} \langle \mathbf{r}w_{n,0}, w_{m,\mathbf{R}} \rangle_{\mathbb{R}^d}^* \\
&= \sum_{\mathbf{R}} \frac{1}{\lambda(B)} \int_B e^{i\mathbf{k}\cdot\mathbf{R}} \langle i\nabla_{\mathbf{k}} u_{n,\mathbf{k}}, u_{m,\mathbf{k}} \rangle_P \left[\frac{1}{\lambda(B)} \int_B e^{i\mathbf{k}'\cdot\mathbf{R}} \langle i\nabla_{\mathbf{k}'} u_{n,\mathbf{k}'}, u_{m,\mathbf{k}'} \rangle_P \right]^* d\mathbf{k}' d\mathbf{k} \\
&= \sum_{\mathbf{R}} \frac{1}{\lambda(B)} \int_B e^{i\mathbf{k}\cdot\mathbf{R}} \langle i\nabla_{\mathbf{k}} u_{n,\mathbf{k}}, u_{m,\mathbf{k}} \rangle_P \left[\frac{1}{\lambda(B)} \int_B e^{-i\mathbf{k}'\cdot\mathbf{R}} \langle i\nabla_{\mathbf{k}'} u_{n,\mathbf{k}'}, u_{m,\mathbf{k}'} \rangle_P \right]^* d\mathbf{k}' d\mathbf{k} \\
&= \frac{1}{\lambda(B)^2} \int_B \langle i\nabla_{\mathbf{k}} u_{n,\mathbf{k}}, u_{m,\mathbf{k}} \rangle_P \int_B \langle i\nabla_{\mathbf{k}'} u_{n,\mathbf{k}'}, u_{m,\mathbf{k}'} \rangle_P^* \underbrace{\sum_{\mathbf{R}} e^{i(\mathbf{k}-\mathbf{k}')\cdot\mathbf{R}}}_{=\delta(\mathbf{k}-\mathbf{k}')\lambda(B)} d\mathbf{k}' d\mathbf{k} \\
&= \frac{1}{\lambda(B)} \int_B |\langle i\nabla_{\mathbf{k}} u_{n,\mathbf{k}}, u_{m,\mathbf{k}} \rangle_P|^2 d\mathbf{k},
\end{aligned}$$

which we discretize by the $|\nabla \cdot|^{\Delta,2}$ scheme from Section 4.3:

$$\begin{aligned}
\sum_{\mathbf{R}} |\langle \mathbf{r}w_{n,0}, w_{m,\mathbf{R}} \rangle_{\mathbb{R}^d}|^{\Delta,2} &:= \frac{1}{N} \sum_{\mathbf{k}} |\langle \nabla_{\mathbf{k}} u_{n,\mathbf{k}}, u_{m,\mathbf{k}} \rangle_P|^{\Delta,2} \\
&= \frac{1}{N} \sum_{\mathbf{k}} \sum_{\mathbf{b}} w_b |M_{n,m}^{\mathbf{k},\mathbf{b}} - \delta_{m,n}|^2.
\end{aligned}$$

In order to match the corrections from Section 4.5, we need to make some changes:

$$\sum_{\mathbf{R}} |\langle \mathbf{r}w_{m,\mathbf{R}}, w_{0,n} \rangle_{\mathbb{R}^d}|^{\Delta,2} := \begin{cases} \frac{1}{N} \sum_{\mathbf{k},\mathbf{b}} w_b [\arg M_{n,n}^{\mathbf{k},\mathbf{b}}]^2 & m = n, \\ \frac{1}{N} \sum_{\mathbf{k},\mathbf{b}} w_b |M_{m,n}^{\mathbf{k},\mathbf{b}}|^2 & m \neq n. \end{cases}$$

This enables us to derive formulae for Ω_I^{Δ} and Ω_{OD}^{Δ} :

$$\begin{aligned}
\Omega_I^{\Delta} &:= \sum_n \left[\langle r^2 w_{n,0}, w_{n,0} \rangle_{\mathbb{R}^d}^{\Delta} - \sum_{\mathbf{R},m} |\langle \mathbf{r}w_{n,0}, w_{m,\mathbf{R}} \rangle_{\mathbb{R}^d}|^{\Delta,2} \right] \\
&= \frac{1}{N} \sum_n \sum_{\mathbf{k},\mathbf{b}} w_b \left[1 - |M_{n,n}^{\mathbf{k},\mathbf{b}}|^2 + [\arg M_{n,n}^{\mathbf{k},\mathbf{b}}]^2 - \sum_{m:n:m \neq n} |M_{n,m}^{\mathbf{k},\mathbf{b}}|^2 - [\arg M_{n,n}^{\mathbf{k},\mathbf{b}}]^2 \right] \\
&= \frac{1}{N} \sum_{\mathbf{k},\mathbf{b}} w_b \left[J - \sum_{m,n} |M_{n,m}^{\mathbf{k},\mathbf{b}}|^2 \right], \\
\Omega_{OD}^{\Delta} &:= \sum_{m,n:m \neq n} \sum_{\mathbf{R}} |\langle \mathbf{r}w_{n,0}, w_{m,\mathbf{R}} \rangle_{\mathbb{R}^d}|^{\Delta,2} = \frac{1}{N} \sum_{m,n:m \neq n} \sum_{\mathbf{k},\mathbf{b}} w_b |M_{n,m}^{\mathbf{k},\mathbf{b}}|^2.
\end{aligned}$$

Adding the above two expressions gives a particularly simple result:

$$\begin{aligned}
\Omega_{I,OD}^{\Delta} := \Omega_I^{\Delta} + \Omega_{OD}^{\Delta} &= \frac{1}{N} \sum_{\mathbf{k},\mathbf{b}} w_b \left[J - \sum_{m,n} |M_{n,m}^{\mathbf{k},\mathbf{b}}|^2 + \sum_{m \neq n} |M_{n,m}^{\mathbf{k},\mathbf{b}}|^2 \right] \\
&= \frac{1}{N} \sum_{\mathbf{k},\mathbf{b}} w_b \sum_n [1 - |M_{n,n}^{\mathbf{k},\mathbf{b}}|^2].
\end{aligned}$$

As preparation for the calculations leading up to our expression for $\Omega_{\mathbb{D}}^{\Delta}$, we prove the equality (that again uses Lemma 4.2)

$$\begin{aligned}
& \frac{1}{N} \sum_{\mathbf{k}, \mathbf{b}} w_b \left[\arg M_{n,n}^{\mathbf{k}, \mathbf{b}} \mathbf{b} \cdot \langle \mathbf{r}w_{n, \mathbf{0}}, w_{n, \mathbf{0}} \rangle_{\mathbb{R}^d} + (\mathbf{b} \cdot \langle \mathbf{r}w_{n, \mathbf{0}}, w_{n, \mathbf{0}} \rangle_{\mathbb{R}^d}^{\Delta})^2 \right] \\
&= \langle \mathbf{r}w_{n, \mathbf{0}}, w_{n, \mathbf{0}} \rangle_{\mathbb{R}^d}^{\Delta} \cdot \frac{1}{N} \sum_{\mathbf{k}, \mathbf{b}} w_b \mathbf{b} \arg M_{n,n}^{\mathbf{k}, \mathbf{b}} + \frac{1}{N} \sum_{\mathbf{k}, \mathbf{b}} (\mathbf{b} \cdot \langle \mathbf{r}w_{n, \mathbf{0}}, w_{n, \mathbf{0}} \rangle_{\mathbb{R}^d}^{\Delta})^2 \\
&= \langle \mathbf{r}w_{n, \mathbf{0}}, w_{n, \mathbf{0}} \rangle_{\mathbb{R}^d}^{\Delta} \cdot \frac{1}{N} \sum_{\mathbf{k}, \mathbf{b}} w_b \mathbf{b} \arg M_{n,n}^{\mathbf{k}, \mathbf{b}} + \sum_{\mathbf{b}} (\mathbf{b} \cdot \langle \mathbf{r}w_{n, \mathbf{0}}, w_{n, \mathbf{0}} \rangle_{\mathbb{R}^d}^{\Delta})^2 \\
&= \langle \mathbf{r}w_{n, \mathbf{0}}, w_{n, \mathbf{0}} \rangle_{\mathbb{R}^d}^{\Delta} \cdot \left(-\langle \mathbf{r}w_{n, \mathbf{0}}, w_{n, \mathbf{0}} \rangle_{\mathbb{R}^d}^{\Delta} \right) + |\langle \mathbf{r}w_{n, \mathbf{0}}, w_{n, \mathbf{0}} \rangle_{\mathbb{R}^d}^{\Delta}|^2 \\
&= 0.
\end{aligned}$$

Recall that

$$\Omega_{\mathbb{D}} := \sum_n \sum_{\mathbf{R} \neq \mathbf{0}} |\langle \mathbf{r}w_{n, \mathbf{0}}, w_{n, \mathbf{R}} \rangle_{\mathbb{R}^d}|^2.$$

In order to make use of the expression derived at the beginning of this section, we rewrite this slightly and define

$$\begin{aligned}
\Omega_{\mathbb{D}}^{\Delta} &:= \sum_n \left[\sum_{\mathbf{R}} |\langle \mathbf{r}w_{n, \mathbf{0}}, w_{n, \mathbf{R}} \rangle_{\mathbb{R}^d}|^{\Delta, 2} - |\langle \mathbf{r}w_{n, \mathbf{0}}, w_{n, \mathbf{0}} \rangle_{\mathbb{R}^d}^{\Delta}|^2 \right] \\
&= \sum_n \left[\frac{1}{N} \sum_{\mathbf{k}, \mathbf{b}} w_b [\arg M_{n,n}^{\mathbf{k}, \mathbf{b}}]^2 - \frac{1}{N^2} \left| \sum_{\mathbf{k}', \mathbf{b}'} w_{b'} \mathbf{b}' \arg M_{n,n}^{\mathbf{k}', \mathbf{b}'} \right|^2 \right] \\
&= \frac{1}{N} \sum_n \left[\sum_{\mathbf{k}, \mathbf{b}} w_b [\arg M_{n,n}^{\mathbf{k}, \mathbf{b}}]^2 - \frac{1}{N} \sum_{\mathbf{k}, \mathbf{b}} w_b \mathbf{b} \arg M_{n,n}^{\mathbf{k}, \mathbf{b}} \cdot \sum_{\mathbf{k}', \mathbf{b}'} w_{b'} \mathbf{b}' \arg M_{n,n}^{\mathbf{k}', \mathbf{b}'} \right] \\
&= \frac{1}{N} \sum_n \sum_{\mathbf{k}, \mathbf{b}} w_b \arg M_{n,n}^{\mathbf{k}, \mathbf{b}} \left[\arg M_{n,n}^{\mathbf{k}, \mathbf{b}} - \mathbf{b} \cdot \frac{1}{N} \sum_{\mathbf{k}', \mathbf{b}'} w_{b'} \mathbf{b}' \arg M_{n,n}^{\mathbf{k}', \mathbf{b}'} \right] \\
&= \frac{1}{N} \sum_n \sum_{\mathbf{k}, \mathbf{b}} w_b \left[[\arg M_{n,n}^{\mathbf{k}, \mathbf{b}}]^2 + \arg M_{n,n}^{\mathbf{k}, \mathbf{b}} \mathbf{b} \cdot \langle w_{n, \mathbf{0}}, \mathbf{r}w_{n, \mathbf{0}} \rangle_{\mathbb{R}^d}^{\Delta} \right] \\
&= \frac{1}{N} \sum_n \sum_{\mathbf{k}, \mathbf{b}} w_b \left[[\arg M_{n,n}^{\mathbf{k}, \mathbf{b}}]^2 + 2 \arg M_{n,n}^{\mathbf{k}, \mathbf{b}} \mathbf{b} \cdot \langle w_{n, \mathbf{0}}, \mathbf{r}w_{n, \mathbf{0}} \rangle_{\mathbb{R}^d}^{\Delta} + (\mathbf{b} \cdot \langle w_{n, \mathbf{0}}, \mathbf{r}w_{n, \mathbf{0}} \rangle_{\mathbb{R}^d}^{\Delta})^2 \right] \\
&= \frac{1}{N} \sum_n \sum_{\mathbf{k}, \mathbf{b}} w_b \left[\arg M_{n,n}^{\mathbf{k}, \mathbf{b}} + \mathbf{b} \cdot \langle w_{n, \mathbf{0}}, \mathbf{r}w_{n, \mathbf{0}} \rangle_{\mathbb{R}^d}^{\Delta} \right]^2.
\end{aligned}$$

4.7 The gradient of the spread functional

We are dealing with a function $\Omega : (\mathbb{C}^{J \times J})^{\mathcal{K}} \rightarrow \mathbb{R}$ whose value we are asked to minimize. We choose to do so by means of a gradient descent. But what is a gradient in this situation? Just the fact that $\Omega_{\mathbb{I}, \text{OD}}^{\Delta}$ uses the absolute value of a complex number (and thus a complex conjugation) is a hint at the fact that Ω is not differentiable in the holomorphic sense. Fortunately, it is only real-valued. So, the best thing we can hope for in this case is to have partial derivatives like $\partial\Omega/\partial \text{Re}[U_{m,n}^{\mathbf{k}}]$ and $\partial\Omega/\partial \text{Im}[U_{m,n}^{\mathbf{k}}]$. Any “holomorphic” derivative of Ω is likely ill-defined.

In what follows, we will examine two approaches of dealing with this problem. Both will consist of giving an expression that will predict a “small change” $d\Omega$ of the spread functional resulting from a small change in the mixing matrices $dU^{\mathbf{k}}$. In regular real analysis in \mathbb{R}^n , you can take an inner product $\nabla f \cdot \mathbf{x}$ of a gradient ∇f and a vector \mathbf{x} to predict how a scalar function’s value might change when moving in the direction of \mathbf{x} . In this case, the gradient is *defined* through partial derivatives. Here, we will go the

opposite route. We will *define* the gradient vector by the condition that

$$\nabla f(\mathbf{x}_0) \cdot \mathbf{x} \stackrel{\text{!}}{=} \lim_{t \rightarrow 0} \frac{f(\mathbf{x}_0 + t\mathbf{x}) - f(\mathbf{x}_0)}{t},$$

for any \mathbf{x} , and not by partial derivatives at all. Since our “points” and “directions” are really \mathbf{k} -dependent complex-valued matrices, it is not at all clear how this inner product “ \cdot ” should be defined. In fact, two completely different approaches exist, yielding different algorithms with wildly different performance characteristics. We will present each in turn, beginning with an approach that is closer to the classical view of partial derivatives.

4.7.1 A straightforward approach to the gradient

When differentiating a function $\Omega : \mathbb{C} \rightarrow \mathbb{R}$, we might be led to write the two partial derivatives $\partial\Omega(x)/\partial \operatorname{Re} x$ and $\partial\Omega(x)/\partial \operatorname{Im} x$ as a single complex number $\partial\Omega(x)/\partial \operatorname{Re} x + i\partial\Omega(x)/\partial \operatorname{Im} x$. But note the subtlety here. This derivative *looks* just like a complex number and is tempting to confuse with a complex derivative, but in fact it is a gradient vector in \mathbb{R}^2 that just happens to use complex notation. You should especially observe that this kind of derivative is *not* \mathbb{C} -linear.

As mentioned above, the defining demand that we place on a gradient vector is that computing an inner product of the gradient and an arbitrarily chosen unit simple vector in $(\mathbb{C}^{J \times J})^{\mathcal{K}}$ will yield the directional derivative of Ω in the given direction. The analogy with \mathbb{R}^2 helps and leads us to the definition

$$\begin{aligned} \langle \cdot, \cdot \rangle_{\nabla\Omega} &: (\mathbb{C}^{J \times J})^{\mathcal{K}} \times (\mathbb{C}^{J \times J})^{\mathcal{K}} \rightarrow \mathbb{C} \\ \langle A, B \rangle_{\nabla\Omega} &:= \frac{1}{\lambda(B)} \int_B \sum_{m,n} A_{m,n}^{\mathbf{k}} * B_{m,n}^{\mathbf{k}} d\mathbf{k} \approx \frac{1}{N} \sum_{\mathbf{k}} \sum_{m,n} A_{m,n}^{\mathbf{k}} * B_{m,n}^{\mathbf{k}} \end{aligned}$$

where $(a + ib) * (c + id) := ac + bd$. A few trivial properties:

Proposition 4.4 For $A, B, C \in (\mathbb{C}^{J \times J})^{\mathcal{K}}$,

- a) $\langle A + B, C \rangle_{\nabla\Omega} = \langle A, C \rangle_{\nabla\Omega} + \langle B, C \rangle_{\nabla\Omega}$,
- b) For $\alpha \in \mathbb{R}$, $\langle \alpha A, B \rangle_{\nabla\Omega} = \alpha \langle A, B \rangle_{\nabla\Omega}$,
- c) $\langle A, B \rangle_{\nabla\Omega} = \langle B, A \rangle_{\nabla\Omega}$,
- d) $\langle A, A \rangle_{\nabla\Omega} \geq 0$ and $\langle A, A \rangle_{\nabla\Omega} = 0 \Leftrightarrow A = 0$.

For $A, B, C, D \in (\mathbb{R}^{J \times J})^{\mathcal{K}}$, $\langle A + iB, C + iD \rangle_{\nabla\Omega} = \langle A, C \rangle_{\nabla\Omega} + \langle B, D \rangle_{\nabla\Omega}$.

In other words, $\langle \cdot, \cdot \rangle_{\nabla\Omega}$ is an \mathbb{R} -linear inner product on a \mathbb{C} -valued space.

Lemma 4.5

- a) For $A, B \in (\mathbb{R}^{J \times J})^{\mathcal{K}}$ with $A^{\mathbf{k}} = -(A^{\mathbf{k}})^T$ and $B^{\mathbf{k}} = (B^{\mathbf{k}})^T$ for $\mathbf{k} \in \mathcal{K}$,

$$\langle A, B \rangle_{\nabla\Omega} = 0.$$

- b) For $A, B \in (\mathbb{C}^{J \times J})^{\mathcal{K}}$ with $A^{\mathbf{k}} = -(A^{\mathbf{k}})^H$ and $B^{\mathbf{k}} = (B^{\mathbf{k}})^H$ for $\mathbf{k} \in \mathcal{K}$,

$$\langle A, B \rangle_{\nabla\Omega} = 0.$$

Proof All this is straightforward calculation:

- a) For $a, b \in \mathbb{R}$, $a * b = ab$.

$$\langle A, B \rangle_{\nabla\Omega} = \frac{1}{\lambda(B)} \int_B \left[\sum_{[m,n]: m < n} \left[A_{m,n}^{\mathbf{k}} B_{m,n}^{\mathbf{k}} + \underbrace{A_{n,m}^{\mathbf{k}}}_{=-A_{m,n}^{\mathbf{k}}} \underbrace{B_{n,m}^{\mathbf{k}}}_{=B_{m,n}^{\mathbf{k}}} \right] + \sum_m \left[\underbrace{A_{m,m}^{\mathbf{k}}}_{=0} B_{m,m}^{\mathbf{k}} \right] \right] d\mathbf{k} = 0.$$

b) Let $A = A^R + iA^I$ and $B = B^R + iB^I$ with $A^R, A^I, B^R, B^I \in (\mathbb{R}^{J \times J})^{\mathcal{K}}$. Then $A^R = -(A^R)^T$, $A^I = (A^I)^T$, $B^R = (B^R)^T$, $B^I = -(B^I)^T$.

$$\begin{aligned} \langle A, B \rangle_{\nabla\Omega} &= \langle A^R + iA^I, B^R + iB^I \rangle_{\nabla\Omega} \\ &= \langle A^R, B^R \rangle_{\nabla\Omega} + \langle A^I, B^I \rangle_{\nabla\Omega} = 0. \quad (\text{using a)}) \end{aligned}$$

□

We will forge ahead with this definition for now, and consider the alternative definition afterwards.

4.7.2 Small changes to $U^{\mathbf{k}}$

Right now, our goal is to find out what happens to Ω if we apply a small change to our mixing matrix U , in the hopes of finding changes that minimize it. Strictly speaking, U has to be unitary, so any change we make to it had better not destroy this property. If we are going to perform a gradient descent, we need to find a way to stay within this set. A common local parametrization of the set of unitary matrices uses matrix exponentials of skew-hermitian matrices. This is easily verified; given a skew-hermitian W :

$$\exp(W)^H \exp W = \exp(W^H) \exp W = \exp(-W) \exp W = \text{Id}.$$

(and likewise for $\exp(tW) \exp(tW)^H$) Our ultimate goal is to find a gradient matrix G such that

$$\langle W, G \rangle_{\nabla\Omega} = \lim_{t \rightarrow 0} \frac{\Omega(\exp(tW)U) - \Omega(U)}{t},$$

for a given notion of a scalar product $\langle \cdot, \cdot \rangle_{\nabla\Omega}$. Since we're dividing by t , we can pretty safely disregard any components of the numerator of second or higher order in t . For ease of calculation, we will in fact “only” attempt to calculate G for

$$\langle W, G \rangle_{\nabla\Omega} \approx \lim_{t \rightarrow 0} \frac{\Omega((\text{Id} + tW)U) - \Omega(U)}{t}, \quad (4.2)$$

which is identical with the previous equation to the first order in t . Even this matrix is “roughly” unitary:

$$(\text{Id} + tW)^H (\text{Id} + tW) = \text{Id} + (tW)^H + tW + O(t^2) = \text{Id} + O(t^2).$$

This subsection shall be devoted to calculating the expression in the numerator of Equation (4.2), and we will worry about the final step involving the scalar product later. At the most basic level, Ω is affected by changes in the mixing matrix U through changes to the periodic Bloch modes. If we let $u_{n,\mathbf{k}}$ be the periodic Bloch modes as obtained by mixing throughout the current U , a multiplication by $\text{Id} + tW$ manifests itself as

$$u_{n,\mathbf{k}}(\mathbf{r}) \mapsto u_{n,\mathbf{k}}(\mathbf{r}) + \sum_m tW_{n,m}^{\mathbf{k}} u_{n,\mathbf{k}}(\mathbf{r}). \quad (4.3)$$

It is probably useful to note that this definition of tW is the transpose of the matrix dW discussed in [MV97] —in this way, the above equation retains the conventional index order of matrix multiplication. We will now estimate the effect of W on $M^{\mathbf{k},\mathbf{b}}$.

$$\begin{aligned} M_{n,m}^{\mathbf{k},\mathbf{b}} &\mapsto \left\langle u_{n,\mathbf{k}+\mathbf{b}} + \sum_{\nu} tW_{n,\nu}^{\mathbf{k}+\mathbf{b}} u_{\nu,\mathbf{k}+\mathbf{b}}, u_{m,\mathbf{k}} + \sum_{\mu} tW_{m,\mu}^{\mathbf{k}} u_{\mu,\mathbf{k}} \right\rangle_P \\ &= M_{n,m}^{\mathbf{k},\mathbf{b}} + \sum_{\nu} tW_{n,\nu}^{\mathbf{k}+\mathbf{b}} \langle u_{\nu,\mathbf{k}+\mathbf{b}}, u_{m,\mathbf{k}} \rangle_P + \sum_{\mu} t[W_{m,\mu}^{\mathbf{k}}]^* \langle u_{n,\mathbf{k}+\mathbf{b}}, u_{\mu,\mathbf{k}} \rangle_P + O(t^2) \\ &= M_{n,m}^{\mathbf{k},\mathbf{b}} + \sum_{\nu} tW_{n,\nu}^{\mathbf{k}+\mathbf{b}} M_{\nu,m}^{\mathbf{k},\mathbf{b}} + \sum_{\mu} t[W_{m,\mu}^{\mathbf{k}}]^* M_{n,\mu}^{\mathbf{k},\mathbf{b}} + O(t^2) \\ &= M_{n,m}^{\mathbf{k},\mathbf{b}} + \sum_{\nu} tW_{n,\nu}^{\mathbf{k}+\mathbf{b}} M_{\nu,m}^{\mathbf{k},\mathbf{b}} + \sum_{\mu} tM_{n,\mu}^{\mathbf{k},\mathbf{b}} [(W^{\mathbf{k}})^H]_{\mu,m} + O(t^2) \\ &= M_{n,m}^{\mathbf{k},\mathbf{b}} + [tW^{\mathbf{k}+\mathbf{b}} M^{\mathbf{k},\mathbf{b}}]_{n,m} + [M^{\mathbf{k},\mathbf{b}} (tW^{\mathbf{k}})^H]_{n,m} + O(t^2) \end{aligned}$$

So,

$$\Delta M_{n,n}^{\mathbf{k},\mathbf{b}} = [tW^{\mathbf{k}+\mathbf{b}}M^{\mathbf{k},\mathbf{b}}]_{n,n} + [M^{\mathbf{k},\mathbf{b}}(tW^{\mathbf{k}})^H]_{n,n}.$$

This is consistent with formula (44) in [MV97] since (let $\overset{M}{*}$ denote $*$ in the convention of [MV97])

- by comparing definition (4.3) and Equation (38) in [MV97], $(tW^{\mathbf{k}})^T = (tW^{\mathbf{k}})^T$ and
- by comparing definition (4.1) and Equation (25) in [MV97], $(M^{\mathbf{k},\mathbf{b}})^T = (M^{\mathbf{k},\mathbf{b}})^T$.

So, using the skew-hermiticity of $tW^{\mathbf{k}}$ in the first equality,

$$\begin{aligned} \Delta M_{n,n}^{\mathbf{k},\mathbf{b}} &= -[(M^{\mathbf{k},\mathbf{b}})^T (tW^{\mathbf{k}})^T]_{n,n} + [(tW^{\mathbf{k}+\mathbf{b}})^T (M^{\mathbf{k},\mathbf{b}})^T]_{n,n} \\ &= -[(tW^{\mathbf{k}})(M^{\mathbf{k},\mathbf{b}})]_{n,n} + [(M^{\mathbf{k},\mathbf{b}})(tW^{\mathbf{k}+\mathbf{b}})]_{n,n} = \Delta M_{n,n}^{\mathbf{k},\mathbf{b}}. \end{aligned}$$

It is easy to verify that $M^{\mathbf{k},\mathbf{b}} = (M^{\mathbf{k}+\mathbf{b},-\mathbf{b}})^H$. In the sequel, we will use

$$\begin{aligned} \Delta M_{n,n}^{\mathbf{k},\mathbf{b}} &= [tW^{\mathbf{k}+\mathbf{b}}(M^{\mathbf{k}+\mathbf{b},-\mathbf{b}})^H]_{n,n} + [M^{\mathbf{k},\mathbf{b}}(tW^{\mathbf{k}})^H]_{n,n} \\ &= [tW^{\mathbf{k}+\mathbf{b}}(M^{\mathbf{k}+\mathbf{b},-\mathbf{b}})^H]_{n,n} + [(M^{\mathbf{k},\mathbf{b}})^H (tW^{\mathbf{k}})^H]_{n,n} \\ &= [tW^{\mathbf{k}+\mathbf{b}}(M^{\mathbf{k}+\mathbf{b},-\mathbf{b}})^H]_{n,n} + [tW^{\mathbf{k}}(M^{\mathbf{k},\mathbf{b}})^H]_{n,n}^*. \end{aligned}$$

This is useful since we can substitute $\mathbf{k} + \mathbf{b} \mapsto \mathbf{k}$ in a sum over all $\mathbf{k} \in \mathcal{K}$ without changing anything, as we saw in Corollary 4.3 —the past-boundary terms can be understood to “wrap around” to the other side, so that we always sum over the same set of \mathbf{k} . Likewise, $-\mathbf{b} \mapsto \mathbf{b}$ does not change the terms we sum over, since $\mathbf{b} \in \mathcal{S}$ implies $-\mathbf{b} \in \mathcal{S}$. Now, let us consider

$$\begin{aligned} &\Delta \Omega_{\text{I,OD}}^{\Delta} \\ &= -\frac{1}{N} \sum_{\mathbf{k},\mathbf{b}} w_b \sum_n \Delta [|M_{n,n}^{\mathbf{k},\mathbf{b}}|^2] = -\frac{1}{N} \sum_{\mathbf{k},\mathbf{b}} w_b \sum_n [|M_{n,n}^{\mathbf{k},\mathbf{b}} + \Delta M_{n,n}^{\mathbf{k},\mathbf{b}}|^2 - |M_{n,n}^{\mathbf{k},\mathbf{b}}|^2] \\ &= -\frac{2}{N} \sum_{\mathbf{k},\mathbf{b}} w_b \sum_n \text{Re}[\Delta M_{n,n}^{\mathbf{k},\mathbf{b}} [M_{n,n}^{\mathbf{k},\mathbf{b}}]^*] + O(t^2) \\ &\approx -\frac{2}{N} \sum_{\mathbf{k},\mathbf{b}} w_b \sum_n \text{Re} \left[[tW^{\mathbf{k}+\mathbf{b}}(M^{\mathbf{k}+\mathbf{b},-\mathbf{b}})^H]_{n,n} [M_{n,n}^{\mathbf{k},\mathbf{b}}]^* + [tW^{\mathbf{k}}(M^{\mathbf{k},\mathbf{b}})^H]_{n,n}^* [M_{n,n}^{\mathbf{k},\mathbf{b}}]^* \right] \\ &= -\frac{2}{N} \sum_{\mathbf{k},\mathbf{b}} w_b \sum_n \text{Re} \left[[tW^{\mathbf{k}+\mathbf{b}}(M^{\mathbf{k}+\mathbf{b},-\mathbf{b}})^H]_{n,n} M_{n,n}^{\mathbf{k}+\mathbf{b},-\mathbf{b}} \right] \\ &\quad - \frac{2}{N} \sum_{\mathbf{k},\mathbf{b}} w_b \sum_n \text{Re} \left[[tW^{\mathbf{k}}(M^{\mathbf{k},\mathbf{b}})^H]_{n,n}^* [M_{n,n}^{\mathbf{k},\mathbf{b}}]^* \right] \\ &= -\frac{2}{N} \sum_{\mathbf{k},\mathbf{b}} w_{|\mathbf{b}|} \sum_n \text{Re} \left[[tW^{\mathbf{k}}(M^{\mathbf{k},\mathbf{b}})^H]_{n,n} M_{n,n}^{\mathbf{k},\mathbf{b}} \right] - \frac{2}{N} \sum_{\mathbf{k},\mathbf{b}} w_b \sum_n \text{Re} \left[[tW^{\mathbf{k}}(M^{\mathbf{k},\mathbf{b}})^H]_{n,n} M_{n,n}^{\mathbf{k},\mathbf{b}} \right]^* \\ &= -\frac{4}{N} \sum_{\mathbf{k},\mathbf{b}} w_b \sum_n \text{Re} \left[[tW^{\mathbf{k}}(M^{\mathbf{k},\mathbf{b}})^H]_{n,n} M_{n,n}^{\mathbf{k},\mathbf{b}} \right] \\ &= -\frac{4}{N} \sum_{\mathbf{k},\mathbf{b}} w_b \sum_n \text{Re} \left[\sum_{\nu} tW_{n,\nu}^{\mathbf{k}} \underbrace{[(M^{\mathbf{k},\mathbf{b}})^H]_{\nu,n} M_{n,n}^{\mathbf{k},\mathbf{b}}}_{R_{\nu,n}^{\mathbf{k},\mathbf{b}}} \right] \\ &= -\frac{4}{N} \sum_{\mathbf{k},\mathbf{b}} w_b \text{Re tr} [tW^{\mathbf{k}} R^{\mathbf{k},\mathbf{b}}]. \end{aligned}$$

Before we attack $\Delta\Omega_D$, we will consider this auxiliary calculation, using the same tricks as before:

$$\begin{aligned}
\Delta \arg M_{n,n}^{\mathbf{k},\mathbf{b}} &= \operatorname{Im} \Delta \ln M_{n,n}^{\mathbf{k},\mathbf{b}} = \operatorname{Im} [\ln(M_{n,n}^{\mathbf{k},\mathbf{b}} + \Delta M_{n,n}^{\mathbf{k},\mathbf{b}}) - \ln M_{n,n}^{\mathbf{k},\mathbf{b}}] \\
&= \operatorname{Im} \ln \left(1 + \frac{\Delta M_{n,n}^{\mathbf{k},\mathbf{b}}}{M_{n,n}^{\mathbf{k},\mathbf{b}}} \right) = \operatorname{Im} \frac{\Delta M_{n,n}^{\mathbf{k},\mathbf{b}}}{M_{n,n}^{\mathbf{k},\mathbf{b}}} + O(t^2) \\
&\approx \operatorname{Im} \frac{[tW^{\mathbf{k}+\mathbf{b}}(M^{\mathbf{k}+\mathbf{b},-\mathbf{b}})^H]_{n,n} + [tW^{\mathbf{k}}(M^{\mathbf{k},\mathbf{b}})^H]_{n,n}^*}{M_{n,n}^{\mathbf{k},\mathbf{b}}} \\
&= \operatorname{Im} \left[\frac{[tW^{\mathbf{k}+\mathbf{b}}(M^{\mathbf{k}+\mathbf{b},-\mathbf{b}})^H]_{n,n}}{M_{n,n}^{\mathbf{k},\mathbf{b}}} + \frac{[tW^{\mathbf{k}}(M^{\mathbf{k},\mathbf{b}})^H]_{n,n}^*}{[[M_{n,n}^{\mathbf{k},\mathbf{b}}]^*]^*} \right] \\
&= \operatorname{Im} \left[\frac{[tW^{\mathbf{k}+\mathbf{b}}(M^{\mathbf{k}+\mathbf{b},-\mathbf{b}})^H]_{n,n}}{[M_{n,n}^{\mathbf{k}+\mathbf{b},-\mathbf{b}}]^*} \right] - \operatorname{Im} \left[\frac{[tW^{\mathbf{k}}(M^{\mathbf{k},\mathbf{b}})^H]_{n,n}}{[M_{n,n}^{\mathbf{k},\mathbf{b}}]^*} \right] \\
&= \operatorname{Im} \left[\sum_{\nu} tW_{n,\nu}^{\mathbf{k}+\mathbf{b}} \frac{[(M^{\mathbf{k}+\mathbf{b},-\mathbf{b}})^H]_{\nu,n}}{[M_{n,n}^{\mathbf{k}+\mathbf{b},-\mathbf{b}}]^*} \right] - \operatorname{Im} \left[\sum_{\nu} tW_{n,\nu}^{\mathbf{k}} \underbrace{\frac{[(M^{\mathbf{k},\mathbf{b}})^H]_{\nu,n}}{[M_{n,n}^{\mathbf{k},\mathbf{b}}]^*}}_{\tilde{R}_{\nu,n}^{\mathbf{k},\mathbf{b}}} \right] \\
&= \operatorname{Im}[tW^{\mathbf{k}+\mathbf{b}} \tilde{R}^{\mathbf{k}+\mathbf{b},-\mathbf{b}}]_{n,n} - \operatorname{Im}[tW^{\mathbf{k}} \tilde{R}^{\mathbf{k},\mathbf{b}}]_{n,n}.
\end{aligned}$$

We can of course simplify

$$\tilde{R}_{\nu,n}^{\mathbf{k},\mathbf{b}} = \frac{[(M^{\mathbf{k},\mathbf{b}})^H]_{\nu,n}}{[M_{n,n}^{\mathbf{k},\mathbf{b}}]^*} = \left[\frac{M_{n,\nu}^{\mathbf{k},\mathbf{b}}}{M_{n,n}^{\mathbf{k},\mathbf{b}}} \right]^*.$$

Homing in on our target, we call

$$q_n^{\mathbf{k},\mathbf{b}} := \arg M_{n,n}^{\mathbf{k},\mathbf{b}} + \mathbf{b} \cdot \langle w_{n,\mathbf{0}}, \mathbf{r}w_{n,\mathbf{0}} \rangle_{\mathbb{R}^d} \triangle \in \mathbb{R},$$

note in passing that

$$\begin{aligned}
q_n^{\mathbf{k}+\mathbf{b},-\mathbf{b}} &= \arg M_{n,n}^{\mathbf{k}+\mathbf{b},-\mathbf{b}} - \mathbf{b} \cdot \langle w_{n,\mathbf{0}}, \mathbf{r}w_{n,\mathbf{0}} \rangle_{\mathbb{R}^d} \triangle \\
&= \arg [[M_{n,n}^{\mathbf{k},\mathbf{b}}]^*] - \mathbf{b} \cdot \langle w_{n,\mathbf{0}}, \mathbf{r}w_{n,\mathbf{0}} \rangle_{\mathbb{R}^d} \triangle \\
&= -q_n^{\mathbf{k},\mathbf{b}}
\end{aligned}$$

as well as, again using Lemma 4.2,

$$\begin{aligned}
\frac{1}{N} \sum_{\mathbf{k},\mathbf{b}} w_b \mathbf{b} q_n^{\mathbf{k},\mathbf{b}} &= \sum_{\mathbf{k},\mathbf{b}} w_b \mathbf{b} \left[\arg M_{n,n}^{\mathbf{k},\mathbf{b}} + \mathbf{b} \cdot \langle w_{n,\mathbf{0}}, \mathbf{r}w_{n,\mathbf{0}} \rangle_{\mathbb{R}^d} \triangle \right] \\
&= \frac{1}{N} \sum_{\mathbf{k},\mathbf{b}} w_b \mathbf{b} \arg M_{n,n}^{\mathbf{k},\mathbf{b}} + \sum_{\mathbf{b}} w_b \mathbf{b} \mathbf{b} \cdot \langle w_{n,\mathbf{0}}, \mathbf{r}w_{n,\mathbf{0}} \rangle_{\mathbb{R}^d} \triangle \\
&= \frac{1}{N} \sum_{\mathbf{k},\mathbf{b}} w_b \mathbf{b} \arg M_{n,n}^{\mathbf{k},\mathbf{b}} + \langle w_{n,\mathbf{0}}, \mathbf{r}w_{n,\mathbf{0}} \rangle_{\mathbb{R}^d} \triangle = \mathbf{0}
\end{aligned}$$

and write

$$\begin{aligned}
& \Delta\Omega_D^\Delta \\
&= \frac{1}{N} \sum_n \sum_{\mathbf{k}, \mathbf{b}} w_b \Delta [q_n^{\mathbf{k}, \mathbf{b}}]^2 = \frac{1}{N} \sum_n \sum_{\mathbf{k}, \mathbf{b}} w_b \left[[q_n^{\mathbf{k}, \mathbf{b}} + \Delta q_n^{\mathbf{k}, \mathbf{b}}]^2 - [q_n^{\mathbf{k}, \mathbf{b}}]^2 \right] \\
&= \frac{2}{N} \sum_n \sum_{\mathbf{k}, \mathbf{b}} w_b q_n^{\mathbf{k}, \mathbf{b}} \Delta q_n^{\mathbf{k}, \mathbf{b}} + O(t^2) \\
&\approx \frac{2}{N} \sum_{\mathbf{k}, \mathbf{b}} w_b \sum_n q_n^{\mathbf{k}, \mathbf{b}} \left[\text{Im}[tW^{\mathbf{k}+\mathbf{b}} \tilde{R}^{\mathbf{k}+\mathbf{b}, -\mathbf{b}}]_{n,n} - \text{Im}[tW^{\mathbf{k}} \tilde{R}^{\mathbf{k}, \mathbf{b}}]_{n,n} + \mathbf{b} \cdot \Delta \langle w_{n, \mathbf{0}}, \mathbf{r}w_{n, \mathbf{0}} \rangle_{\mathbb{R}^d}^\Delta \right] \\
&= \frac{2}{N} \sum_{\mathbf{k}, \mathbf{b}} w_b \sum_n q_n^{\mathbf{k}, \mathbf{b}} \left[-\text{Im}[tW^{\mathbf{k}} \tilde{R}^{\mathbf{k}, \mathbf{b}}]_{n,n} + \mathbf{b} \cdot \Delta \langle w_{n, \mathbf{0}}, \mathbf{r}w_{n, \mathbf{0}} \rangle_{\mathbb{R}^d}^\Delta \right] \\
&\quad - \frac{2}{N} \sum_{\mathbf{k}, \mathbf{b}} w_b \sum_n q_n^{\mathbf{k}+\mathbf{b}, -\mathbf{b}} \text{Im}[tW^{\mathbf{k}+\mathbf{b}} \tilde{R}^{\mathbf{k}+\mathbf{b}, -\mathbf{b}}]_{n,n} \\
&= \frac{2}{N} \sum_{\mathbf{k}, \mathbf{b}} w_b \sum_n q_n^{\mathbf{k}, \mathbf{b}} \left[-\text{Im}[tW^{\mathbf{k}} \tilde{R}^{\mathbf{k}, \mathbf{b}}]_{n,n} + \mathbf{b} \cdot \Delta \langle w_{n, \mathbf{0}}, \mathbf{r}w_{n, \mathbf{0}} \rangle_{\mathbb{R}^d}^\Delta \right] \\
&\quad - \frac{2}{N} \sum_{\mathbf{k}, \mathbf{b}} w_b \sum_n q_n^{\mathbf{k}, \mathbf{b}} \text{Im}[tW^{\mathbf{k}} \tilde{R}^{\mathbf{k}, \mathbf{b}}]_{n,n} \\
&= -\frac{4}{N} \sum_{\mathbf{k}, \mathbf{b}} w_b \sum_n q_n^{\mathbf{k}, \mathbf{b}} \text{Im}[tW^{\mathbf{k}} \tilde{R}^{\mathbf{k}, \mathbf{b}}]_{n,n} + \underbrace{\sum_n \frac{2}{N} \sum_{\mathbf{k}, \mathbf{b}} w_b q_n^{\mathbf{k}, \mathbf{b}} \mathbf{b} \cdot \Delta \langle w_{n, \mathbf{0}}, \mathbf{r}w_{n, \mathbf{0}} \rangle_{\mathbb{R}^d}^\Delta}_{=0} \\
&= -\frac{4}{N} \sum_{\mathbf{k}, \mathbf{b}} w_b \sum_n \text{Im} \left[\sum_\nu tW_{n, \nu}^{\mathbf{k}} \underbrace{q_n^{\mathbf{k}, \mathbf{b}} \tilde{R}_{\nu, n}^{\mathbf{k}, \mathbf{b}}}_{T_{\nu, n}^{\mathbf{k}, \mathbf{b}}} \right] \\
&= -\frac{4}{N} \sum_{\mathbf{k}, \mathbf{b}} w_b \text{Im} \text{tr} [tW^{\mathbf{k}} T^{\mathbf{k}, \mathbf{b}}]
\end{aligned}$$

4.7.3 A first gradient of Ω

We can easily regard any (constant) matrix as \mathbf{k} -dependent and use it with the inner product defined earlier. Since we defined it in a \mathbf{k} -averaging way, it simply degrades to

$$\langle A, B \rangle_{\nabla\Omega} = \sum_{\mu, \nu} A_{\mu, \nu} * B_{\mu, \nu}.$$

Let's start with some notation:

$$\begin{aligned}
\text{Sym}(A) &:= \frac{A + A^T}{2} && \text{for real matrices } A, \\
\text{Skew}(A) &:= \frac{A - A^T}{2} && \text{for real matrices } A, \\
\text{Herm}(A) &:= \text{Sym}(\text{Re}[A]) + i \text{Skew}(\text{Im}[A]) && \text{for complex matrices } A,
\end{aligned}$$

with their straightforward extensions to \mathbf{k} -dependent matrices. $\text{Sym}(A)$ is called the symmetric part of A , $\text{Skew}(A)$ is the skew-symmetric part of A , and $\text{Herm}(A)$ is the hermitian part of A . It is easy to see that these names are appropriate, i.e. that the symmetric part of A really is symmetric and so on.

As a first step towards our desired G , we derive the gradient of a subexpression. Let $W := W^R + iW^I$

and $B := B^R + iB^I$, with $W^R, W^I, B^R, B^I \in \mathbb{R}^{J \times J}$ and $t \in \mathbb{R}$. We are trying to find a G such that

$$\begin{aligned}
\langle W, G \rangle_{\nabla\Omega} &= \lim_{t \rightarrow 0} \frac{\operatorname{Re} \operatorname{tr}[(W_0 + tW)B] - \operatorname{Re} \operatorname{tr}[WB]}{t} = \lim_{t \rightarrow 0} \frac{\operatorname{Re} \operatorname{tr}[tWB]}{t} \\
&= \operatorname{Re} \operatorname{tr}[WB] = \sum_{\mu} \operatorname{Re}[WB]_{\mu,\mu} \\
&= \sum_{\mu,\nu} \operatorname{Re}[W_{\mu,\nu} B_{\nu,\mu}] = \sum_{\mu,\nu} [W_{\mu,\nu}^R B_{\nu,\mu}^R - W_{\mu,\nu}^I B_{\nu,\mu}^I] \\
\Rightarrow \sum_{\mu,\nu} W_{\mu,\nu} * G_{\mu,\nu} &= \sum_{\mu,\nu} [W_{\mu,\nu}^R B_{\nu,\mu}^R - W_{\mu,\nu}^I B_{\nu,\mu}^I] \\
\Rightarrow G &= [B^R]^T - i[B^I]^T = B^H.
\end{aligned}$$

The last implication stems from the fact that in the fully general case we can choose any W we like, especially $\mathbf{e}_{\mu} \otimes \mathbf{e}_{\nu}$ and $i\mathbf{e}_{\mu} \otimes \mathbf{e}_{\nu}$, and thus identify the real and imaginary parts of each entry of the matrix G one by one. (\mathbf{e}_{μ} is the μ th unit vector.)

However there's a hitch: our G needs to be skew-hermitian to fit the bill for our gradient search in a subsequent step of our gradient descent. We will want to use the matrix $-G$ as the new search direction W . But for arbitrary matrices B , the hermitian transpose B^H is not necessarily skew-hermitian. Fortunately, the fact that W is skew-hermitian gives us the chance to change G a little so that, thanks to Lemma 4.5,

$$\begin{aligned}
\langle W, G - \operatorname{Herm}(G) \rangle_{\nabla\Omega} &= \langle W, G \rangle_{\nabla\Omega}, \\
G - \operatorname{Herm}(G) &= (B^R)^T - i(B^I)^T - \operatorname{Sym}((B^R)^T) - i \operatorname{Skew}(-(B^I)^T) \\
&= (B^R)^T - i(B^I)^T - \frac{(B^R)^T + B^R}{2} + i \frac{(B^I)^T - B^I}{2} \\
&= \operatorname{Skew}((B^R)^T) - i \operatorname{Sym}(B^I),
\end{aligned}$$

which is clearly skew-hermitian as planned. Compared to Equation (41) in [MV97], I believe that this constitutes an essential difference. Similar to the above deduction,

$$\begin{aligned}
\langle W, G \rangle_{\nabla\Omega} &= \lim_{t \rightarrow 0} \frac{\operatorname{Re} \operatorname{tr}[(W_0 + tW)B] - \operatorname{Re} \operatorname{tr}[WB]}{t} = \lim_{t \rightarrow 0} \frac{\operatorname{Im} \operatorname{tr}[tWB]}{t} \\
&= \operatorname{Im} \operatorname{tr}[WB] = \sum_{\mu} \operatorname{Im}[WB]_{\mu,\mu} \\
&= \sum_{\mu,\nu} \operatorname{Im}[W_{\mu,\nu} B_{\nu,\mu}] = \sum_{\mu,\nu} [W_{\mu,\nu}^R B_{\nu,\mu}^I + W_{\mu,\nu}^I B_{\nu,\mu}^R] \\
\Rightarrow \sum_{\mu,\nu} W_{\mu,\nu} * G_{\mu,\nu} &= \sum_{\mu,\nu} [W_{\mu,\nu}^R B_{\nu,\mu}^I + W_{\mu,\nu}^I B_{\nu,\mu}^R] \\
\Rightarrow G &= [B^I]^T + i[B^R]^T = iB^H.
\end{aligned}$$

Next, we need to fix G 's non-skew-hermiticity:

$$\begin{aligned}
\langle W, G - \operatorname{Herm}((B^I)^T + iB^R) \rangle_{\nabla\Omega} &= \langle W, G \rangle_{\nabla\Omega}, \\
G - \operatorname{Herm}((B^I)^T + iB^R) &= (B^I)^T + i(B^R)^T - \operatorname{Sym}((B^I)^T) + i \operatorname{Skew}(B^R) \\
&= (B^I)^T + i(B^R)^T - \frac{(B^I)^T + B^I}{2} + i \frac{B^R - (B^R)^T}{2} \\
&= \operatorname{Skew}((B^I)^T) + i \operatorname{Sym}(B^R).
\end{aligned}$$

We're finally able to find a G with

$$\begin{aligned} \langle W, G \rangle_{\nabla\Omega} &= \lim_{t \rightarrow 0} \frac{\Delta\Omega}{t} = \lim_{t \rightarrow 0} \frac{\Delta\Omega_{I,OD} + \Delta\Omega_D}{t} \\ &= \frac{4}{N} \sum_{\mathbf{k}, \mathbf{b}} w_b \lim_{t \rightarrow 0} \frac{-\operatorname{Re} \operatorname{tr} [tW^{\mathbf{k}} R^{\mathbf{k}, \mathbf{b}}] - \operatorname{Im} \operatorname{tr} [tW^{\mathbf{k}} T^{\mathbf{k}, \mathbf{b}}]}{t} \\ \rightsquigarrow \frac{1}{N} \sum_{\mathbf{k}} \langle W^{\mathbf{k}}, G^{\mathbf{k}} \rangle_{\nabla\Omega} &= \frac{4}{N} \sum_{\mathbf{k}, \mathbf{b}} w_b \lim_{t \rightarrow 0} \frac{-\operatorname{Re} \operatorname{tr} [tW^{\mathbf{k}} R^{\mathbf{k}, \mathbf{b}}] - \operatorname{Im} \operatorname{tr} [tW^{\mathbf{k}} T^{\mathbf{k}, \mathbf{b}}]}{t}. \end{aligned}$$

Since this equation needs to hold for any $W^{\mathbf{k}}$, we are free to choose $W^{\mathbf{k}}$ in such a way that it is non-zero for only one \mathbf{k} at a time. Thus we can identify the $G^{\mathbf{k}}$ for each \mathbf{k} individually. Our problem now takes the easily solvable form:

$$\langle W^{\mathbf{k}}, G^{\mathbf{k}} \rangle_{\nabla\Omega} = 4 \sum_{\mathbf{b}} w_b \lim_{t \rightarrow 0} \frac{-\operatorname{Re} \operatorname{tr} [tW^{\mathbf{k}} R^{\mathbf{k}, \mathbf{b}}] - \operatorname{Im} \operatorname{tr} [tW^{\mathbf{k}} T^{\mathbf{k}, \mathbf{b}}]}{t}.$$

So,

$$G^{\mathbf{k}} = 4 \sum_{\mathbf{b}} w_b \left[-\operatorname{Skew}((\operatorname{Re} R^{\mathbf{k}, \mathbf{b}})^T) + i \operatorname{Sym}(\operatorname{Im} R^{\mathbf{k}, \mathbf{b}}) - \operatorname{Skew}(\operatorname{Im} (T^{\mathbf{k}, \mathbf{b}})^T) - i \operatorname{Sym}(\operatorname{Re} T^{\mathbf{k}, \mathbf{b}}) \right].$$

This result stands by itself as one possible notion of a “gradient” to be used for minimization. However, besides common ancestry, it has no relationship whatsoever with the one derived in [MV97]. In practice, it turns out that the above expression can be used to minimize Ω , but frequently only leads to local minima. In order to fix this, we will have to consider a different, less obvious inner-product-like expression, introduced next.

4.7.4 Marzari and Vanderbilt's gradient of Ω

The fact that the above derivation necessitated a corrective term to make up for the lack of skew-hermiticity in the gradient is a reason enough to ask whether there is a more “natural” approach that automatically leads to a skew-hermitian gradient. It turns out that there is, even though I am unaware of the reason why this might be so.

The key is to define the expression taking the role of the inner product in the following, different manner:

$$\begin{aligned} \langle \cdot, \cdot \rangle_{\nabla\Omega}^{\text{MV}} &: (\mathbb{C}^{J \times J})^{\mathcal{K}} \times (\mathbb{C}^{J \times J})^{\mathcal{K}} \rightarrow \mathbb{C}, \\ \langle A, B \rangle_{\nabla\Omega}^{\text{MV}} &:= \frac{1}{\lambda(B)} \int_B \operatorname{tr}[A^{\mathbf{k}} B^{\mathbf{k}}] d\mathbf{k} \approx \frac{1}{N} \sum_{\mathbf{k}} \operatorname{tr}[A^{\mathbf{k}} B^{\mathbf{k}}]. \end{aligned}$$

A few things come to mind immediately:

- Definiteness, or rather, lack thereof. $\langle A, A \rangle_{\nabla\Omega}^{\text{MV}} = \operatorname{tr}[AA] \geq 0$ is not necessarily true. To see this, consider $A \in \mathbb{C}^{1 \times 1}$ with $A = (i)$, for which $\langle A, A \rangle_{\nabla\Omega}^{\text{MV}} = -1$.
- Symmetry. Obviously, $\langle A, B \rangle_{\nabla\Omega}^{\text{MV}} = \langle B, A \rangle_{\nabla\Omega}^{\text{MV}}$.
- \mathbb{C} -Linearity. $\langle \cdot, \cdot \rangle_{\nabla\Omega}^{\text{MV}}$ is \mathbb{C} -linear in both arguments.
- The change that we are trying to predict by means of $\langle \cdot, \cdot \rangle_{\nabla\Omega}^{\text{MV}}$ is real-valued. There is no immediate reason that the value of $\langle \cdot, \cdot \rangle_{\nabla\Omega}^{\text{MV}}$ should not also have an imaginary part.

As such, this definition is somewhat unusual, and, by the letter of the mathematical definition, it does not yield an inner product on $(\mathbb{C}^{J \times J})^{\mathcal{K}}$.

Still, we will see that predicting a change in Ω using this construction actually works. For clarity, we will forgo the \mathbf{k} -dependency of the matrices involved, applying our demand that G be chosen such that for any skew-hermitian $W \in \mathbb{C}^{J \times J}$ and a given $B \in \mathbb{C}^{J \times J}$

$$\langle W, G \rangle_{\nabla\Omega}^{\text{MV}} = \lim_{t \rightarrow 0} \frac{\operatorname{Re} \operatorname{tr}[(W_0 + tW)B] - \operatorname{Re} \operatorname{tr}[WB]}{t} = \lim_{t \rightarrow 0} \frac{\operatorname{Re} \operatorname{tr}[tWB]}{t}$$

yields

$$\begin{aligned}
\langle W^{\mathbf{k}}, G_{\text{Re}} \rangle_{\nabla\Omega}^{\text{MV}} &\stackrel{!}{=} \lim_{t \rightarrow 0} \frac{\text{Re tr}[tWB]}{t} = \text{Re tr}[WB] \\
&= \frac{1}{2} \{ \text{tr}[WB] + \text{tr}[WB]^* \} \\
&= \frac{1}{2} \sum_{\mu, \nu} \{ W_{\mu, \nu} B_{\nu, \mu} + W_{\mu, \nu}^* B_{\nu, \mu}^* \} \\
&= \frac{1}{2} \sum_{\mu, \nu} \{ W_{\mu, \nu} B_{\nu, \mu} - W_{\nu, \mu} B_{\nu, \mu}^* \} \\
&= \frac{1}{2} \sum_{\mu, \nu} \{ W_{\mu, \nu} B_{\nu, \mu} - W_{\mu, \nu} B_{\mu, \nu}^* \} \\
&= \text{tr} \left[dW \frac{1}{2} (B - B^H) \right] = \left\langle W, \frac{1}{2} (B - B^H) \right\rangle_{\nabla\Omega}^{\text{MV}}.
\end{aligned}$$

Of course, choosing $G_{\text{Re}} := 1/2(B - B^H)$ is an obvious choice, but it is not quite clear that this is *the* unique choice. However, I have evidence from the 2×2 case that this is the only choice that will work across all W .

Likewise, we compute, once again for any skew-hermitian $W \in \mathbb{C}^{J \times J}$ and a given $B \in \mathbb{C}^{J \times J}$

$$\begin{aligned}
\langle W, G_{\text{Im}} \rangle_{\nabla\Omega}^{\text{MV}} &\stackrel{!}{=} \lim_{t \rightarrow 0} \frac{\text{Im tr}[tWB]}{t} = \text{Re tr}[WB] \\
&= \frac{1}{2i} \{ \text{tr}[WB] - \text{tr}[WB]^* \} \\
&= \frac{1}{2i} \sum_{\mu, \nu} \{ W_{\mu, \nu} B_{\nu, \mu} - W_{\mu, \nu}^* B_{\nu, \mu}^* \} \\
&= \frac{1}{2i} \sum_{\mu, \nu} \{ W_{\mu, \nu} B_{\nu, \mu} + W_{\nu, \mu} B_{\nu, \mu}^* \} \\
&= \frac{1}{2i} \sum_{\mu, \nu} \{ W_{\mu, \nu} B_{\nu, \mu} + W_{\mu, \nu} B_{\mu, \nu}^* \} \\
&= \text{tr} \left[dW \frac{1}{2i} (B + B^H) \right] = \left\langle W, \frac{1}{2i} (B + B^H) \right\rangle_{\nabla\Omega}^{\text{MV}},
\end{aligned}$$

making $G_{\text{Im}} := 1/(2i)(B + B^H)$ the obvious choice. Putting these two together, and considering the \mathbf{k} -dependency, the complete equation

$$\langle W^{\mathbf{k}}, G^{\mathbf{k}} \rangle_{\nabla\Omega} = 4 \sum_{\mathbf{b}} w_{\mathbf{b}} \lim_{t \rightarrow 0} \frac{-\text{Re tr} [tW^{\mathbf{k}} R^{\mathbf{k}, \mathbf{b}}] - \text{Im tr}[tW^{\mathbf{k}} T^{\mathbf{k}, \mathbf{b}}]}{t} \quad \text{for all } \mathbf{k} \in \mathcal{K}$$

gives us

$$\begin{aligned}
G^{\mathbf{k}} &= -4 \sum_{\mathbf{b}} w_{\mathbf{b}} \left[\frac{R^{\mathbf{k}, \mathbf{b}} - (R^{\mathbf{k}, \mathbf{b}})^H}{2} + \frac{T^{\mathbf{k}, \mathbf{b}} + (T^{\mathbf{k}, \mathbf{b}})^H}{2i} \right] \\
&= 2 \sum_{\mathbf{b}} w_{\mathbf{b}} \left[i(T^{\mathbf{k}, \mathbf{b}} + (T^{\mathbf{k}, \mathbf{b}})^H) - R^{\mathbf{k}, \mathbf{b}} + (R^{\mathbf{k}, \mathbf{b}})^H \right]
\end{aligned}$$

4.8 Minimizing the spread

In this section, we will put all the puzzle pieces together and describe the whole algorithm used to minimize the Wannier function spread. Here is a rough outline of the procedure, which we will describe in more detail later on:

1. Choose an initial (\mathbf{k} -dependent) mixing matrix U .

2. Compute the initial inner product matrices $M^{\mathbf{k},\mathbf{b},(0)}$, set $M^{\mathbf{k},\mathbf{b}} \leftarrow M^{\mathbf{k},\mathbf{b},(0)}$.
3. Set $R_{\text{last}}, D \leftarrow -G(U)$. [*begin CG minimization*]
4. Repeat the steps i. to vii. until $\Omega^\Delta(U)$ fails to become smaller in every step by at least some constant ε for 3 consecutive steps:

- i. Compute the gradient $G(U)$ of Ω^Δ ,
- ii. Perform a line search for the α that minimizes

$$\Omega(\exp(\alpha/(4 \sum_{\mathbf{b}} w_{\mathbf{b}})D)U),$$

using $\alpha = 1/2$ as a starting guess. Using Brent's method (chapter 5 of [Bre73]) after finding a bracket on a minimum works well here,

- iii. Update the mixing matrix $U \leftarrow \exp(\alpha/(4 \sum_{\mathbf{b}} w_{\mathbf{b}})D)U$ and the inner product matrix M ,
- iv. $R \leftarrow -G(U)$,
- v. $\beta \leftarrow \max\left(0, \frac{\langle R, R - R_{\text{last}} \rangle_{\nabla \Omega}}{\langle R_{\text{last}}, R_{\text{last}} \rangle_{\nabla \Omega}}\right)$, [*Polak-Ribière formula*]
- vi. $D \leftarrow R + \beta D$,
- vii. $R_{\text{last}} \leftarrow R$.

[*end CG minimization*]

5. Compute the Wannier functions according to

$$w_{n,\mathbf{0}}(\mathbf{r}) = \frac{1}{N} \sum_{\mathbf{k}} \sum_m U_{n,m}^{\mathbf{k}} \psi_{m,\mathbf{k}}(\mathbf{r}).$$

6. Verify that $w_{n,\mathbf{0}}$ has a constant overall phase, as conjectured in [MV97], and can be made real-valued by multiplication by a complex number of absolute value 1. When checking the phase, it may be useful to disregard elements close to 0.

The nonlinear CG used here was patterned after introductory material in [She94]. Let us elaborate on some of these steps in more detail.

4.8.1 The starting strategy

To begin our search for maximally-localized Wannier functions, we need a starting guess for our unitary mixing matrices $U^{\mathbf{k}}$. In a straightforward manner, we could initialize the $U^{\mathbf{k}}$ as identity matrices (which are trivially unitary). However, this turns out to be a bad idea.

Consider that the functional Ω is built on finite differences between values associated with the same band number at adjacent \mathbf{k} -vectors. These differences are certainly very sensitive to the band number assignment at each \mathbf{k} -point: If the number assignment at one \mathbf{k} -point does not match its neighbor's assignment, their difference will not reflect a proper derivative, since the difference will not approach 0 as we let $\mathbf{b} \rightarrow \mathbf{0}$. The number assignment, however, is part of the minimization problem: At each \mathbf{k} -point, we are free to choose a permutation matrix as a part of $U^{\mathbf{k}}$. In the face of degeneracies and general erratic band behavior, this numbering is notoriously hard to guess, especially if the Brillouin Zone discretization is fairly coarse. To make matters worse, CG is built for *differentiable* optimization, but picking a permutation matrix at each \mathbf{k} -point is not even a *continuous* optimization problem! As a result, it would be foolish to expect the CG minimization step to be of any help in this matter. Without a good starting guess for the $U^{\mathbf{k}}$, the minimization problem of Ω^Δ is bound to fail or get stuck in a local minimum.

So, what should we choose? Intuitively, Gaussian bell curves are nicely localized functions which at least partly resemble the functions we are looking for. So, let us go with those. Since we need a few different Gaussians, we will randomly pick $\mu_m \in [-0.9, 0.9]^d$ (in coordinates relative to $\mathcal{B}(L)$) as the center of the Gaussian and $\sigma_{m,n} \in [0.1, \min(\mu_{m,n}, 1 - \mu_{m,n})]$ as the second moment (for the band number $m = 1, \dots, J$ and the coordinate $n = 1, \dots, d$). Note that we do not even need to bother to

use randomized principal axes, so the ‘‘covariance matrix’’ $[\Sigma_m]_{\mu,\nu} := \delta_{\mu,\nu}\sigma_{m,\mu}$ of the Gaussian remains diagonal. Our random Gaussians are

$$g_m(\mathbf{r}) := \exp(-|\Sigma_m^{-1}(\mathbf{r} - \boldsymbol{\mu}_m)|^2).$$

Since we are only free to choose a mixture in the subspace of Bloch functions, we project the Gaussians into the span of the Bloch modes at each \mathbf{k} -point:

$$g_{m,\mathbf{k}} := \sum_{n=1}^J \langle g_m, \psi_{n,\mathbf{k}} \rangle_P \psi_{n,\mathbf{k}}.$$

But this will still not provide the unitary mixing matrix we desire: an orthonormalization step is necessary. Letting $S_{\mu,\nu}^{\mathbf{k}} := \langle g_{\mu,\mathbf{k}}, g_{\nu,\mathbf{k}} \rangle_P$, the definition

$$\tilde{g}_{m,\mathbf{k}} := \sum_{n=1}^J [S^{-1/2}]_{m,n} g_{n,\mathbf{k}}$$

as recommended in [MV97] provides the desired orthonormal functions $\tilde{g}_{m,\mathbf{k}}$. Their mixing matrix representations are

$$[U^{\mathbf{k}}]_{\mu,\nu} := \sum_{n=1}^J [S^{-1/2}]_{\mu,n} \langle g_n, \psi_{\nu,\mathbf{k}} \rangle_P,$$

which is what will serve as our starting guess.

A good measure of how well the starting strategy worked are the absolute values and arguments of the diagonal of $M^{\mathbf{k},\mathbf{b}}$ after the initially-guessed mixing matrix is first applied. For $\mathbf{b} \rightarrow 0$, it would be natural to expect

$$\begin{aligned} |M_{n,n}^{\mathbf{k},\mathbf{b}}|^2 &\rightarrow 1, \\ \arg M_{n,n}^{\mathbf{k},\mathbf{b}} &\rightarrow 0. \end{aligned}$$

Values close to these indicate that the starting strategy worked. Note that it is quite natural to have slightly different values along the Brillouin zone boundary.

Subsequently, minimization of the $\Omega_{\text{OD}}^{\Delta}$ part of the spread functional boils down to codiagonalizing a set of given matrices. The use of a specialized algorithm (such as [BGBM93] with [CS96]) for this purpose as an extension of the given starting strategy did unfortunately not yield measurable benefits.

4.8.2 The initial inner products $M^{\mathbf{k},\mathbf{b},(0)}$

In step 2 above, obtaining the initial inner product matrices through the relation

$$M_{m,n}^{\mathbf{k},\mathbf{b},(0)} := \left\langle u_{m,\mathbf{k}}^{(0)}, u_{n,\mathbf{k}+\mathbf{b}}^{(0)} \right\rangle_P$$

is largely straightforward, except for the cases where $\mathbf{k} + \mathbf{b} \notin \mathcal{K}$. Since the $u_{m,\mathbf{k}}^{(0)}$ obey no easily exploitable symmetry (except for

$$u_{n,\mathbf{k}}^{(0)} = u_{n,-\mathbf{k}}^{(0)*} \quad (4.4)$$

in inversion-symmetric crystals) and no translational periodicity like the one the Bloch modes $\psi_{n,\mathbf{k}}$ obey, it is necessary to calculate $u_{n,\mathbf{k}}^{(0)}$ for \mathbf{k} -points outside the Brillouin zone to be able to calculate the $M^{\mathbf{k},\mathbf{b}}$ for all pairs \mathbf{k}, \mathbf{b} . Note that the \mathbf{k} -inversion symmetry of Equation (4.4) does not continue to hold for mixtures of the $u_{n,\mathbf{k}}^{(0)}$ since the mixing matrices $U^{\mathbf{k}}$ are not guaranteed to have any such symmetry. In contrast to that, the symmetry

$$M^{\mathbf{k},\mathbf{b}} = [M^{\mathbf{k}+\mathbf{b},-\mathbf{b}}]^H$$

continues to hold always and can be used here as well as for the updated inner product matrices of Section 4.8.3 to cut the cost of computing the $M^{\mathbf{k},\mathbf{b}}$ in half.

4.8.3 Updating the inner product matrices

In step 4 iii) above, given a new unitary mixing matrix $U^{\mathbf{k}} \in \mathbb{C}^{n,n}$ that mixes the initial periodic Bloch modes according to

$$u_{n,\mathbf{k}}^{(0)}(\mathbf{r}) \mapsto \sum_m U_{n,m}^{\mathbf{k}} u_{n,\mathbf{k}}^{(0)}(\mathbf{r}),$$

we would like to find out how this affects $M^{\mathbf{k},\mathbf{b}}$. Again, this is the transpose of the $U^{\mathbf{k}}$ used in [MV97]. We find

$$\begin{aligned} M_{n,m}^{\mathbf{k},\mathbf{b}} &\mapsto \left\langle \sum_{\nu} U_{n,\nu}^{\mathbf{k}+\mathbf{b}} u_{\nu,\mathbf{k}+\mathbf{b}}^{(0)}, \sum_{\mu} U_{m,\mu}^{\mathbf{k}} u_{\mu,\mathbf{k}}^{(0)} \right\rangle_P \\ &= \sum_{\nu,\mu} U_{n,\nu}^{\mathbf{k}+\mathbf{b}} U_{m,\mu}^{\mathbf{k}} * M_{\nu,\mu}^{\mathbf{k},\mathbf{b}} = [U^{\mathbf{k}+\mathbf{b}} M^{\mathbf{k},\mathbf{b}} (U^{\mathbf{k},\mathbf{b}})^H]_{n,m}. \end{aligned}$$

to be the appropriate update formula. The matrix exponential of W contained in the update formula for $U^{\mathbf{k}}$ itself can be easily computed through an eigendecomposition since $(iW)^H = iW$ is hermitian.

4.8.4 Other implementation notes

To avoid problems with the floating-point nature of a computer implementation of this algorithm, it may be wise to change the summation order in many of the terms involved in Ω^{Δ} and its gradient, such that

$$\sum_{\mathbf{k}} \sum_{\mathbf{b}} \dots \quad \text{becomes} \quad \sum_{\mathbf{b}} \sum_{\mathbf{k}} \dots .$$

More often than not, terms for opposing \mathbf{b} have opposite signs. In order to avoid cancellation and/or roundoff error, accumulating all the terms for each \mathbf{b} and then having one cancellation-prone addition per \mathbf{b} is usually preferable to the naïve way of calculating these sums.

Chapter 5

Implementation

This chapter describes the numerical software that I created for the purposes of this thesis. With a little more than 14,000 lines, the implementation has many aspects worthy of discussion. We will go through the software from the bottom up, meaning that we will take a look at the most basic layers first and then move up to higher levels of abstraction.

One of my fundamental beliefs with respect to scientific computation is that simplicity and correctness always wins over raw speed. While a pure C++ solution would probably have provided the fastest end result, the implementational burdens of C++, such as permanent type annotation or explicit memory management, would have eaten up resources that could be put to better use in a proof-of-concept implementation. So, I decided to write only the bottom layers in C++ and control the C++ core by means of a scripting language called Python. Python distinguishes itself by

- being fundamentally high-level,
- supporting a wide range of programming paradigms from object-oriented to functional,
- its excellent documentation,
- having implicit memory management,
- supporting and encouraging exact and convenient error reporting,
- having powerful built-in data structures such as sets, dictionaries and lists,
- and being easy to interface with C/C++.

Compared to a compiled language, Python runs more slowly by at least an order of magnitude. Many people confirm that the increase in programming speed more than outweighs the runtime performance. I expect this to become much less of a problem over time, given the success of projects like Jim Hugunin’s IronPython, Greg Ewing’s Pyrex, or Armin Rigo’s Psyco. Also, given the ease of executing C++ code from Python, moving an expensive inner loop into compiled code is usually easy. I could go on and on about Python’s beauty and utility for quite a while. Rather than do that, I would recommend that you read Tim Peters’ “The Zen of Python” [Pet]. It captures Python’s spirit in a few lines of prose.

The software written for this thesis consists of four parts, each of which we will describe in turn:

- PyLinear. This package provides sparse and dense matrices along with a range of algorithms on them.
- PyAngle provides a 2D mesh generator.
- FemPy is a finite element package.
- PyWannier puts all the above components to use to compute Bloch modes and maximally localized Wannier functions.

The first three packages are developed in a manner that allows them to be used easily outside of the scope of this thesis.

Before we begin our description, I would also like to mention “C++ Boost”, another project whose results were instrumental in the creation of the software used for this thesis. It goes back to the initiative of several members of the ISO C++ standards committee who were dissatisfied with the meager class library that is currently part of the C++ language standard. In an effort to change this, they created a process of peer review, testing and regular releases of new prospective parts of the standard library. From their large collection, the libraries Boost.Python and Boost.UBlas were used in this work. Boost.Python is a “wrapper creation library” that makes accessing C++ code from Python very easy. Boost.UBlas aims to be a more generic replacement of the Fortran BLAS. It includes code for various types of sparse matrices.

In order to make the results of my work more accessible, I tried to avoid tools which are not available under Free Software or open-source licenses. Due to the wealth of open software on the Internet today, achieving this goal was not terribly difficult. Fittingly, the results of my work will be released under an open-source license once this thesis is completed.

5.1 PyLinear

PyLinear is hardly the first linear algebra package ever written for Python. The most successful of such packages today are Numerical Python [ADHO01] and its successor, numarray. They are fast and mature, but they only offer dense matrices. Geus’s PySparse [Geu02] and its as yet unpublished successor code fix this deficiency by providing sparse matrices on top of NumPy. Initial versions of PySparse did not provide complex arithmetic, but I was able to integrate support during the first few weeks of this thesis. While I was still adding to PySparse, I came across the libraries Boost.UBlas and Boost.Python, both mentioned above. I realized that combining the two had the potential to yield a much more mature and comprehensive matrix package than I would have been able to write on the basis of Geus’s code. So, I changed my plan and in about a week’s time, I had a working prototype. Since then, PyLinear has grown quite a bit and more than fulfilled the expectations that I had of it.

For a better understanding of some of my design decisions, let me elaborate on using C++ to do linear algebra. Within the scientific computation community, C++ has so far failed to gain widespread acceptance, as it is commonly seen as complicated, slow and bloated. The reasons for the perceived slowness of C++ in matrix applications are many.

Imagine computing an expression like $D \leftarrow \alpha A + \beta B + \gamma C$ for matrices A, B, C . In a naïve matrix package, this requires the creation and destruction of five (invisible) temporary variables of matrix type, with a total of $5n^2$ unnecessary load-store cycles. It would be much faster and, theoretically, very easy to evaluate the expression in a componentwise fashion. Todd Veldhuizen pioneered a technique known as *expression templates* that allows for componentwise evaluation with no overhead in end-user notation. A template-type-based tree representation of the expression is created at compile time, and upon conversion or assignment to a matrix storage type, the actual computation is performed. At this point, the whole expression is already known, so that simplifications such as componentwise evaluation can be applied. Another source of slowness is the misguided use of runtime polymorphism. It is desirable to write matrix algorithms, e.g. LU decomposition, polymorphically, so that one function works for all matrix types. But if the implementor chooses to do this by having a matrix base class with virtual methods for things like element access, her code will likely be too slow for most practical purposes. Virtual method calls by themselves are already slower than static method calls since virtual method calls involve a vtable lookup. But this is not the main problem. Virtual method calls also cannot be inlined by the compiler, which is a major problem for inner-loop matters like element access. A more useful way to achieve the goal of writing most code just once is through the use of templates (a.k.a. compile-time polymorphism) and, more specifically, by the *Barton-and-Nackman trick*. This trick involves a declaration like the following:

```
class my_matrix : public matrix_base<my_matrix> { ... };
```

Note that the class is inheriting from a base class whose template parameter is the class that we are only just defining. Surprisingly, this is legal in C++, and it allows methods of the base class to resolve calls into the superclass at compile time, making them eligible for inlining. Veldhuizen’s article [Vel00] provides much more insight and many more suggestions on linear algebra in C++. Boost.UBlas uses both expression templates and the Barton-and-Nackman trick.

N	Numeric [s^{-1}]	PyLinear [s^{-1}]	UBlas [s^{-1}]	Peak MFlops [$10^6 s^{-1}$]
2	59050.645	23206.191	571426.200	4.571
4	53284.607	22565.564	330008.000	21.121
8	38481.501	19550.263	120976.800	61.940
16	17036.477	11792.990	25107.400	102.838
32	3586.073	3251.958	3861.200	126.523
64	524.826	449.573	499.000	137.579
100	156.300	137.100	140.000	156.300
200	6.921	6.582	6.800	55.368
300	1.956	1.837	1.682	52.812
400	0.722	0.704	0.642	46.208

Table 5.1. Some performance measurements of PyLinear vs. other codes. Each entry in the first three columns shows the number of matrices of size $N \times N$ multiplied per second. All matrices were dense and filled with random numbers. The last column extrapolates an MFlops value from the largest entry in the first three columns. These numbers were obtained on an Intel Pentium III running at 800 MHz clock speed.

Furthermore, C++ allows abstraction, such as using one code base for any scalar type, or any type of matrix. With a little bit of consideration, it can be made to run just as fast as Fortran. Template metaprogramming makes C++ a fully staged language, i.e. a language where a Turing-complete environment is available at compile time to “write programs that write programs”. Finally, it is a modern, general-purpose language that has just about penetrated every field except scientific computation. Despite some problems, C++ proved a useful tool in the implementation.

When designing PyLinear, I felt that out of the multitude of matrix types offered by UBlas, I should offer at least three: a dense one, one for fast sparse assembly and one for fast sparse computation. Geus’s package sported the same choice. The sparse “assembly” type is represented as a list of $[i, j, a_{i,j}]$, where insertion takes an amortized $O(1)$, and the sparse “computation” type is a conventional compressed row storage type. Each of these three types is provided in double precision real and double precision complex.

Unfortunately, while I was writing Python wrappers for these types, several difficulties arose. First, each of these is its own class, with no common base class, thanks to the Barton-and-Nackman trick. This means that an entire copy of the wrapper code needs to exist once per wrapped matrix type, increasing memory use quite a bit. Another consequence of this is that, even though offering all the matrix types in single precision as well as double is desirable, the price would be doubling both the code size and compilation time. Second, while two different data types are usable on opposite sides of many of the operators (e.g. “+”) in C++ thanks to clever expression template code (which implicitly performs the necessary type conversions without any copying), the same is not necessarily true in Python: Each type for which compatibility is desired must be explicitly mentioned in the wrapper code, again adding to the code size, and detracting from its readability. Some of this can be circumvented with an extra Python layer around the core to make up for the missing type conversions, but this yet again increases run time. Also, when called from Python, many of the advantages gained by the use of expression templates are lost again, since a new matrix is created and filled for each intermediate computation.

However, one clear advantage of PyLinear’s approach is that the same capable matrix types are available at both the C++ and Python levels, which allows the simple porting of slow loops. Overall performance of PyLinear can be characterized as acceptable, as illustrated by Table 5.1.

A few things can be said about these performance numbers:

- Numerical Python loses about half as much time in the transition to C as PyLinear. This is expected to improve with future versions of Boost.Python. Increased use of Python’s native method tables (such as `PyNumberMethods`) will probably bring PyLinear closer to NumPy’s speed.
- As soon as processor arithmetic (and not scripting overhead) becomes the limiting factor, C++ and both Python libraries perform comparably. ($N \geq 32$)
- Past this point, we can observe the system exhaust the 64 KB level 1 cache and become bound to the level 2 cache interface bandwidth. Another (steeper) drop begins at $N \geq 400$ where the 1 MB of L2 cache is also exhausted, and memory bandwidth finally becomes the defining factor. At

this point, cache-adapted tiling as practiced in modern implementations of the Fortran BLAS could greatly improve the speed.

- Unfortunately, I have no explanation why pure UBlas performs worse than the UBlas-based PyLinear for large matrices—the same library versions, the same compiler and the same compiler options were used for both.

On top of this basic linear algebra layer, PyLinear provides a fairly comprehensive set of more advanced matrix algorithms. The following things are available:

- Many LAPACK functions, such as SVD, eigenvalues of dense matrices and linear system solvers for various matrix types,
- UMFPACK for solving large sparse systems,
- ARPACK for large sparse eigenvalue problems,
- Preconditioned CG, Preconditioned BiCGSTAB, LU, Cholesky, Jacobi eigendecomposition and the Bunse-Gerstner codiagonalization algorithm ([BGBM93], [CS96]),
- and an assortment of other tools such as random matrix generation or condition estimation.

In order to avoid instantiating these algorithms for all matrix types, I reused Geus’s idea of an abstract “*matrix operator*”, basically an interface that reduces any matrix to its matrix-vector multiplication capability. Implementing this interface is then all that is required to run CG or the ARPACK eigensolver. This is a partial solution to the mentioned problem of code growth by multiple instantiation, as it allows to have one polymorphic piece of compiled code for any kind of linear map, even those where an explicit matrix is not even present. This abstraction includes every matrix type offered by PyLinear, preconditioners, sums, and compositions of other matrix operators. Even the iterative solvers and the UMFPACK wrapper implement this interface, effectively providing the capability to invert any matrix operator if numerically feasible.

Some of the bindings were very easy to write thanks to Kresimir Fresl’s numerical bindings library. This library allows easy access from within the UBlas framework to many of the more well-known linear algebra libraries. However, some libraries, such as ARPACK and a few LAPACK functions, had to be wrapped individually for PyLinear. Part of this work has already been submitted to Boost and accepted back into the framework, while other parts will be submitted after the completion of this thesis.

Since PyLinear was written as the foundation of an extensive software system, verifiable correctness was of utmost importance. A large number of automated unit tests ensures that every part of PyLinear works as expected. At this point in time, documentation is somewhere between sparse and non-existent. Since I mimicked the Numerical Python API for the most part, the lack of documentation may be less apparent. But in any case, writing documentation is a top priority of mine.

There are many directions in which the package could be extended and areas where further work and research could be done. Armin Rigo’s Psycho could be adapted to handle Pylinear as a special case, promising large speed benefits. Cache tiling or Geus’s idea of software prefetch could be implemented in UBlas, each probably yielding a substantial boost in speed. Symmetric matrices in UBlas clearly need some work before they can be recommended for practical use. And finally, lazy evaluation could be implemented as a Python equivalent to expression templates.

5.2 PyAngle

The next component in our implementation stack is the mesh generator. It is a thin wrapper around Jonathan R. Shewchuk’s C code Triangle [She96], which won the 2003 J. H. Wilkinson prize for numerical software. Triangle bills itself as a high-quality mesh generator and refiner, with certain guaranteed mesh properties that allow for well-conditioned finite element discretization. Unlike previous wrapping efforts, such as John C. Mollis’s and Alexander Pletzer’s Ellipt2D (which can be found on Sourceforge), PyAngle carries Triangle’s full feature set over to the Python level. A convenience layer in Python makes the code easy to use. Just like for PyLinear, a two-tiered approach was used. Triangle was first wrapped in a layer of C++, which was then exported to the scripting level using Boost.Python.

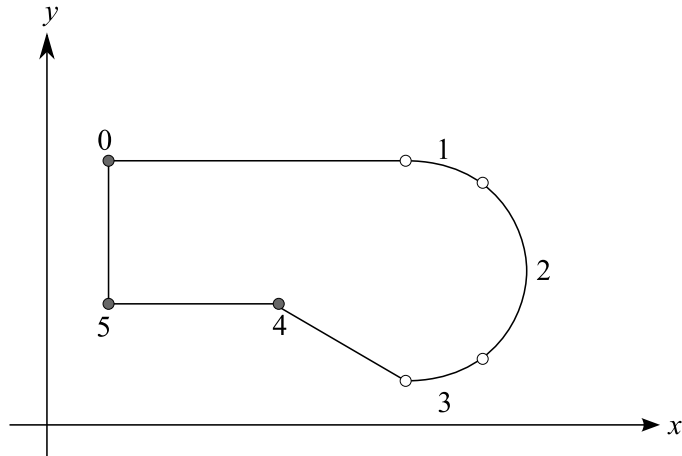


Figure 5.1. Representation of a `tShapeSection` in FemPy.

5.3 FemPy

FemPy is a one- and two-dimensional finite element package capable of solving the Poisson equation as well as the Laplace eigenvalue problem. It has sophisticated mesh generation capabilities and an easy-to-use interface.

5.3.1 The user interface

First, let us explore what FemPy looks like from the user's perspective, in the form of step-by-step instructions for its use.

1. The user generates a number of `tShapeSections`, each of which is a description of a closed curve. A shape section is represented as a list of points and `tShapeGuides`. Let's take a look at the example domain in Figure 5.1. Our shape section is represented as a list of six elements, as numbered in the Figure. Indices 0, 4 and 5 are simple `PyLinear` vectors, containing point coordinates for the corresponding vertices. At indices 1 to 3, we have `tShapeGuide` objects. A shape guide represents a piece of a curve that can be described as either $y = f(x)$ or $x = f(y)$. So, for example, for the shape guide at index 1

$$y = c_y + \sqrt{r^2 - (x - c_x)^2} =: f(x),$$

where c_x and c_y are the coordinates of the center of the full circle and r is its radius. It is usually a good idea to split shape guides at points where $f' \gg 1$; this is why we have segmented the half-circle. More precisely, to initialize a `tShapeGuide` object, you need:

- The deformation coordinate; this is the number of the dependent coordinate. Integer, 0 for x and 1 for y .
- An interval in the independent coordinate. A tuple of float values. For the shape guide labelled 1 in the figure, for example, this would be the x -coordinates of its adjacent vertices.
- The expression of the guiding function f . The independent coordinate needs to be a variable named `t`. (More on expressions can be found later in this section.)
- Two flags: whether to render the final point and whether to use exact elements to approximate the boundary.

Note that the vertices with non-filled dots in the figure are automatically generated by the shape guides and should *not* be part of the representing list. To avoid duplication of the end points of shape guide 1 and 2, their `render - final - point` flag should be set to `False`.

2. This list of shape sections is fed to a `tTwoDimensionalMesh` object in order to generate a triangular mesh. The shape sections may freely intersect. The meshed area will always be the union of

their interiors, and the boundaries of the shape sections are guaranteed to be maintained as inner boundaries in the generated mesh. An optional list of hole starting points allows you to have the mesh generator “eat” holes into the mesh. It will eat up every triangle that is not separated from the hole starting point by a shape section. At this point, you may also specify the desired element order (linear or quadratic, i.e. 1 or 2), and a refinement function that controls the fineness of the discretization locally.

3. The `tTwoDimensionalMesh` as an implementation of the abstract `tMesh` interface gives you access to a `tDOFManager` (a.k.a. “Degree Of Freedom Manager”, much like a glorified list of `tNodes`) and a list of `tFiniteElements`. When generating the `tShapeSections`, you are free to choose something called a tracking identifier, and after the mesh is generated, you may use this tracking identifier to query the `tDOFManager` for all nodes that are associated with a shape section with this identifier. For example, you may set the identifier “`dirichlet`” on certain shape sections, and later find the nodes on all shape sections marked thus, for example to enforce Dirichlet boundary conditions on them.
4. Next, the user constructs a constraint map. A constraint map is a Python dictionary whose keys are nodes and whose values are of the structure

```
offset, [(coefficient_1, node_1), ..., (coefficient_n, node_n)]
```

i.e. a tuple which contains a scalar and a list of scalar-and-`tNode` tuples. If we let $f(\mathbf{node})$ be the value of the solution at `node`, then this constraint expresses

$$f(\mathbf{node}) \stackrel{!}{=} \text{offset} + \sum_{j=1}^n \text{coefficient}_j f(\mathbf{node}_j).$$

5. This constraint map together with the mesh from step 3 is finally fed to a solver (such as `solvePoisson()` or `tLaplacianEigenproblemSolver`), which will yield its result in the form of `tMeshFunctions`, essentially “functions defined on the mesh”. Instances of this class can be called like regular functions for pointwise evaluation, they support vector space operations and they can compute their own gradient if asked to as well.
6. Finally, the computed solution can be visualized using the `fempy.visualization` module. VTK (for use with programs like Paraview or MayaVi), Matlab and Gnuplot are supported as visualization targets.

This completes our walkthrough of a typical use of FemPy. While FemPy is generally at the proof-of-concept stage of development, I feel that its design is already quite mature. To provide maximum flexibility to its users, FemPy was written to provide “*mechanisms, not policies*”. I strived to provide a set of clean, modularly-separate single-purpose functional blocks that the user can employ in whatever fashion she likes, without being obliged to use all the other parts of the package. This is substantially different from a *framework*, where all the parts are interwoven and mutually dependent.

Next, let us take a look at what is happening behind the scenes.

5.3.2 FemPy’s inner workings

FemPy’s internals are also built in several interdependent layers. Like on the large scale, let us explore them from the bottom up.

- At the core, there is a small symbolic computation module called `fempy.expression` that can perform symbolic differentiation, substitution and elementary term simplification tasks. Whenever the term “*expression*” is mentioned in this section, this means the data type used by this module. This data type is loosely patterned after LISP and consists of nested tuples. For example,

```
(eo.POWER, (eo.PLUS, (eo.POWER, (eo.VARIABLE, 'x'), 2),
                    (eo.POWER, (eo.VARIABLE, 'y'), 2))),
0.5)
```

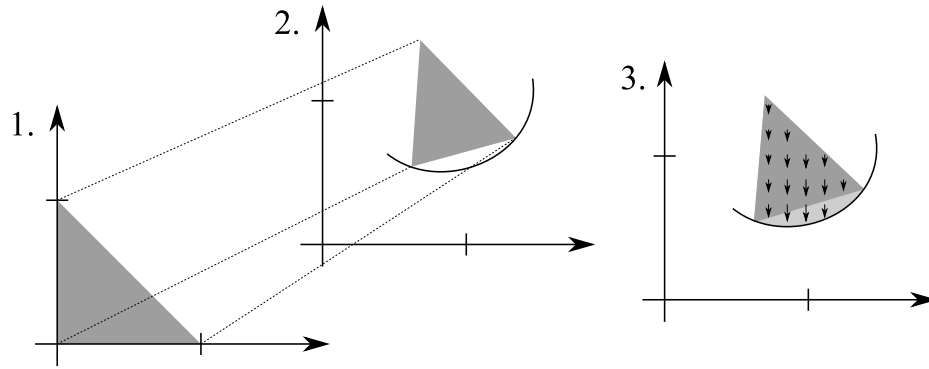


Figure 5.2. How exact boundaries work in FemPy. Starting from the unit triangle (1.), a linear transform is applied (2.) to line up the vertices, and finally the interior is stretched (3.) to match the curved edge.

embodies the expression $\sqrt{x^2 + y^2}$ if the module `femPy.expression_operators` has been imported as “eo”. Thanks to Python’s dynamic nature, these expressions can be compiled into native Python functions at runtime, yielding usable evaluation speed.

- Next are `tFormFunctionKits`. A form function kit contains FEM ansatz functions for an elementary shape, such as a triangle. These form functions are represented as expressions. FemPy provides linear and quadratic functions on triangles and linear functions on lines.
- As mentioned in the usage description, a full FEM discretization (represented as a `tMesh`) consists of `tFiniteElements` which store the geometric details of each triangle in the mesh. A `tFormFunctionKit` then plugs into each `tFiniteElement` to provide the FEM ansatz functions on this geometry. As long as the untransformed shapes of the finite element and the form function kit match (i.e. unit triangle or unit interval, respectively), any form function kit can be used with any type of element. FemPy provides lines (1D), triangles and deformed triangles as geometric shapes. It could be said that the ansatz function code is *orthogonal* in features to the geometric deformation code.

Deformed triangles are FemPy’s way to produce an exact discretization of curved boundaries. Let us explore how they work, using Figure 5.2 as a guide. We begin with the unit triangle in step 1. Next, we calculate a linear transform to line up the vertices, but do not yet worry about the curved boundary. Now consider the *shape control function* φ . φ is defined on the unit triangle as

$$\varphi(x, y) := \frac{-4xy}{(x - y)^2 - 1}.$$

A graph of φ is shown in Figure 5.3. Observe that on two of the triangle edges $\varphi(\mathbf{x}) = 0$, while on the third one $\varphi(\mathbf{x}) = 1$, with differentiable transitions between those values. φ (understandably) has a singularity in two of the triangle’s vertices. As the last step (3.), the necessary amount of deformation along the boundary is calculated and propagated throughout the triangle in the direction of deformation given by the relevant shape section. This amount of deformation is then scaled by the value of φ in the corresponding point of the unit triangle, yielding the final transform.

The same deformation procedure could easily be applied in an analogous fashion to more than one edge of the triangle, but this is currently unimplemented.

- FemPy does not use precalculated element matrices; nearly everything is derived from first principles within the source code. For example, integrals are computed by means of a 7-point Gaussian quadrature on the unit triangle.

Note that to simply *use* FemPy as described above, no knowledge about these internals is necessary.

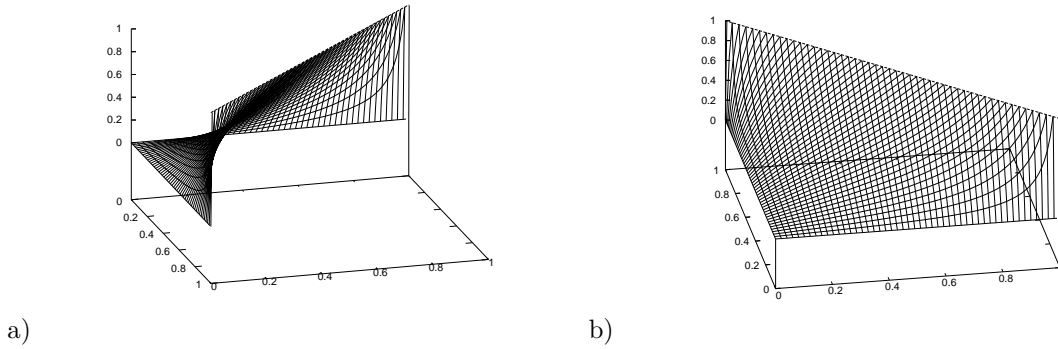


Figure 5.3. Two views of the shape control function φ .

5.3.3 Numerical experiments

In order to inspire some confidence in FemPy's methods, we will explore a few problems with known solutions and see how FemPy fares solving them. We will be considering a well-known boundary value problem and an eigenvalue problem with piecewise-constant coefficients, providing some insight in the analytical solution and detailed convergence data for each.

The Laplace equation on an annulus

Consider the boundary value problem

$$\begin{aligned} -\nabla^2 u(\mathbf{r}) &= 0, \\ u(\mathbf{r}) &= \alpha_1 \quad \text{for } r = \rho_1, \\ u(\mathbf{r}) &= \alpha_2 \quad \text{for } r = \rho_2 \end{aligned}$$

on an open annulus-shaped domain Ω with inner and outer radii $\rho_1 < \rho_2$. The solution to this problem is unique, explicitly known and easily verified:

$$u(\mathbf{r}) = \frac{\alpha_2 - \alpha_1}{\log(\rho_2/\rho_1)} \log(r/\rho_1) + \alpha_1.$$

Solving this problem in FemPy using second-order elements, adaptive mesh generation and exact boundary approximation, we obtain an empirical order of convergence of around 2 in the energy norm and of around 3.2 in the L^2 norm. We refer to Figure 5.4 for details.

An eigenvalue problem with piecewise constant coefficients

As our next test for FemPy's numerics, consider the eigenvalue problem

$$\begin{aligned} -\nabla^2 u(\mathbf{r}) &= \lambda \alpha(\mathbf{r}) u(\mathbf{r}), \\ u(\mathbf{r}) &= 0 \quad \text{for } r = 1 \end{aligned}$$

on the circle with Dirichlet boundary conditions. Our function $\alpha(\mathbf{r})$ shall be piecewise-constant:

$$\alpha(\mathbf{r}) := \begin{cases} \alpha_1 & \text{for } 0 \leq r < 1/2, \\ \alpha_2 & \text{for } 1/2 \leq r \leq 1. \end{cases}$$

This problem arises, for example, in the solution of the wave equation on a circular, inhomogeneous drumhead. We present it here as an indicator of FemPy's performance in solving eigenvalue problems with piecewise constant coefficients, such as the TM problem as covered in Section 6.4. The above problem was designed to be analytically solvable while bearing enough semblance to our final target equation, for which no analytic solutions are known.

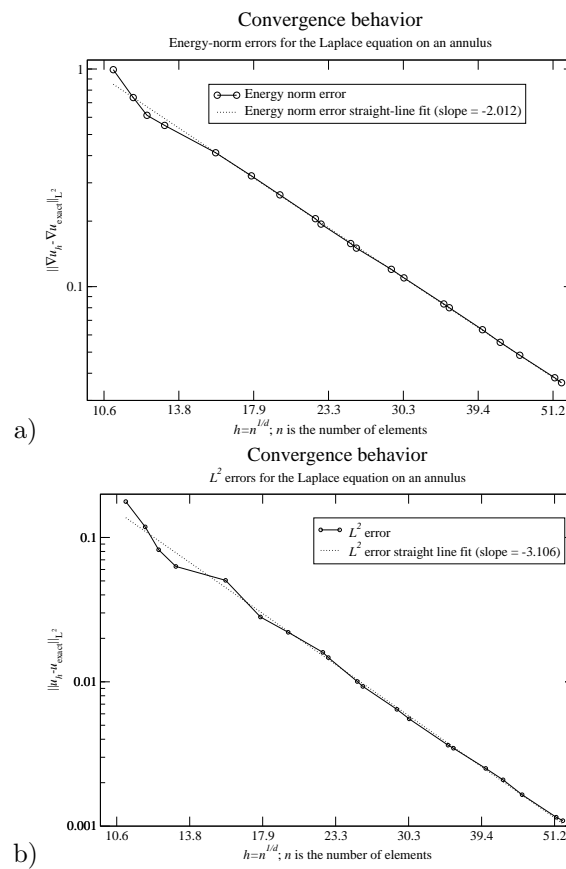


Figure 5.4. Convergence data for the solution of the Laplace equation a) in the energy norm, and b) in the L^2 norm. While it may not be immediately apparent, the X axis in both plots is actually logarithmic.

First, we will briefly present an approach to the analytic solution. Separating the variables in polar coordinates such that $u(\mathbf{r}) = R(r)\Theta(\theta)$, we obtain

$$-\lambda\alpha(r) = \frac{R''(r)}{R(r)} + \frac{R'(r)}{rR(r)} - \frac{\Theta''(\theta)}{r^2\Theta(\theta)}.$$

Obviously, $\Theta''(\theta)/\Theta(\theta) =: -n^2$ must be a constant, and considering $\Theta(\theta + 2\pi) = \Theta(\theta)$, we have

$$\Theta(\theta) = A_n \cos(n\theta) + B_n \sin(n\theta),$$

and we know that $n \in \mathbb{N}_0$. Moving on, we obtain an ordinary differential equation for R :

$$0 = R''(r) + \frac{R'(r)}{r} + \left(\lambda\alpha(r) - \frac{n^2}{r^2} \right) R(r).$$

We will solve this equation separately on $\Omega_1 := [0, 1/2)$ and $\Omega_2 := [1/2, 1]$. Under the change of scale $\rho_i := \sqrt{\lambda\alpha_i}r$ on Ω_i for $i = 1, 2$, R becomes $\bar{R}_i(\rho_i) := R(\rho_i/\sqrt{\lambda\alpha_i}) = R(r)$, and our differential equation becomes the Bessel differential equation on both domains:

$$0 = \bar{R}_i''(\rho_i) + \frac{1}{\rho_i}\bar{R}_i'(\rho_i) + \left(1 - \frac{n^2}{\rho_i^2} \right) \bar{R}_i(\rho_i).$$

We get the following boundary conditions:

$$\begin{aligned} \bar{R}_1(0) &= \text{finite}, \\ \bar{R}_1(\sqrt{\lambda\alpha_1}/2) &= \bar{R}_2(\sqrt{\lambda\alpha_2}/2), \\ \bar{R}_1'(\sqrt{\lambda\alpha_1}/2)\sqrt{\lambda\alpha_1} &= \bar{R}_2'(\sqrt{\lambda\alpha_2}/2)\sqrt{\lambda\alpha_2} \\ \bar{R}_2(\sqrt{\lambda\alpha_2}) &= 0. \end{aligned}$$

The Bessel equation of order n is solved by the n th-order Bessel functions $J_n(\cdot)$ and $Y_n(\cdot)$. Therefore, we consider the ansatz

$$\begin{aligned} \bar{R}_1(\rho) &:= c_{1,1}J_n(\rho) + c_{1,2}Y_n(\rho), \\ \bar{R}_2(\rho) &:= c_{2,1}J_n(\rho) + c_{2,2}Y_n(\rho). \end{aligned}$$

The finiteness condition at 0 dictates $c_{1,2} = 0$. Otherwise, we are left with a nonlinear system of equations, which we will solve numerically. Note that there are three equations and three unknowns ($\mu := c_{2,1}/c_{1,1}$, $\nu := c_{2,2}/c_{1,1}$ and λ) because the Bessel equation is linear. Dividing by $c_{1,1}$ is fine if we exclude the trivial solution $\bar{R}_i = 0$. Many hints in writing this discussion were taken from the introductory book [Str92].

Let us turn to FemPy's performance in this instance. As above, we use second-order elements and exact domain representation. As an extra twist, the jump discontinuity in α is traced by the mesh exactly. The computation yields an empirical order of convergence for the eigenvalues of 3.53, and the L^2 error in the eigenfunctions exhibits an EOC of about 2.778. Both values are roughly in the expected range. Figure 5.5 has the details. For simplicity, we limited our convergence studies to the lowest-energy eigenpair where always $\Theta(\theta) = 1$. The computation of the analytic solutions and the solution of the nonlinear system of equations was based on the excellent SciPy package by Travis Oliphant, Konrad Hinsen et al.

This is also an excellent opportunity to back up the claim that FemPy is easy to use, even for problems of moderate difficulty. Including mesh generation and visualization, the solution code for our problem spans a mere 31 easily-readable lines:

```
import pylinear.matrix_tools as mtools
import fempy.mesh
import fempy.geometry as geometry
import fempy.solver as solver
import fempy.visualization as visualization

def needsRefinement( vert_origin, vert_destination, vert_apex, area ):
```

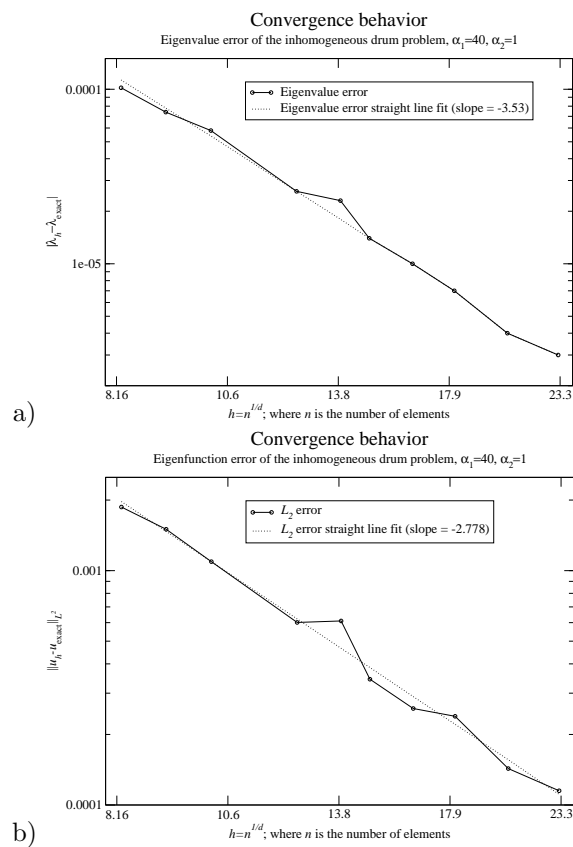


Figure 5.5. Convergence data for the solution of the Laplace eigenvalue problem a) for the lowest eigenvalue, and b) for the lowest eigenfunction in the L^2 norm. While it may not be immediately apparent, the X axis in both plots is actually logarithmic.

```

    return area >= 3e-2

alpha_1 = 40; alpha_2 = 1

def alpha(x):
    if mtools.norm2(x) < 0.5:
        return alpha_1
    else:
        return alpha_2

mesh = fempy.mesh.tTwoDimensionalMesh(
    [fempy.mesh.tShapeSection(fempy.geometry.getCircle(1), "dirichlet"),
     fempy.mesh.tShapeSection(fempy.geometry.getCircle(0.5), "unconstrained")],
    refinement_func = needsRefinement)
constraints = solver.getDirichletConstraints(mesh, u_d = lambda x: 0)
eigensolver = solver.tLaplacianEigenproblemSolver(
    mesh, constrained_nodes = constraints, g = alpha)
eigensolver.setupConstraints(constraints)
solutions = eigensolver.solve(0, tolerance = 1e-10, number_of_eigenvalues = 20)

for evalue, emode in solutions:
    visualization.visualize("vtk", ("",result.vtk", "",result_grid.vtk"), emode.real)
    if raw_input("[enter for next, q for quit]") == "q":
        break

```

This program is included in the FemPy source package as `example/bessel - piecewise.py`.

5.3.4 Further work

There are several areas where FemPy could greatly benefit from additional work. Probably the most urgent need is better documentation. Adaptivity has already been taken into account in FemPy's design, however, these routines currently lack a usable error estimator, short of comparing to a known solution. It would not be hard at all to allow the package to deal with more general elliptic PDEs. Given the infrastructure, adding initial support for 3D elements would also not take too long. Finally, major speedups could be achieved by moving common element kernels into C++.

5.4 PyWannier

This last package of our software stack is the application of all the generic modules described before to the specific problem of this thesis. It is responsible for all the functionality that cannot be generalized enough to warrant its inclusion in FemPy.

5.4.1 Weak formulation of the eigenproblem

As a first step, PyWannier needs to solve the eigenvalue problem

$$-\nabla^2 \psi(\mathbf{r}) = \frac{\omega^2}{c^2} \varepsilon(\mathbf{r}) \psi(\mathbf{r}) \quad (5.1)$$

on the domain P with Floquet boundary conditions

$$\psi(\mathbf{r} + \mathbf{R}) = e^{i\mathbf{k} \cdot \mathbf{R}} \psi(\mathbf{r}), \quad \nabla \psi(\mathbf{r} + \mathbf{R}) \cdot \mathbf{n} = e^{i\mathbf{k} \cdot \mathbf{R}} \nabla \psi(\mathbf{r}) \cdot \mathbf{n}$$

for $\mathbf{r} \in \partial P$ and $\mathbf{R} \in L$ and a unit-length vector \mathbf{n} normal to ∂P in \mathbf{r} . Projected onto a trial function $\varphi \in L^2_\varepsilon(P)$, the above eigenproblem becomes

$$-\int_P \nabla^2 \psi(\mathbf{r}) \varphi^*(\mathbf{r}) d\mathbf{r} = \frac{\omega^2}{c^2} \int_P \varepsilon(\mathbf{r}) \psi(\mathbf{r}) \varphi^*(\mathbf{r}) d\mathbf{r},$$

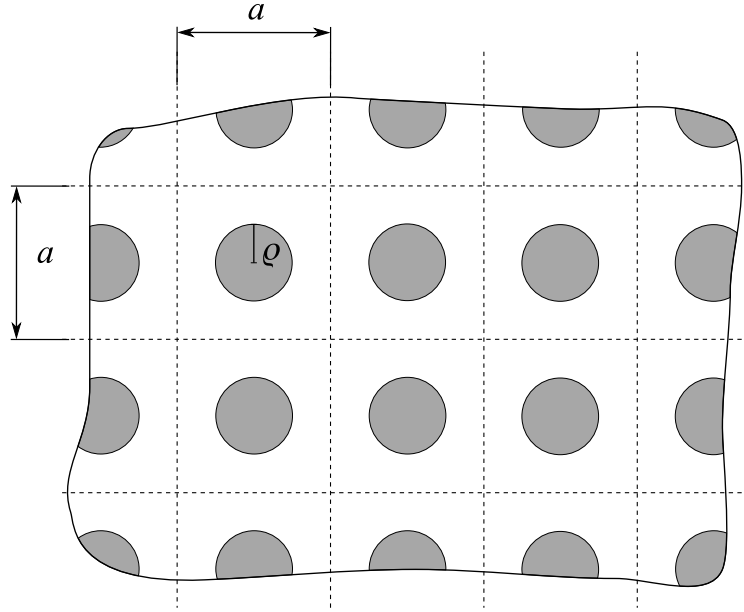


Figure 5.6. A common rod-type permittivity layout for a 2D photonic crystal. White is air or vacuum ($\varepsilon = 1$), shaded is some dielectric medium with $\varepsilon > 1$, for example glass.

or equivalently, by Green’s First Identity

$$\int_P \nabla \psi(\mathbf{r}) \cdot \nabla \varphi^*(\mathbf{r}) d\mathbf{r} - \int_{\partial P} \nabla \psi(\mathbf{r}) \cdot \mathbf{n} \varphi^*(\mathbf{r}) dS = \frac{\omega^2}{c^2} \int_P \varepsilon(\mathbf{r}) \psi(\mathbf{r}) \varphi^*(\mathbf{r}) d\mathbf{r}. \quad (5.2)$$

The center term is

$$\begin{aligned} \int_{\partial P} \frac{\partial \psi(\mathbf{r})}{\partial n} \varphi^*(\mathbf{r}) dS &= \sum_{\mathbf{R} \in L / \{-1, 1\}} \int_{\partial P(\mathbf{R})} \nabla \psi(\mathbf{r}) \cdot \mathbf{n} \varphi^*(\mathbf{r}) + \nabla \psi(\mathbf{r} + \mathbf{R}) \cdot (-\mathbf{n}) \varphi^*(\mathbf{r} + \mathbf{R}) dS \\ &= \sum_{\mathbf{R} \in L / \{-1, 1\}} \int_{\partial P(\mathbf{R})} \nabla \psi(\mathbf{r}) \cdot \mathbf{n} \varphi^*(\mathbf{r}) - e^{i\mathbf{k} \cdot \mathbf{R}} \nabla \psi(\mathbf{r}) \cdot \mathbf{n} e^{-i\mathbf{k} \cdot \mathbf{R}} \varphi^*(\mathbf{r}) dS \\ &= 0. \end{aligned}$$

So, we have rewritten our eigenvalue problem in the weak (or variational) form

$$\int_P \nabla \psi(\mathbf{r}) \cdot \nabla \varphi^*(\mathbf{r}) d\mathbf{r} = \frac{\omega^2}{c^2} \int_P \varepsilon(\mathbf{r}) \psi(\mathbf{r}) \varphi^*(\mathbf{r}) d\mathbf{r} \quad (5.3)$$

There is one slightly subtle point worth noting here: If we solve Equation (5.3) instead of Equation (5.1), we only need to enforce the first (non-derivative) part of the Floquet boundary conditions. The second (derivative) part follows automatically, since the center term of Equation (5.2) is forced to zero.

5.4.2 Discretization of the eigenproblem

Using FemPy, we create an appropriate finite element mesh, taking care to adapt the mesh to the characteristic appearance of the given permittivity function ε . For a common “rod-type” layout as illustrated in Figure 5.6, the following requirements should be placed on a good photonic unit cell mesh:

- For many photonic crystals, ε has jump discontinuities. Those discontinuities should be traced by element boundaries as accurately as possible to avoid loss of precision. Exact elements are a good way to achieve this.
- Element size should vary with location. While a very fine mesh can easily cause a prohibitive amount of computation, a coarse one may not yield accurate results. Thus, it is advisable to vary mesh

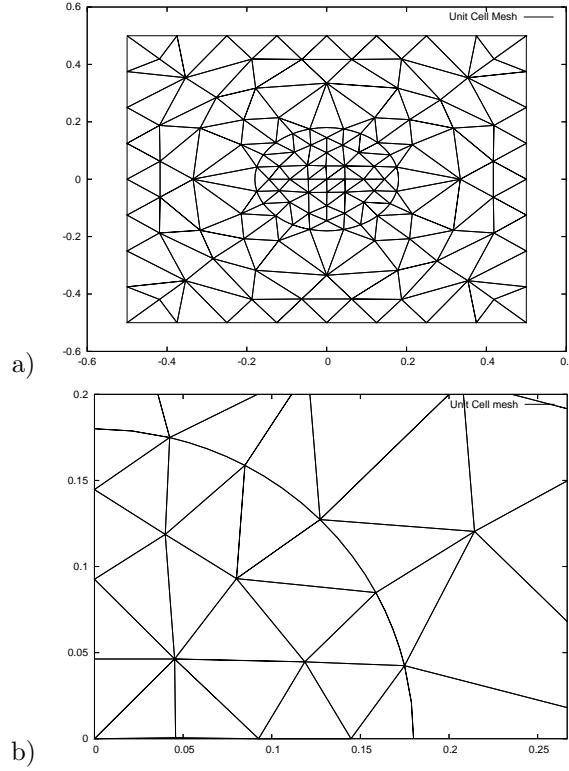


Figure 5.7. a) shows the mesh generated by FemPy for one unit cell of the photonic crystal shown in Figure 5.6. b) is a magnified detail of a), illustrating the use of exact elements.

density over the unit cell. Since high-permittivity regions act as centers of excitation, especially fine meshes are required within and around them.

- The use of higher order elements is highly advisable.

FemPy performs well on each of these points, as can be seen from the resulting example meshes of Figure 5.7. Actually, provisions for mesh adaptivity are already made in the code, but not fully used for lack of time to implement an appropriate error estimator.

As a result of mesh generation, we have a number of finite element node functions which make up our (naturally \mathbf{k} -dependent) finite element space $V_{h,\mathbf{k}} := \{\varphi_{\nu,\mathbf{k}}(\mathbf{r}) : \nu = 1, \dots, N\}$. Each $\varphi_{\nu,\mathbf{k}} \in V_{h,\mathbf{k}}$ must satisfy the first Floquet condition

$$\varphi_{\nu,\mathbf{k}}(\mathbf{r} + \mathbf{R}) = e^{i\mathbf{k} \cdot \mathbf{R}} \varphi_{\nu,\mathbf{k}}(\mathbf{r})$$

for $\mathbf{r} \in \partial P(\mathbf{R})$ and $\mathbf{R} \in L/\{-1, 1\}$. Naturally, this condition does not affect node functions which are zero along the boundary, or, rather, it only affects functions for which $\partial P \cap \text{supp}(\varphi_{\nu,\mathbf{k}}) \neq \emptyset$. Now, suppose we expand

$$\psi_{\mathbf{k}}(\mathbf{r}) \approx \sum_{\mu=1}^N c_{\mu,\mathbf{k}} \varphi_{\mu,\mathbf{k}}(\mathbf{r}).$$

We obtain ($\nu = 1, \dots, N$)

$$\sum_{\mu=1}^N c_{\mu,\mathbf{k}} \int_P \nabla \varphi_{\mu,\mathbf{k}}(\mathbf{r}) \cdot \nabla \varphi_{\nu,\mathbf{k}}^*(\mathbf{r}) d\mathbf{r} = \frac{\omega^2}{c^2} \sum_{\mu=1}^N c_{\mu,\mathbf{k}} \int_P \varepsilon(\mathbf{r}) \varphi_{\mu,\mathbf{k}}(\mathbf{r}) \varphi_{\nu,\mathbf{k}}^*(\mathbf{r}) d\mathbf{r}.$$

Further, using

$$\begin{aligned} S_{\mu,\nu}^{\mathbf{k}} &:= \int_P \nabla \varphi_{\mu,\mathbf{k}}(\mathbf{r}) \cdot \nabla \varphi_{\nu,\mathbf{k}}^*(\mathbf{r}) d\mathbf{r}, \\ M_{\mu,\nu}^{\mathbf{k}} &:= \int_P \varepsilon(\mathbf{r}) \varphi_{\mu,\mathbf{k}}(\mathbf{r}) \varphi_{\nu,\mathbf{k}}^*(\mathbf{r}) d\mathbf{r}, \end{aligned}$$

and $\mathbf{c}_{\mathbf{k}} := [c_{\mu,\mathbf{k}}]_{\mu=1}^N$, the eigenvalue problem becomes

$$\mathbf{c}_{\mathbf{k}}^T S^{\mathbf{k}} = \frac{\omega^2}{c^2} \mathbf{c}_{\mathbf{k}}^T M^{\mathbf{k}} \Leftrightarrow S^{\mathbf{k}} \mathbf{c}_{\mathbf{k}}^* = \frac{\omega^2}{c^2} M^{\mathbf{k}} \mathbf{c}_{\mathbf{k}}^*.$$

Let us now take a moment to understand how the Floquet conditions are enforced. We know that our trial functions $\varphi_{\nu,\mathbf{k}}$ need to satisfy $\varphi_{\nu,\mathbf{k}}(\mathbf{r} + \mathbf{R}) = e^{i\mathbf{k} \cdot \mathbf{R}} \varphi_{\nu,\mathbf{k}}(\mathbf{r})$. However, when generating the mesh, FemPy knows nothing about this requirement. We obtain a mesh with independent coefficients for both halves of what is later to become a ‘‘coupled node’’. Let this mesh be called $\mathcal{M} = \{\mathbf{n}_{\mu}\}_{\mu=1}^M$, its node functions be $\bar{\varphi}_{\mu}$ and let a corresponding coefficient vector be known as $\bar{\mathbf{c}} = [\bar{c}_{\mu}]_{\mu=1}^M \in \mathbb{C}^M$.

For each lattice basis vector $\mathbf{R} \in L/\{-1, 1\}$ and each boundary node $\mathbf{n}_{\nu} \in \partial P(\mathbf{R})$, we find another node in the finite element mesh \mathcal{M} such that $\mathbf{n}_{o(\nu,\mathbf{R})} = \mathbf{n}_{\nu} + \mathbf{R}$. We obtain a set of nodes whose values $\bar{c}_{o(\nu,\mathbf{R})}$ are determined by the equation $\bar{c}_{\nu} = e^{i\mathbf{k} \cdot \mathbf{R}} \bar{c}_{o(\nu,\mathbf{R})}$ for a given \mathbf{k} . Let’s call these the ‘‘dependent nodes’’ and the remainder the ‘‘independent nodes’’ and make sure that the independent nodes are numbered $1, \dots, N$, while the dependent ones are numbered $N + 1, \dots, M$, by renumbering if necessary. Then for each $\mathbf{n}_{\nu} \in \partial P(\mathbf{R})$ we set

$$\varphi_{\nu,\mathbf{k}} := \bar{\varphi}_{\nu} + e^{i\mathbf{k} \cdot \mathbf{R}} \bar{\varphi}_{o(\nu,\mathbf{R})}.$$

If we consider the formal vectors $\varphi_{\mathbf{k}} := [\varphi_{\nu,\mathbf{k}}]_{\nu=1}^N$ and $\bar{\varphi} := [\bar{\varphi}_{\mu,\mathbf{k}}]_{\mu=1}^M$, then we can see them connected by a matrix $A \in \mathbb{C}^{N \times M}$ as

$$\varphi = \underbrace{\begin{pmatrix} 1 & & * & \cdots & * \\ & \ddots & \vdots & & \vdots \\ & & 1 & * & \cdots & * \end{pmatrix}}_{=A} \bar{\varphi},$$

where $[A^{\mathbf{k}}]_{\nu,o(\nu,\mathbf{R})} = e^{i\mathbf{k} \cdot \mathbf{R}}$ for $\mathbf{n}_{\nu} \in \partial P(\mathbf{R})$. In general, there are no more than four entries per row in A , so it remains sparse despite the impression to the contrary given by the formula above. Given the matrices

$$\begin{aligned} \bar{S}_{\mu,\nu} &:= \int_P \nabla \bar{\varphi}_{\mu}(\mathbf{r}) \cdot \nabla \bar{\varphi}_{\nu}^*(\mathbf{r}) d\mathbf{r}, \\ \bar{M}_{\mu,\nu} &:= \int_P \varepsilon(\mathbf{r}) \bar{\varphi}_{\mu}(\mathbf{r}) \bar{\varphi}_{\nu}^*(\mathbf{r}) d\mathbf{r}, \end{aligned}$$

(which are supplied by FemPy) it is a matter of simple calculation to see that

$$\begin{aligned} [M^{\mathbf{k}}]_{\mu,\nu} &= \langle \varphi_{\mu,\mathbf{k}}, \varphi_{\nu,\mathbf{k}} \rangle_P = \left\langle \sum_{\mu'} [A^{\mathbf{k}}]_{\mu,\mu'} \bar{\varphi}_{\mu'}, \sum_{\nu'} [A^{\mathbf{k}}]_{\nu,\nu'} \bar{\varphi}_{\nu'} \right\rangle_P \\ &= \sum_{\mu',\nu'} [A^{\mathbf{k}}]_{\mu,\mu'} [\bar{M}]_{\mu',\nu'} [A^{\mathbf{k}}]_{\nu,\nu'}^* = [A^{\mathbf{k}} \bar{M} [A^{\mathbf{k}}]^H]_{\mu,\nu} \end{aligned}$$

and analogously $S^{\mathbf{k}} = A^{\mathbf{k}} \bar{S} [A^{\mathbf{k}}]^H$. Using the matrices $S^{\mathbf{k}}$ and $M^{\mathbf{k}}$, the above eigenproblem is solved by ARPACK [LSY97] applied as part of a shift-invert strategy, using a plain-vanilla, unpreconditioned CG for the ‘‘invert’’ part. Given an eigenpair $(\lambda, \mathbf{c}_{\mathbf{k}})$, we may expand $\mathbf{c}_{\mathbf{k}}$ to $\bar{\mathbf{c}}$ by $\bar{\mathbf{c}} = [A^{\mathbf{k}}]^T \mathbf{c}_{\mathbf{k}}$. As a side note, the naïve expectation that $(\lambda, \bar{\mathbf{c}})$ is an eigenvalue of the ‘full’ problem $\bar{S} \bar{\mathbf{c}}^* = \bar{M} \bar{\mathbf{c}}^*$ is unfounded, because for $\nu \geq N + 1$

$$\sum_{\mu=1}^N c_{\mu} \int_P \nabla \varphi_{\mu}(\mathbf{r}) \cdot \nabla \varphi_{\nu}^*(\mathbf{r}) d\mathbf{r} = \frac{\omega^2}{c^2} \sum_{\mu=1}^N c_{\mu} \langle \varphi_{\mu}, \varphi_{\nu} \rangle_P$$

does not imply both

$$\begin{aligned} \sum_{\mu=1}^N c_{\mu} \int_P \nabla \varphi_{\mu}(\mathbf{r}) \cdot \nabla \bar{\varphi}_{\nu}^*(\mathbf{r}) d\mathbf{r} &= \frac{\omega^2}{c^2} \sum_{\mu=1}^N c_{\mu} \langle \varphi_{\mu}, \bar{\varphi}_{\nu} \rangle_P, \\ \sum_{\mu=1}^N c_{\mu} \int_P \nabla \varphi_{\mu}(\mathbf{r}) \cdot \nabla \bar{\varphi}_{o(\nu, \mathbf{R})}^*(\mathbf{r}) d\mathbf{r} &= \frac{\omega^2}{c^2} \sum_{\mu=1}^N c_{\mu} \langle \varphi_{\mu}, \bar{\varphi}_{o(\nu, \mathbf{R})} \rangle_P. \end{aligned}$$

Sometimes, however, especially in cases where the finite element mesh generation cannot be tightly controlled, finding a proper node that satisfies $\mathbf{n}_{o(\nu, \mathbf{R})} = \mathbf{n}_{\nu} + \mathbf{R}$ to a desired precision may turn out to be difficult. As a last-ditch recovery possibility for this case, the present code will try to find two adjacent nodes such that $\alpha(\nu, \mathbf{R})\mathbf{n}_{o_1(\nu, \mathbf{R})} + (1 - \alpha(\nu, \mathbf{R}))\mathbf{n}_{o_2(\nu, \mathbf{R})} = \mathbf{n}_{\nu} + \mathbf{R}$, i.e. the desired location is a convex combination of two real nodes. The corresponding constraint equation is $\bar{c}_{\nu} = e^{i\mathbf{k} \cdot \mathbf{R}}[\alpha(\nu, \mathbf{R})\bar{c}_{o_1(\nu, \mathbf{R})} + (1 - \alpha(\nu, \mathbf{R}))\bar{c}_{o_2(\nu, \mathbf{R})}]$. Note that in constructing the previous equation you have the freedom to choose whether to interpolate first and then apply the Floquet condition (as done here) or the other way around; ostensibly, none of the two alternatives is better than the other. For our case, the conditions for assembly of $A^{\mathbf{k}}$ are $[A^{\mathbf{k}}]_{\nu, o_1(\nu, \mathbf{R})} = \alpha(\nu, \mathbf{R})e^{i\mathbf{k} \cdot \mathbf{R}}$ and $[A^{\mathbf{k}}]_{\nu, o_2(\nu, \mathbf{R})} = (1 - \alpha(\nu, \mathbf{R}))e^{i\mathbf{k} \cdot \mathbf{R}}$. The implementation also allows higher-order approximations and cases where $\mathbf{n}_{\nu} + \mathbf{R}$ cannot be expressed as a convex combination of two nodes; this functionality was included at the last minute. However, it seems that the use of interpolation has an adverse effect on convergence, as noted in Section 6.3.

Axmann and Kuchment [AK99] suggest a different finite element scheme for the computation of Bloch modes. Yet another one was suggested by Dobson [Dob99].

5.4.3 The way to maximally localized Wannier functions

After computing between 10 and 20 eigenpairs for each $\mathbf{k} \in \mathcal{K}$, the Bloch modes and their corresponding eigenvalues are written out to disk using Python’s “pickle” serialization facility. It is notable that `pickle` manages to serialize (i.e. convert to a stream of octets) complex data structures like `tMesh` and `tMeshFunction` along with all the data they refer to almost without help, which makes storing a fairly complex finite element discretization very straightforward. The driver code that does everything described up to this point can be found in the PyWannier source tree as `src/compute_eigenmodes.py`.

A few postprocessing steps for the Bloch modes are collected in the script file `src/postprocess_eigenmodes.py`. This program computes the periodic Bloch modes $u_{n, \mathbf{k}}$ from the calculated Bloch modes $\psi_{n, \mathbf{k}}$. After that, all eigenfunctions of each type (periodic and not) are multiplied by a factor so that their imaginary part at a specific point is forced to zero. This point is chosen for the functions to have maximum possible summed absolute value at this point over all the n and \mathbf{k} . Next, we make sure that all

$$\langle u_{n, \mathbf{k}}, u_{n, \mathbf{k}} \rangle_P = 1 = \langle \psi_{n, \mathbf{k}}, \psi_{n, \mathbf{k}} \rangle_P \quad \text{for all } n, \mathbf{k}.$$

As a final step, bands are separated and degeneracies are found. Band separation consists of the task of looking at a fairly coarse sampling of the dispersion relation and deciding, at each $\mathbf{k} \in \mathcal{K}$, which eigenvalues of this and of a neighboring \mathbf{k} -point are considered to be in the same band. Given the coarse sampling of the dispersion relation and the possibility for band crossings and degeneracies, this job is surprisingly nontrivial. It is a fairly complex two-dimensional “path” following problem that would warrant some research in its own right. (The term “sheet following” seems more appropriate.) Since this was not the main focus of this thesis, we were satisfied with a more simplistic approach.

While far from perfect, the following method has proven to yield consistently acceptable results: We start by ordering the eigenvalues by magnitude at each \mathbf{k} . As a preliminary guess, all the eigenvalues with the index 0 are considered to be in the first band, all with index 1 in the second, and so on. Given this starting guess, a refinement procedure is run. For each band index and each $\mathbf{k} \in \mathcal{K}$, we linearly extrapolate a guess of the current eigenvalue from two neighbors in each direction. The eigenvalue that lies closest to these guesses in a least-squares sense is chosen as a member of the band. One could wish for a band finding scheme which considers a more global view of the dispersion relation than ours. However, as illustrated in Section 4.8.1, this choice of bands fortunately does not influence the performance of the Wannier localization. Its main importance lies in allowing proper band-wise visualization and generating the opportunity for a few plausibility checks.

After the results of postprocessing are once again written to disk, the script file `src/localize_wanniers.py` takes over and controls the last step of the process. It is a fairly literal implementation of the method described in Chapter 4. A number of visualization and utility scripts complement the software.

5.4.4 Future work

As we shall see in Section 6.5, the Wannier functions obtained through this software still have a few flaws. Fixing this is probably the most urgent need at this point. Of course, it would be a worthwhile extension to actually use the Wannier functions for the eigenmode calculations that are their original purpose. Generalizing the software to treat more shapes of primitive unit cell than just simple cubic should also prove to be not too hard. The extension to three-dimensional structures is probably not as straightforward, but it would more than likely give an opportunity to pursue a set of interesting research questions.

Chapter 6

Results

In this chapter, we will examine a compilation of results obtained both analytically and numerically using the methods described in this thesis. We will also discuss issues surrounding convergence speed and numerical stability where appropriate. Before we begin, let us note that TM-type equations do not have a preference for a certain scale: Any solution may be scaled isotropically (along with the PDE's coefficient function ε) and is still a solution afterwards. So, without loss of generality, we will study the unit interval and unit square as primitive unit cells in one and two dimensions, respectively. Similarly, we will demand that, for simplicity, the speed of light $c = 1/\sqrt{\varepsilon_0\mu_0}$ is 1; the only way in which we would notice a different value is by a constant factor in ω , which is clearly insignificant.

6.1 Constant permittivity in one dimension

The simplest possible case of our general problem can be found in one dimension with constant relative permittivity ε . Without loss of generality, we will assume ε to be 1. Then the eigenvalue problem simplifies to

$$\begin{aligned} -\psi''_{n,k}(r) &= \omega^2\psi_{n,k}(r), \\ \psi_{n,k}(1) &= e^{ik}\psi_{n,k}(0), \\ \psi'_{n,k}(1) &= e^{ik}\psi'_{n,k}(0). \end{aligned}$$

An complete set of solutions can be found analytically: the functions $\psi_{n,k}(r) := e^{i(k+2\pi\nu(n))r}$ solve the system, where $k \in [0, 2\pi)$, $r \in [0, 1]$ and $\nu(n) : \mathbb{N}_0 \rightarrow \mathbb{Z}$ is a bijective map. The eigenvalues are $(k + 2\pi\nu(n))^2$. Figure 6.1 shows the dispersion relation and a few eigenfunctions for this simple problem. Note that in the figure we chose the *line style* of the bands according to their $\nu(n)$ index, to show their correspondence to a given function $\psi_{n,k}$. In contrast to that, bands are usually *numbered* from the bottom up in order of increasing magnitude. So, the “bottom” band, which we will assign the number 0, consists of the left half of the line labelled “ $\nu = 0$ ” and the right half of the line labelled “ $\nu = 1$ ”. Consequently, we have a degeneracy at $k = \pi$. More generally, there are degeneracies at each $k = \pi m$ for $m \in \{0, 1, 2\}$.

Our main interest in this elementary case lies in the differences which arise when switching to a more general ε , which is what we will be doing next.

6.2 The non-constant case in one dimension

Generalizing the results of Section 6.1, we now allow an r -dependent relative permittivity ε . As an example, we will use the piecewise constant function

$$\varepsilon_\alpha(r) := \begin{cases} 1 & \text{for } 0 \leq r < 0.4, \\ \alpha & \text{for } 0.4 \leq r < 0.6, \\ 1 & \text{for } 0.6 \leq r \leq 1 \end{cases}$$

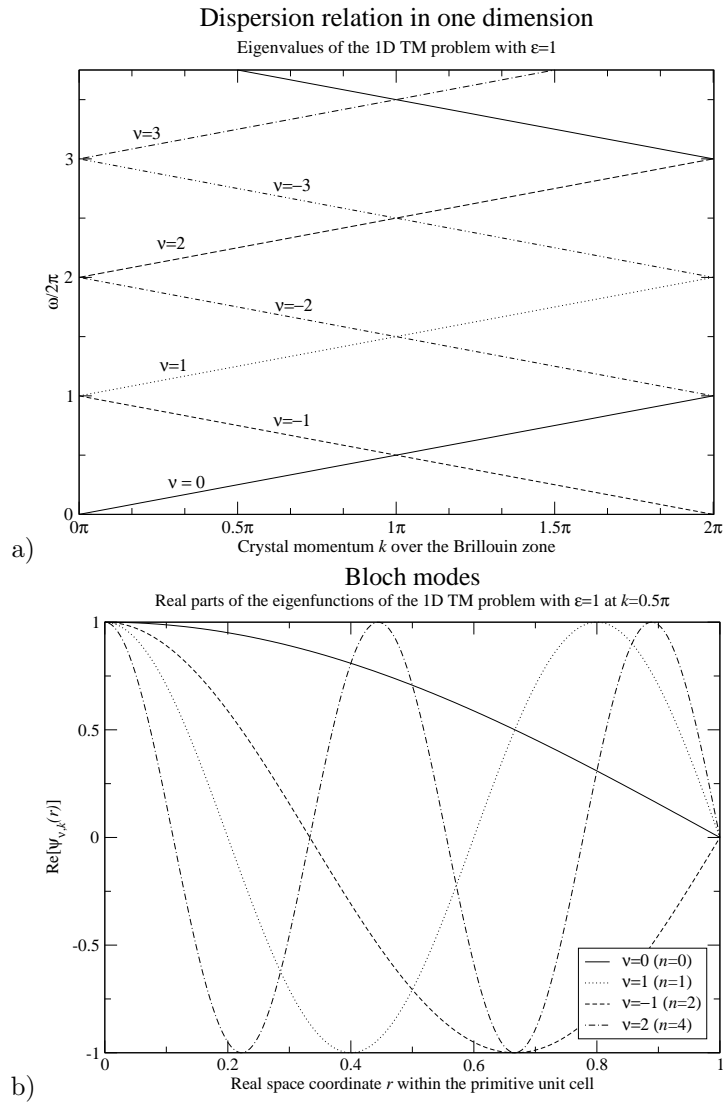


Figure 6.1. Eigenvalues and eigenfunctions of the 1D problem with $\varepsilon = 1$. The annotations “ $(n = \cdot)$ ” in b) reflect the number of the band in the dispersion relation ordered by magnitude. They establish a link to Figure 6.2b).

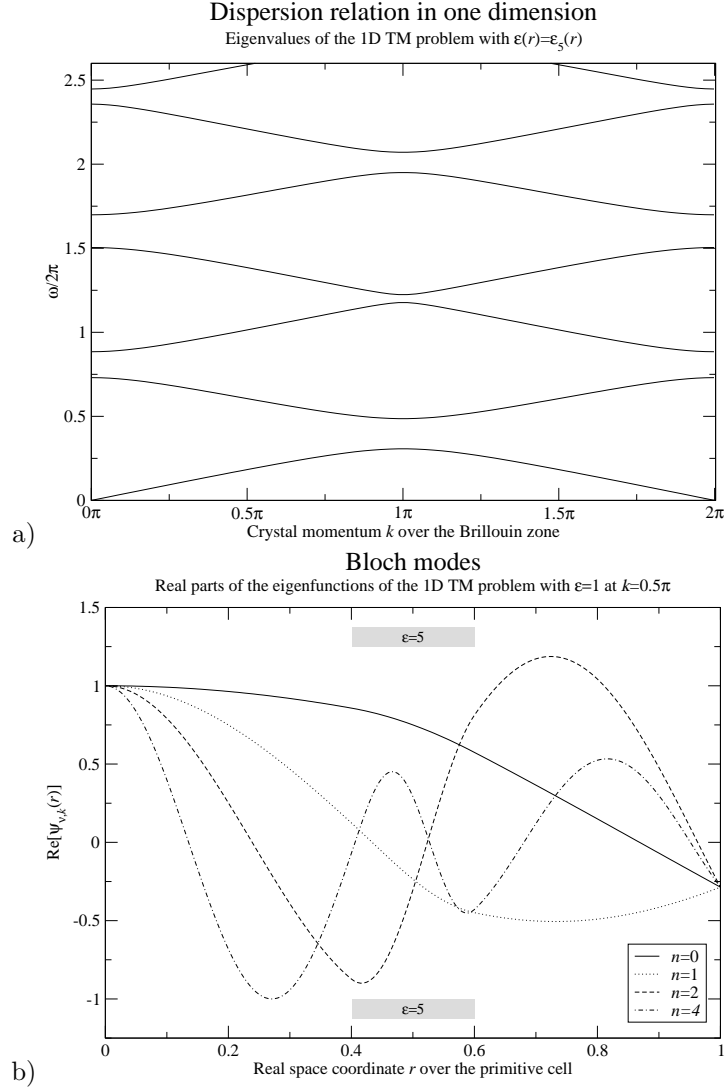


Figure 6.2. Eigenvalues and eigenfunctions of the 1D problem with piecewise-constant permittivity $\varepsilon(r)$. The highlighted area in b) shows where a higher permittivity was applied.

for values of $\alpha > 0$. For completeness' sake, the eigenproblem is now

$$\begin{aligned} -\psi''_{n,k}(r) &= \omega^2 \varepsilon_\alpha(r) \psi_{n,k}(r), \\ \psi_{n,k}(1) &= e^{ik} \psi_{n,k}(0), \\ \psi'_{n,k}(1) &= e^{ik} \psi'_{n,k}(0). \end{aligned}$$

In this case, finding an analytic solution would be quite tedious, if not impossible. Numerical results are fairly easy to obtain, however. Ours were generated by FemPy using linear one-dimensional elements.

Comparing Figures 6.1 and 6.2, we observe the following:

- Even for the smallest perturbation of ε , the degeneracies at $k = n\pi$ split. This seems to happen as the bands try to “smooth themselves out” and fix the non-differentiability at the points $k = n\pi$. Consequently, a band gap opens where each pair of crossed-over bands has split its degeneracy.
- The fact that the title “lowest-energy band” changes hands halfway through the Brillouin zone at $k = \pi$ can also be observed at the eigenfunction level. Figure 6.3 shows a plot of $u_{0,k}(r)$ over k and r for two values of α . For the α near 1, there is still a massive discontinuity at $k = \pi$, almost a “switch” between the two different eigenfunctions, as the harsh transition in Figure 6.1a)

- a) Image u-development-sharp.eps is excluded here for size reasons.
 b) Image u-development-smooth.eps is excluded here for size reasons.

Figure 6.3. The development of $u_{0,k}$ over $k \in B$ and r for a) $\alpha = 1.1$ and b) $\alpha = 5$. Note the smoothing of the discontinuity at $k = \pi$ as α increases.

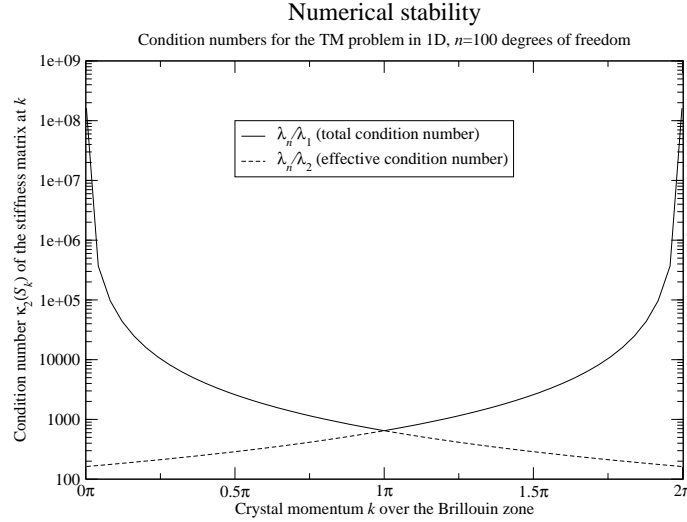


Figure 6.4. An impression of FEM conditioning over the Brillouin zone.

suggests. As the band smoothes out (Figure 6.2a)), so does the development of the eigenfunction, as can be seen in Figure 6.3b). Be aware that picking $u_{0,k}$ over $\psi_{0,k}$ for this visualization makes the effect seem more severe than it really is: The transition of the $\psi_{0,k}$ is continuous, but still not differentiable.

- For $\alpha > 1$, the dispersion relation is “compressed”, all bands move down towards lower energy levels. For $\alpha < 1$, the reverse happens.
- Consider the speed of propagation in a medium, elementarily expressed as $c_{\text{med}} = 1/\sqrt{\epsilon\mu} = \lambda f$ with wavelength λ and frequency f . As ϵ increases, the wavelength must decrease. This is exactly what we observe in Figure 6.2b) in the section where ϵ is higher.
- The bands assume an uneven spacing, which seems to reverse itself for $\alpha \mapsto 1/\alpha$. The origin of this phenomenon is as yet unclear to me.

Empirically, these observations hold for any value of $\alpha > 0$ in a qualitative manner. As $\alpha \rightarrow 0$ or $\alpha \rightarrow \infty$, the upper bands flatten out more and more.

Numerically, the one-dimensional problem is just as unfriendly as the two-dimensional one, as we will see. For $k = n\pi$, the system’s stiffness matrix becomes exactly singular, and even in neighborhoods of these points the finite element matrices are fairly ill-conditioned. Figure 6.4 shows the condition number of the stiffness matrix as a function of k . In two dimensions, these problems are somewhat less grave, but still not negligible.

Many of the phenomena observed in the one-dimensional case carry over analogously to two dimensions, even though they are often harder to detect and visualize. For example, instructive graphs like Figure 6.3 become either impossible or very cumbersome to make. As such, the problem in one dimension is a good indicator of what to expect in two.

6.3 Constant permittivity in two dimensions

As in one dimension, we will begin by examining a constant- ε version of the eigenproblem:

$$\begin{aligned} -\nabla^2 \psi_{n,\mathbf{k}}(\mathbf{r}) &= \omega^2 \psi_{n,\mathbf{k}}(\mathbf{r}), \\ \psi_{n,\mathbf{k}}(\mathbf{r} + \mathbf{R}) &= e^{i\mathbf{k}\cdot\mathbf{R}} \psi_{n,\mathbf{k}}(\mathbf{r}), \\ \nabla \psi_{n,\mathbf{k}}(\mathbf{r} + \mathbf{R}) \cdot \mathbf{n} &= e^{i\mathbf{k}\cdot\mathbf{R}} \nabla \psi_{n,\mathbf{k}}(\mathbf{r}) \cdot \mathbf{n} \end{aligned}$$

for $\mathbf{r} \in \partial P(\mathbf{R})$ for any $\mathbf{R} \in L$ and a unit-length vector \mathbf{n} normal to ∂P in \mathbf{r} .

Formally, the notable differences between this and the simpler one-dimensional case are that \mathbf{k} and \mathbf{r} are now vector-valued, that the second derivative is replaced by the Laplace operator, and that the boundary conditions now take slightly more effort to write down. Fortunately, finding analytic solutions of this problem is just as easy as in one dimension: The functions $\psi_{n,\mathbf{k}}(\mathbf{r}) := e^{i(\mathbf{k}+2\pi\boldsymbol{\nu}(n))\cdot\mathbf{r}}$ form a complete set of solutions for $\mathbf{r} \in P$, $\mathbf{k} \in B$ and $\boldsymbol{\nu} : \mathbb{N}_0 \rightarrow \mathbb{Z}^2$ an enumeration of all two-integer tuples. The eigenvalue corresponding to $\psi_{n,\mathbf{k}}$ is $|\mathbf{k} + 2\pi\boldsymbol{\nu}(n)|^2$. Figure 6.5, which you might recognize from the title page, illustrates the resulting dispersion relation. But wait, you may say, the dispersion relation of a two-dimensional structure should be three-dimensional! Since three-dimensional visualizations are often a bit unwieldy and tend to obscure more than they show, we have resorted to simply fixing a path around a patch of the Brillouin zone (shown in Figure 6.5b)) and plotting the eigenvalues over its parametrization. Note that the chosen patch is irreducible by symmetry, i.e. no smaller patch can be mapped to all of the Brillouin zone solely by rotations and inversions. This simplification is common in solid state literature and is based on the assumption that observing values on the boundary of the patch is sufficient to be able to guess/interpolate the eigenvalues in its interior. In order to give you a feeling for what is *really* happening, we have included a true three-dimensional view of the a few bands, cf. Figure 6.6.

Turning to the actual *subjects* of these images, we can see that the band structure is *highly* degenerate, much more so than in one dimension. In fact, in Figure 6.6 I could only show 3 out of the bottom seven bands without cluttering the image too much.

The data up to this point were fairly easy to obtain: every eigenpair is explicitly known. This gives us the opportunity to assess the quality of our numerics, by comparing computed results to their exact counterparts. What kinds of imprecision do we have to expect? How much deviation from the true solution do we experience, both in eigenvalues and eigenfunctions? What is the empirical order of convergence of the eigenvalues and the eigenfunctions? Let us answer these questions.

First, let us study the deviations in the dispersion relation qualitatively. Figure 6.7 shows both the computed and the exact version overlaid. Generally, the approximation is quite good. Especially in the numerically troublesome high-symmetry points Γ , M and X, a worse approximation was to be expected. In these points, as in the analogous points in one dimension, the bands “try to” smooth themselves out. This can be observed, for example, in a slight upwards tendency of the lowest band at Γ . Also, it seems that some degeneracies are numerically extremely unstable; they seem to split very easily.

On a more quantitative note, a numerical study of approximation behavior yielded an empirical order of convergence for the eigenvalues of 3.8, for the eigenfunctions in the L^2 norm of 3.24, and for the eigenfunctions in the energy norm of 1.97. These values are well within the expected range for exact-boundary, second-order FEM approximations. Figure 6.8 shows the complete convergence history for these three measures. Let us understand how these numbers were computed. Basically, the values shown in the figure represent the quantity

$$\sqrt{\frac{1}{(\#\gamma)N} \sum_{\mathbf{k} \in \gamma} \sum_{n=1}^N X_{n,\mathbf{k}}}$$

for each (quadratic) error measure $X_{n,\mathbf{k}}$. N is the number of bands considered in the computation, five in our case. γ is a tighter variant of the path shown in Figure 6.5b), slightly scaled away from the troublesome points Γ , X and M, with 5 points on each leg. For degenerate eigenvalues, the eigenvalue errors are calculated as usual, but the eigenfunction errors are omitted for simplicity (in this case, the count $\#\gamma N$ is adjusted accordingly). These numerical results were generated using FemPy as described in Section 5.4.2.

Besides the above quantities which can only be computed if the real solution is explicitly known, there is another measure of numerical quality that can be computed even without a reference solution. It relies

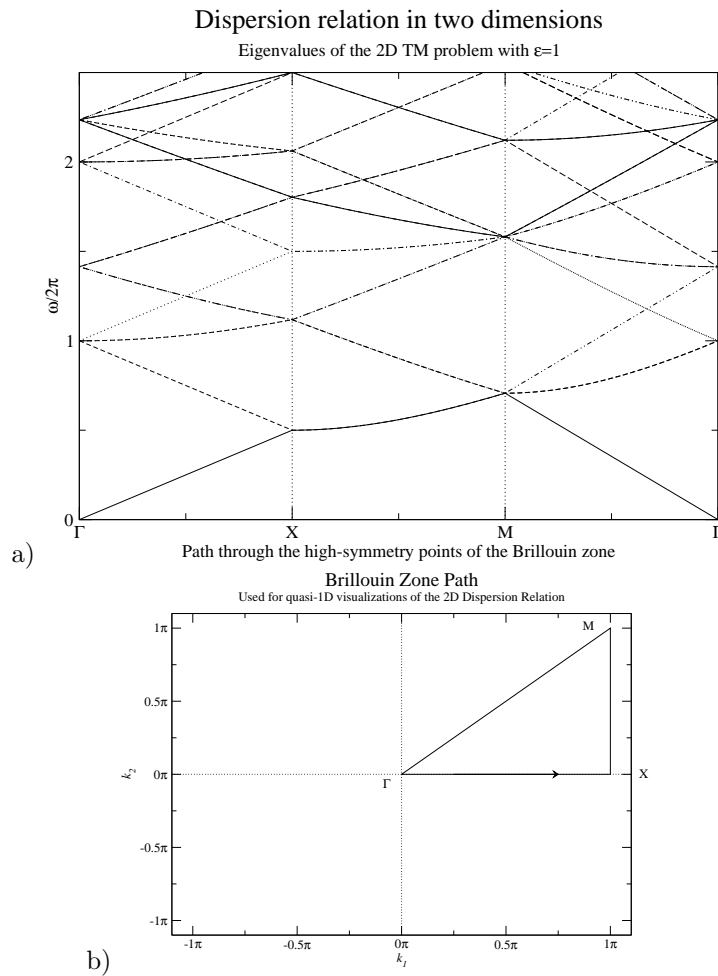


Figure 6.5. a) shows the dispersion relation of homogeneous ($\epsilon \equiv 1$) space sampled along the Brillouin zone path shown in b).

Image e1-dispersion-relation-3d.eps is excluded here for size reasons.

Figure 6.6. A true 3D view of the dispersion relation of homogeneous ($\epsilon \equiv 1$) space. Specifically, bands 0, 2 and 6 are shown. The label “w/2pi” should read “ $\omega/2\pi$ ”, which could not be produced in the visualizer.

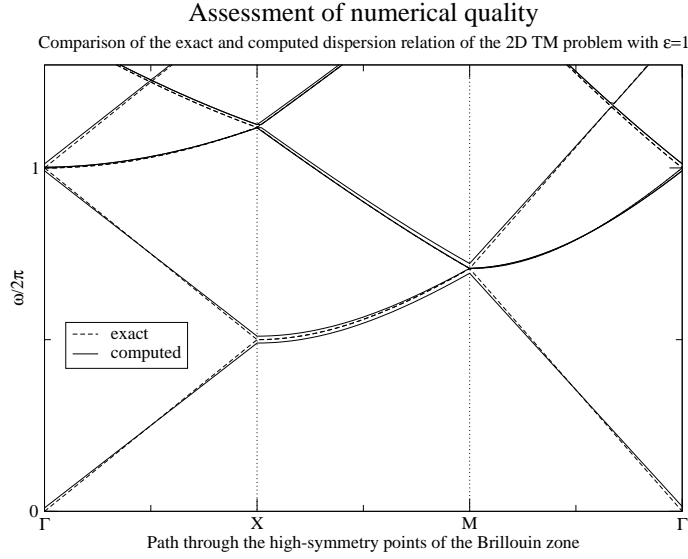


Figure 6.7. A comparison of the exact and the computed version of Figure 6.5.

on the fact that the derivative part of the Floquet boundary conditions is automatically enforced by the variational formulation of the eigenvalue problem, as outlined in Section 5.4.1. The error term

$$B := \frac{1}{(\#\mathcal{K})N} \sum_{\mathbf{k} \in \mathcal{K}} \sum_{n=1}^N \int_{\partial P(\mathbf{R})} |(\nabla \psi_{n,\mathbf{k}}(\mathbf{r} + \mathbf{R}) - e^{i\mathbf{k} \cdot \mathbf{R}} \nabla \psi_{n,\mathbf{k}}(\mathbf{r})) \cdot \mathbf{n}|^2 d\mathbf{r},$$

which we will call the *boundary derivative error*, should also obey a certain order of convergence, around two for second-order elements. The vector \mathbf{n} in the definition of B represents a unit-length boundary normal vector of P in \mathbf{r} . Figure 6.9 has some convergence data for B in the constant-permittivity case. It seems that the use of boundary approximations as described in Section 5.4.2 more or less destroys any convergence of B , for as yet unknown reasons. This drawback is not captured in Figure 6.9.

Fortunately, the fact that B does not require a fixed solution to compare to means that it can just as well be applied in the case of non-constant permittivity, which we will be considering next.

6.4 The non-constant case in two dimensions

First, let us define a permittivity function analog to the one above, as

$$\varepsilon_{\rho,\alpha}(\mathbf{r}) := \begin{cases} 1 & \text{for } r > \rho \\ \alpha & \text{for } r \leq \rho \end{cases}.$$

Figure 5.6 illustrates the meaning of ρ , and in that figure, α would be the permittivity of the shaded areas. For completeness, the eigenvalue problem for this section is

$$\begin{aligned} -\nabla^2 \psi_{n,\mathbf{k}}(\mathbf{r}) &= \omega^2 \varepsilon_{\rho,\alpha}(\mathbf{r}) \psi_{n,\mathbf{k}}(\mathbf{r}), \\ \psi_{n,\mathbf{k}}(\mathbf{r} + \mathbf{R}) &= e^{i\mathbf{k} \cdot \mathbf{R}} \psi_{n,\mathbf{k}}(\mathbf{r}), \\ \nabla \psi_{n,\mathbf{k}}(\mathbf{r} + \mathbf{R}) \cdot \mathbf{n} &= e^{i\mathbf{k} \cdot \mathbf{R}} \nabla \psi_{n,\mathbf{k}}(\mathbf{r}) \cdot \mathbf{n} \end{aligned}$$

for $\mathbf{r} \in \partial P(\mathbf{R})$ for any $\mathbf{R} \in L$ and a unit-length vector \mathbf{n} normal to ∂P in \mathbf{r} .

Unfortunately, just like in one dimension, finding any eigenvalues or eigenfunctions explicitly is probably a hopeless endeavor for $\alpha \neq 1$, so we will concentrate on numerical calculations, which are quite feasible. A numerically-calculated dispersion relation of a crystal of the type described here is shown in Figure 6.10. Since we do not have authoritative sources for comparison, there is not much that we can say in the way of hard evidence about the quality of our finite-element approximations of the Bloch functions in this case. Fortunately, we can appeal to a number of “softer” arguments which make it seem reasonable that our results reflect the truth as much as possible:

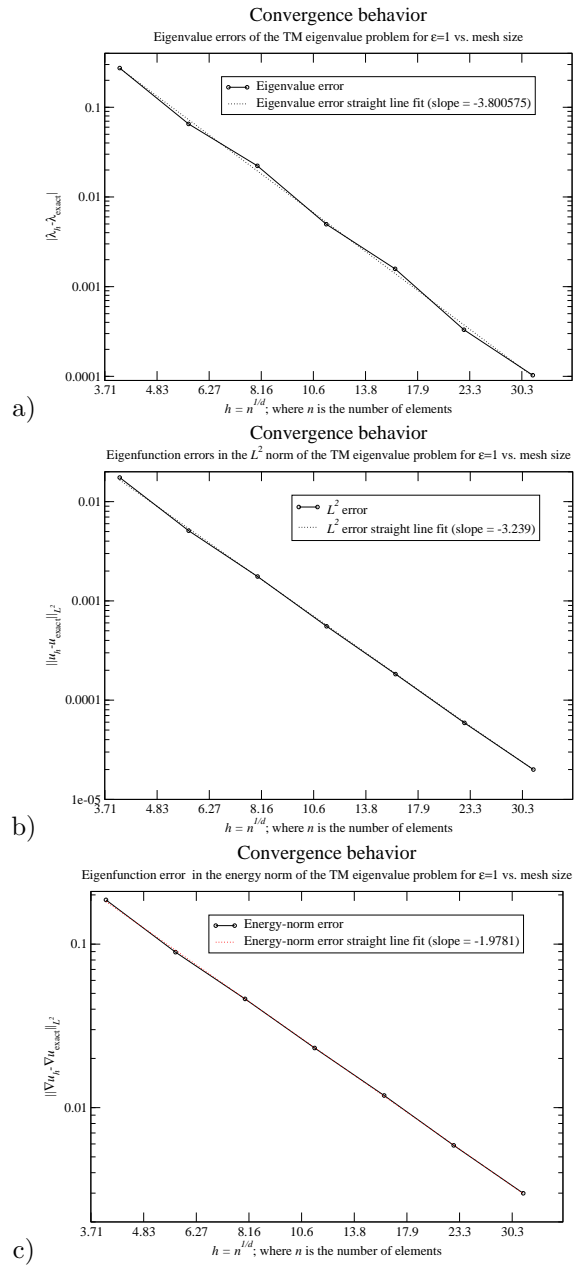


Figure 6.8. A detailed convergence study of the 2D FEM approximation to the dispersion relation and the Bloch functions in the case $\epsilon \equiv 1$. While it may not be immediately apparent, the X axis in all plots is actually logarithmic.

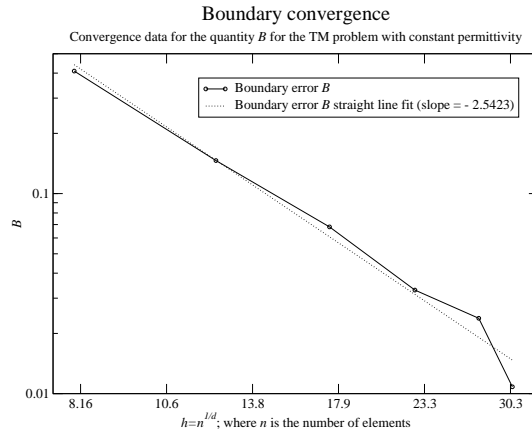


Figure 6.9. Convergence data for the boundary derivative error B in the case of constant permittivity.

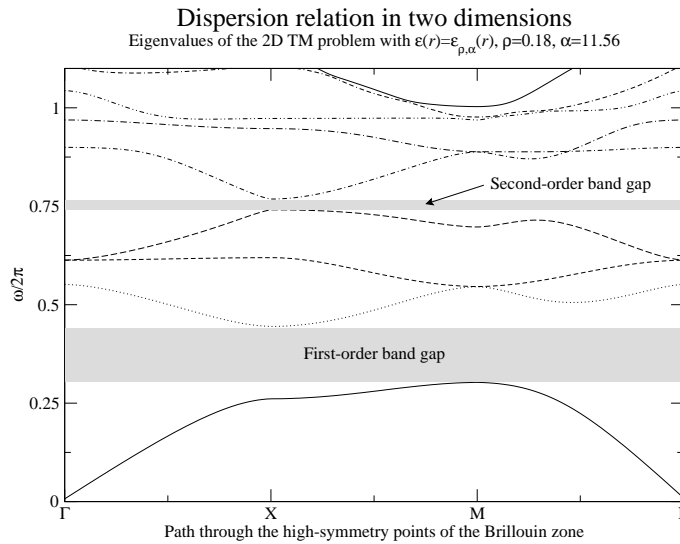


Figure 6.10. A dispersion relation of a crystal with a nontrivial, location-dependent permittivity ε . The same Brillouin zone patch and parametrization is used as in Figure 6.5.

- First, the exhaustive study conducted in the last section suggests that our FEM code deals well with eigenvalue problems under Floquet boundary conditions.
- Second, the only addition with respect to the last section are the piecewise-constant coefficients. The discussion of a similar problem with such coefficients in Section 5.3.3, where we could actually demonstrate convergence to the true solution, also bodes well for this case.
- Third, the article [BMGM⁺03] by Busch et al. contains imagery depicting the dispersion relation of the same crystal as the one of Figure 6.10. Within reasonable limits, the two graphs look very similar.
- And finally, we can see from Figure 6.11 that B converges almost as quickly as in the constant-coefficient case. This also suggests good numerical behavior.

These arguments should suffice to give you at least a certain amount of confidence in the solutions obtained, and thus we can turn to the application of the computed Bloch functions in the Wannier localization process.

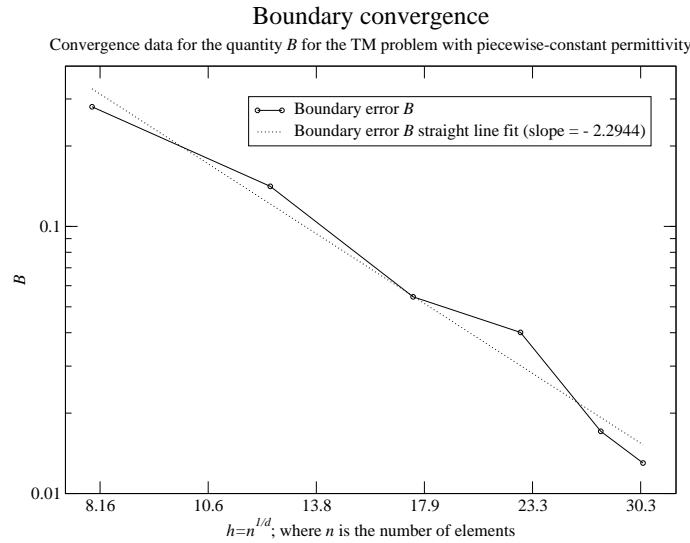


Figure 6.11. Convergence data for the boundary derivative error B in the case of piecewise constant permittivity $\varepsilon(\mathbf{r}) = \varepsilon_{\rho, \alpha}(\mathbf{r})$, with $\alpha = 11.56$ and $\rho = 0.18$.

Image `raw-wannier.eps` is excluded here for size reasons.

Figure 6.12. A “raw” Wannier function centered at $\mathbf{0}$ for the crystal of Figure 6.10. This function was computed from unprocessed Bloch functions.

6.5 Maximally localized Wannier functions

In concluding this chapter, we examine the results of our implementation of Marzari and Vanderbilt’s method, as described in Chapter 4. Before we begin, let us confirm the necessity of localization by reviewing an example Wannier function computed without localization. Figure 6.12 shows such a function and makes it fairly clear that these functions are far too irregular to be used as they are.

After applying the localization procedure, we obtain functions like the ones pictured in Figure 6.13ff. While definitely more usable than the “raw” functions, they have a number of shortcomings, some plainly visible, some not:

- Looking at the outer regions of the functions visualized in Figure 6.13ff., a few “humps” can be seen. While it is not clear that these can be eliminated by the minimization procedure, they at least have a decidedly suspicious look to them.
- As mentioned in Chapter 4, Marzari and Vanderbilt conjectured from their experience that maximally-localized Wannier functions have inherently constant phase, and thus can be made real by multiplying with a complex number of absolute value 1. Our functions do *not* satisfy this criterion. On the other hand, the criterion itself is not yet proven.

The single biggest problem we are facing right now is: While the computed functions doubtlessly *are* Wannier functions, how do we decide whether they are really maximally localized? Short of Marzari’s and Vanderbilt’s conjectured criterion, how do we reliably detect that we have fallen into a local minimum of the spread functional? To the best of my knowledge, there is no easily-decidable answer to this question.

Finally, let us shed some light on the internal performance characteristics of the Wannier spread minimization algorithm. Figure 6.16a) shows the value of the spread functional Ω for each step, Figure 6.16b) shows the step size in gradient units. The iteration is halted when it is decided that significant progress is no longer being made, indicated by the fact that the last 3 minimization steps each yielded less than 10^{-5} in progress.

From the convergence rate of the CG algorithm, it seems reasonable to believe that by our method we do find at least a local minimum of the spread functional. This is confirmed by Figure 6.17. This figure

Image `maxloc-wannier-1.eps` is excluded here for size reasons.

Figure 6.13. The maximally localized Wannier functions number 1, 2 (Fig. 6.14) and 3 (Fig. 6.15) centered at $\mathbf{0}$ for the crystal of Figure 6.10. Recall that we start numbering bands at 0. These Wannier functions were computed from bands 1, 2 and 3 of the underlying crystal—which make up the entangled block of bands between the first- and second-order band gap. All Wannier functions are drawn over a grid of 3×3 unit cells. Continued in Figure 6.14.

Image `maxloc-wannier-2.eps` is excluded here for size reasons.

Figure 6.14. The maximally localized Wannier function number 2. Continued from Figure 6.13. Continued in Figure 6.15.

depicts one step in the line minimization performed in each step of the CG algorithm described in Section 4.8. Along the search line, the value of Ω and its line derivative computed by centered differences are shown. To verify our previous results, the derivative of Ω as computed in Chapter 4 is also shown. The two derivatives agree fairly well if we ignore the substantial noise in the finite difference approximation for larger values of $|\alpha|$. This suggests that the gradient expression given at the end of Section 4.7 has some truth to it. The mistake in Marzari and Vanderbilt's paper [MV97] was found by disagreements in this kind of graph.

Image `maxloc-wannier-3.eps` is excluded here for size reasons.

Figure 6.15. The maximally localized Wannier function number 3. Continued from Figure 6.14.

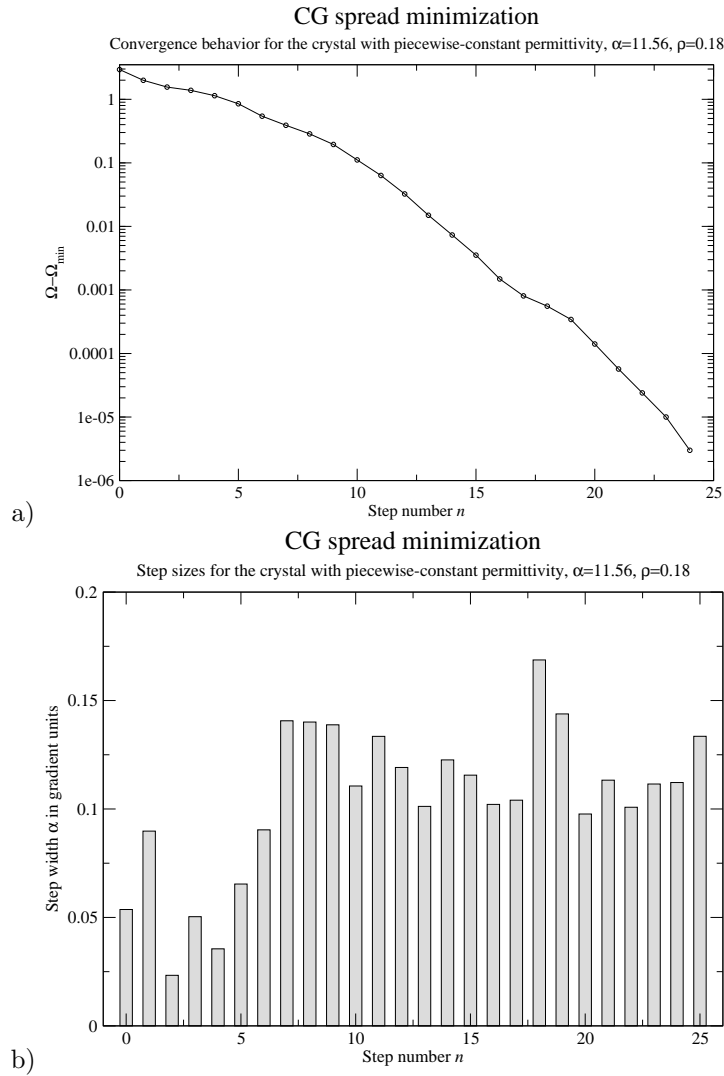


Figure 6.16. A characterization of the progress of the CG minimization algorithm applied to the crystal of Figure 6.10. a) shows the absolute values of Ω at each step, b) shows the step width used in each step in gradient units. Since the gradient vector's length decreases as we approach a minimum, it makes sense that the algorithm makes fairly constant-size steps.

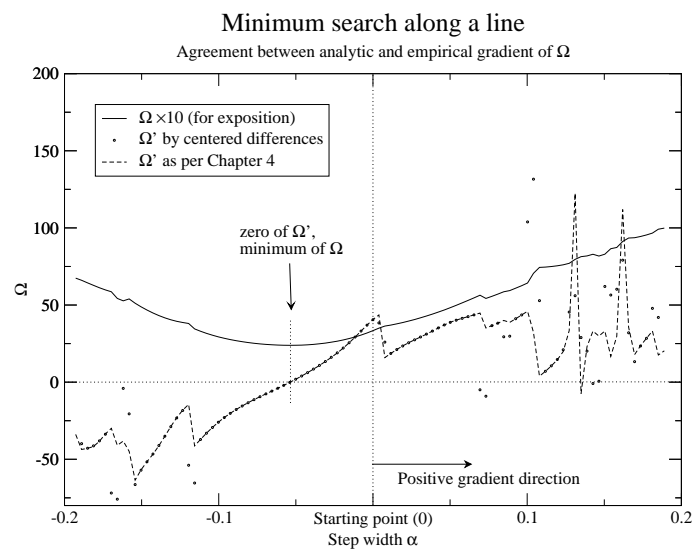


Figure 6.17. A line minimization step in the algorithm of Section 4.8.

Chapter 7

Conclusions and Future Work

After a fast-paced trip through many exciting areas of research, we are finally coming to the end of this thesis. What have we seen? From the bird's eye view, we have explored the foundations of the computation of maximally localized Wannier functions.

We started from Maxwell's equations with periodic coefficients and were able to represent them in a solvable way, summarizing the necessary results from functional analysis as we went along. We barely touched upon the distribution theory that would be used in a more in-depth treatment of the Dirac relations of Section 2.4, because this was beyond the scope of this work. Next, we spent some time examining the problem and its eigensolutions analytically. We obtained enough information to justify the construction of the Wannier functions and found ourselves deeply within the realm of solid state physics, which is rooted in the quantum theory. After obtaining a clearer view of the Wannier functions' role, we built some tools which were needed in following and justifying Marzari and Vanderbilt's method. The subsequent, detailed examination enabled some deeper insights into the gradient calculations, made possible by an inner-product view of directional derivative evaluation, as inspired by Prof. Dörfler. In concluding that section, we built up a complete description of a relatively fast and practical algorithm for obtaining maximally-localized Wannier functions. From there, we described an actual implementation of these ideas.

Foraying into applied computer science, a complete set of computational tools, ranging from matrix computations to a finite element toolkit, was built up from freely available bases. To the best of my knowledge, PyLinear is the first toolkit which allows the use of the same comprehensive set of powerful matrix primitives on both sides of a language barrier. While it has probably been done before in this way, the scheme with which FemPy attains exact boundary approximation was created by me without help from the literature. Also, FemPy's internal design and powerful interface were designed without substantial influence from other finite element packages. We concluded by presenting the results of the conducted computer experiments and providing a thorough numerical analysis of their validity. The knowledge contained in this thesis was unfortunately not all to be found in one place, many points only became clear as I scavenged through a diverse array of literature ranging from solid state physics journals from the 1960s, over classical mathematical volumes, to recent articles in numerical analysis. It was my goal to provide an accessible treatment that accurately reflects what I have learned over the last half year.

At the end of a sizable project such as this one, there are always things left unfinished, areas that could use some more attention and corners that seem worthy of being explored. In Chapter 5, we outlined opportunities for fruitful future work in each section. Of course, the theory can always be generalized and treated in more depth. In my opinion, it would be especially rewarding to keep studying the Wannier functions, which have so far not received much attention from within the mathematical community as far as I can tell. Neither Reed and Simon's treatment in [RS78], nor Kuchment's monograph [Kuc93] make any mention of the topic. They provide an alternative and interesting view, and I am confident that they will enable many exciting discoveries, both in functional and numerical analysis.

Appendix A

Auxiliary results

Theorem A.1 Let $U \subset \mathbb{R}^d$ be open, $d \in \{2, 3\}$, $f \in C^1(U, \mathbb{R})$, $\Omega \subset U$ admissible for Gauss' integral theorem. Then

$$\int_{\Omega} \nabla f(\mathbf{r}) d\mathbf{r} = \int_{\partial\Omega} f(\mathbf{r}) \mathbf{n} dS.$$

Proof Let $g_i(\mathbf{r}) := f(\mathbf{r})\mathbf{e}_i$, where \mathbf{e}_i is the i th unit vector. Then $\nabla \cdot f(\mathbf{r}) = \sum_i \operatorname{div} g_i(\mathbf{r})$. For linearity reasons, it suffices to show the claimed equality for just one g_i , or equivalently, for just one component of the vector, say the i th:

$$\begin{aligned} \left[\int_{\Omega} \nabla f(\mathbf{r}) d\mathbf{r} \right]_i &= \int_{\Omega} \operatorname{div} g_i(\mathbf{r}) d\mathbf{r} \\ &= \int_{\partial\Omega} g_i(\mathbf{r}) \cdot \mathbf{n} dS \\ &= \int_{\partial\Omega} f(\mathbf{r}) n_i dS, \end{aligned}$$

which establishes the claim. □

Corollary A.2 Let Ω be a primitive cell to a lattice L , f just like in Theorem A.1 and also let f be L -periodic. Then

$$\int_{\Omega} \nabla f(\mathbf{r}) d\mathbf{r} = 0.$$

Theorem A.3 (Characterization of operators with compact resolvent, Theorem XIII.64 in [RS78]) Let A be a self-adjoint operator bounded from below. Then the following are equivalent:

- a) A has compact resolvent, i.e. $(A - \mu \operatorname{Id})^{-1}$ is compact for all $\mu \in \rho(A)$.
- b) There exists a complete orthonormal basis $\{\varphi_n\}_{n=1}^{\infty}$ in $D(A)$ so that $A\varphi_n = \mu_n \varphi_n$ with $\mu_1 \leq \mu_2 \leq \dots$ and $\mu_n \rightarrow \infty$ as $n \rightarrow \infty$.
- c) $\{\psi \in D(A) : \|\psi\| \leq 1, \|A\psi\| \leq b\}$ is compact for all b .

Theorem A.4 (An implicit function theorem, Theorem 4.3 in [Sch02]) Let

$$f : X \times Y \supset U \rightarrow Z$$

be continuous in a neighborhood U of a solution (x_0, y_0) of $f(x, y) = 0$. Let the derivative $D_x f$ exist and be continuous on U . If

$$D_x f(x_0, y_0) : X \rightarrow Z$$

is an isomorphism, then the equation $f(x, y) = 0$ can be solved for x uniquely in a neighborhood of y_0 :

a) There exists an $r > 0$ such that for exactly one function $u : Y \supset B_r(y_0) \rightarrow X$ the equalities

$$f(u(y), y) = 0 \quad \text{and} \quad u(y_0) = x_0$$

hold for all $y \in B_r(0)$.

b) $f \in C^1$ implies $u \in C^1(B_r(y_0), X)$ and we have

$$D_y u(y) = -[D_x f(u(y), y)]^{-1} D_y f(u(y), y).$$

Bibliography

- [ADHO01] David Ascher, Paul F. Dubois, Jim Hugunin, and Travis Oliphant. *Numerical Python*. Lawrence Livermore National Laboratory, Livermore, CA, 2001.
- [AK99] Waldemar Axmann and Peter Kuchment. An efficient Finite Element Method for Computing Spectra of Photonic and Acoustic Band Gap Materials, I. Scalar Case. *Journal of Computational Physics*, 150:468–481, 1999.
- [AM76] Neil W. Ashcroft and N. David Mermin. *Solid State Physics*. Saunders College Publishing, 1976.
- [BGBM93] Angelika Bunse-Gerstner, Ralph Byers, and Volker Mehrmann. Numerical Methods for Simultaneous Diagonalization. *SIAM Journal of Matrix Analysis and Applications*, 14(4):927–949, 1993.
- [Blo62] E. I. Blount. Formalisms of Band Theory. *Solid State Physics*, 13:305–373, 1962.
- [BMGM⁺03] Kurt Busch, Sergei F. Mingaleev, Antonio Garcia-Martin, Matthias Schillinger, and Daniel Hermann. The Wannier function approach to Photonic Crystal Circuits. *J. Phys. Cond. Mat.*, 15, 2003.
- [Bre73] Richard P. Brent. *Algorithms for Minimization without Derivatives*. Prentice-Hall, Englewood Cliffs, NJ, 1973.
- [CS96] Jean-Francois Cardoso and Antoine Souloumiac. Jacobi Angles for Simultaneous Diagonalization. *SIAM Journal of Matrix Analysis and Applications*, 17:161, 1996.
- [Dob99] David C. Dobson. An efficient method for band structure calculations in 2D photonic crystals. *Journal of Computational Physics*, 149:363–376, 1999.
- [Geu02] Roman Geus. *The Jacobi-Davidson algorithm for solving large sparse symmetric eigenvalue problems with application to the design of accelerator cavities*. PhD thesis, Eidgenössische Technische Hochschule, Zürich, 2002.
- [GFS03] Francois Gygi, Jean-Luc Fattebert, and Eric Schwegler. Computation of Maximally Localized Wannier Functions using a simultaneous diagonalization algorithm. *Comput. Phys. Comm.*, 155(1):1–6, 2003.
- [Kuc93] Peter Kuchment. *Floquet Theory for Partial Differential Equations*. Birkhäuser Verlag, Basel, 1993.
- [Kuc01] Peter Kuchment. The Mathematics of Photonic Crystals. In Gang Bao, Lawrence Cowsar, and Wen Masters, editors, *Mathematical Modelling in Optical Science*, volume 22 of *Frontiers in Applied Mathematics*, chapter 7, pages 207–272. SIAM, 2001.
- [LSY97] R. B. Lehoucq, D. C. Sorensen, and C. Yang. *The ARPACK Users' Guide*. Department of Computational and Applied Mathematics, Rice University, Houston, TX, 1997.
- [MP76] Hendrik J. Monkhorst and James D. Pack. Special points for Brillouin-zone integrations. *Physical Review B*, 13:5188–5192, 1976.

- [MV97] Nicola Marzari and David Vanderbilt. Maximally-localized generalized Wannier functions for composite energy bands. *Physical Review B*, 56:12847, 1997.
- [MVS04] Nicola Marzari, David Vanderbilt, and Ivo Souza. wannier.f. Available from <http://www.wannier.org/>, 2004.
- [Pet] Tim Peters. The Zen of Python. Available from any Python interpreter by typing `import this`.
- [RS78] Michael Reed and Barry Simon. *Analysis of operators*, volume 4 of *Methods of modern mathematical physics*. Academic Press, Inc., New York, 1978.
- [Rud91] Walter Rudin. *Functional Analysis*. McGraw-Hill, Singapore, second international edition, 1991.
- [Sch81] Martin Schechter. *Operator Methods in Quantum Mechanics*. Elsevier/North Holland, New York, 1981.
- [Sch02] Ben Schweizer. Nichtlineare Funktionalanalysis. Lecture in the Winter Term of 2001/02 at Universität Heidelberg. Available from <http://www.iwr.uni-heidelberg.de/groups/amj/People/Ben.Schweizer/skripte.html>, 2002.
- [She94] Jonathan Richard Shewchuk. An Introduction to the Conjugate Gradient Method Without the Agonizing Pain. <http://www-2.cs.cmu.edu/~jrs/jrspapers.html>, Aug 1994.
- [She96] Jonathan Richard Shewchuk. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In Ming C. Lin and Dinesh Manocha, editors, *Applied Computational Geometry: Towards Geometric Engineering*, volume 1148 of *Lecture Notes in Computer Science*, pages 203–222. Springer-Verlag, May 1996. From the First ACM Workshop on Applied Computational Geometry.
- [Str92] Walter A. Strauss. *Partial Differential Equations—An Introduction*. John Wiley and Sons, Inc., 1992.
- [Vel00] Todd Veldhuizen. Techniques for Scientific C++. Technical Report 542, Indiana University Computer Science Department, 2000.
- [Wan37] Gregory H. Wannier. The Structure of Electronic Excitation Levels in Insulating Crystals. *Physical Review*, 52:191–197, 1937.
- [WC03] D. M. Whittaker and M. P. Croucher. Maximally localized Wannier functions for photonic lattices. *Physical Review B*, 67:085204, 2003.