Pattern formation in interacting systems

Elias Alstead, Advisor: Xiaofeng Ren

May 16, 2020

Abstract

A one species interacting system is a periodic assembly of discs that are arranged to minimize energy. We study systems that have either one or two discs per period cell. Numerical verification is provided that for the one disc configuration, the minimal assembly is the hexagonal lattice. For the two disc configuration, we place one disc at a lattice point and search for the optimal spot to place the second disc. The energy is minimized numerically utilizing the Green's function of the Laplace operator on a torus. The Green's function has either three or five critical points, where three critical points are well-known. Extra critical points are determined in the case where the Green's function has five critical points. Placing the second disc at one of these extra points minimizes the energy of a lattice. After the optimal critical point for each lattice is computed, the minimal assembly is determined.

1 Introduction

Periodic structures arise in physical systems ranging from honeycomb to graphene to Bose-Einstein condensates. A one-species interacting system is modeled as an infinite lattice of parallelogram cells with either one or two discs arranged periodically in each cell. The study of these structures is motivated by the density functional theory for diblock copolymers. We look for the optimal assembly, which is the pattern that minimizes energy.

The pattern of each assembly is related to the Voronoi cell of a lattice point, or the corner of one of the parallelograms. The Voronoi cell is the collection of all points in the plane that are closer to a given lattice point than to any other lattice point. For example, the Voronoi cell for the hexagonal lattice, an important pattern, forms a regular hexagon.

2 The model

A lattice Λ on the complex plane is generated by two nonzero complex numbers α_1 and α_2. Define the parameter $\tau = \alpha_2/\alpha_1$ and require that $\mathrm{Im}(\tau) > 0$. Then

$$\Lambda = \{j_1\alpha_1 + j_2\alpha_2 : \ j_1, j_2 \in \mathbb{Z}\}. \tag{2.1}$$

Define by D_α the parallelogram cell

$$D_\alpha = \{t_1\alpha_1 + t_2\alpha_2 : t_1, t_2 \in [0, 1)\} \tag{2.2}$$

associated to the basis $\alpha = (\alpha_1, \alpha_2)$ of the lattice Λ. The lattice Λ defines an equivalence relation on $\mathbb{C}$ where two complex numbers are equivalent if their difference is in Λ. The resulting space of equivalence classes is denoted $\mathbb{C}/\Lambda$. It can be represented by D_α. Each equivalence class has precisely one representation in D_α.

Given a subset Ω of fixed area,

$$|\Omega| = \omega|\mathbb{C}/\Lambda|, \tag{2.3}$$

consider the energy

$$\mathcal{J}_\Lambda(\Omega) = \mathcal{P}_\Lambda(\Omega) + \frac{\gamma}{2}\int_{\mathbb{C}/\Lambda} |\nabla I_\Lambda(\Omega)(z)|^2 \, dz. \tag{2.4}$$

The function $I_\Lambda(\Omega)$ is the Λ-periodic solution of Poisson's equation

$$-\Delta I_\Lambda(\Omega)(z) = \chi_\Omega(z) - \omega \text{ in } \mathbb{C}/\Lambda, \quad \int_{\mathbb{C}/\Lambda} I_\Lambda(\Omega)(z)\, dz = 0, \tag{2.5}$$

where χ_{Ω_j} is the characteristic function of Ω_j. Despite the appearance, the functional $\mathcal{J}_\Lambda$ depends on the lattice Λ instead of the particular basis α.

Denote

$$\mathcal{I}_\Lambda(\Omega) = \int_{\mathbb{C}/\Lambda} |\nabla I_\Lambda(\Omega)(z)|^2\, dz, \tag{2.6}$$

so that

$$\mathcal{J}_\Lambda(\Omega) = \mathcal{P}_\Lambda(\Omega) + \Big(\frac{\gamma}{2}\Big)\mathcal{I}_\Lambda(\Omega). \tag{2.7}$$

It is more appropriate to consider the energy per cell area

$$\widetilde{\mathcal{J}}_\Lambda(\Omega) = \frac{1}{|\mathbb{C}/\Lambda|}\mathcal{J}_\Lambda(\Omega). \tag{2.8}$$

The size and the shape of a configuration play different roles in its energy. To separate the two factors write the basis of a lattice as $t\alpha = (t\alpha_1, t\alpha_2)$ where $t \in (0,\infty)$ and the parallelogram generated by $\alpha = (\alpha_1, \alpha_2)$ has the unit area, i.e. $|D_\alpha| = 1$. The configuration is now denoted by $t\Omega$, with t measuring the size, and Ω describing the shape. The lattice generated by α is denoted Λ and the lattice generated by $t\alpha$ is $t\Lambda$.

Between the two lattices, the functions $I_{t\Lambda}(t\Omega)$ and $I_\Lambda(\Omega)$ are related by

$$I_{t\Lambda}(t\Omega)(w) = t^2 I_\Lambda(\Omega)(z), \quad tz = w,\ z, w \in \mathbb{C} \tag{2.9}$$

because of the equation (2.5).

Then the energy per cell area is

$$\widetilde{\mathcal{J}}_{t\Lambda}(t\Omega) = \frac{1}{t}\mathcal{P}_\Lambda(\Omega) + \frac{t^2\gamma}{2}\mathcal{I}_\Lambda(\Omega).$$

With respect to t, the last quantity is minimized at

$$t = t_\Omega = \Big(\frac{\mathcal{P}_\Lambda(\Omega)}{\gamma\mathcal{I}_\Lambda(\Omega)}\Big)^{1/3} \tag{2.10}$$

and the minimum value is

$$\Big(\frac{3}{2}\Big)\big(\mathcal{P}_\Lambda(\Omega)\big)^{2/3}\big(\gamma\mathcal{I}_\Lambda(\Omega)\big)^{1/3}. \tag{2.11}$$

Now drop the unimportant constants and define

$$\mathcal{K}_\Lambda(\Omega) = \big(\mathcal{P}_\Lambda(\Omega)\big)^{2/3}\big(\mathcal{I}_\Lambda(\Omega)\big)^{1/3}, \quad \Omega \subset \mathbb{C}/\Lambda,\ |\Lambda| = 1. \tag{2.12}$$

We study this quantity only for Ω being one disc or a union of two equal size discs. In this scenario $\mathcal{K}_\Lambda(\Omega)$ may be expressed in more explicit forms.

Let the Green's function of the Laplace operator be

$$\begin{aligned} G_\Lambda(z) = \frac{|z|^2}{4|\Lambda|} - \frac{1}{2\pi}\log\Big|\,\mathrm{e}\Big(\frac{z^2\overline{\alpha_1}}{4i|\Lambda|\alpha_1} - \frac{z}{2\alpha_1} + \frac{\alpha_2}{12\alpha_1}\Big)\Big(1-\mathrm{e}(\frac{z}{\alpha_1})\Big) \\ \prod_{n=1}^{\infty}\Big(\big(1-\mathrm{e}(n\tau+\frac{z}{\alpha_1})\big)\big(1-\mathrm{e}(n\tau-\frac{z}{\alpha_1})\big)\Big)\Big|. \end{aligned} \tag{2.13}$$

Sometimes one singles out the singularity of G_Λ at 0 and decomposes G_Λ into

$$G_\Lambda(z) = -\frac{1}{2\pi}\log\frac{2\pi|z|}{\sqrt{|\Lambda|}} + \frac{|z|^2}{4|\Lambda|} + H_\Lambda(z) \tag{2.14}$$

where

$$H_\Lambda(z) = -\frac{1}{2\pi}\log\Big|\,\mathrm{e}\Big(\frac{z^2\overline{\alpha_1}}{4i|\Lambda|\alpha_1} - \frac{z}{2\alpha_1} + \frac{\alpha_2}{12\alpha_1}\Big)\frac{\sqrt{|\Lambda|}}{2\pi z}\Big(1-\mathrm{e}(\frac{z}{\alpha_1})\Big)$$
$$\prod_{n=1}^{\infty}\Big(\big(1-\mathrm{e}(n\tau+\frac{z}{\alpha_1})\big)\big(1-\mathrm{e}(n\tau-\frac{z}{\alpha_1})\big)\Big)\Big| \tag{2.15}$$

is a harmonic function on $(\mathbb{C}\backslash\Lambda)\cup\{0\}$.

Take

$$\Omega = \bigcup_{k=1}^{K} B(\xi_k) = \bigcup_{k=1}^{K}\{z\in\mathbb{C}/\Lambda : |z-\xi_k|\le\rho\} \tag{2.16}$$

with

$$K\pi\rho^2 = \omega. \tag{2.17}$$

Given a disc $B(\xi_k)$ one finds

$$I_\Lambda(B(\xi_k))(z) = \left\{\begin{array}{ll} -\frac{|z-\xi_k|^2}{4}+\frac{\rho^2}{4}-\frac{\rho^2}{2}\log\rho, & \text{if } |z-\xi_k|\in[0,\rho] \\ -\frac{\rho^2}{2}\log|z-\xi_k|, & \text{if } |z-\xi_k|>\rho \end{array}\right\}$$
$$-\frac{\rho^2}{2}\log\frac{2\pi}{\sqrt{|\Lambda|}} + \frac{1}{4|\Lambda|}\Big(\pi\rho^2|z-\xi_k|^2+\frac{\pi\rho^4}{2}\Big) + \pi\rho^2 H_\Lambda(z-\xi_k) \tag{2.18}$$

by (2.14) and the mean value property of the harmonic function H_Λ. Then

$$\int_{B(\xi_k)}\int_{B(\xi_k)} G_\Lambda(z-w)\,dwdz = \int_{B(\xi_k)} I_\Lambda(B(\xi_k))(z)\,dz \tag{2.19}$$
$$= \pi^2\rho^4 H_\Lambda(0) - \frac{\pi\rho^4}{2}\log\frac{2\pi\rho}{\sqrt{|\Lambda|}} + \frac{\pi\rho^4}{8} + \frac{\pi^2\rho^6}{4|\Lambda|}.$$

When $k\neq l$,

$$\int_{B(\xi_k)}\int_{B(\xi_l)} G_\Lambda(z-w)\,dwdz = \int_{B(\xi_k)} I_\Lambda(B(\xi_l))(z)\,dz \tag{2.20}$$
$$= \pi^2\rho^4 G_\Lambda(\xi_k-\xi_l) + \frac{\pi^2\rho^6}{4|\Lambda|}.$$

In summary,

$$\mathcal{P}_\lambda\big(\bigcup_{k=1}^{K} B(\xi_k)\big) = 2K\pi\rho \tag{2.21}$$

$$\mathcal{I}_\Lambda\big(\bigcup_{k=1}^{K} B(\xi_k)\big) = \sum_{k=1}^{K}\sum_{l\neq k}\pi^2\rho^4 G_\Lambda(\xi_k-\xi_l) + K\Big(\pi^2\rho^4 H_\Lambda(0) - \frac{\pi\rho^4}{2}\log\frac{2\pi\rho}{\sqrt{|\Lambda|}} + \frac{\pi\rho^4}{8}\Big) + \frac{K^2\pi^2\rho^6}{4|\Lambda|}. \tag{2.22}$$

These quantities are substituted into (2.12).

3 The modular group and a fundamental region

A lattice has many bases. If $\alpha_1', \alpha_2' \in \mathbb{C}$, $\mathrm{Im}(\alpha_2'/\alpha_1') > 0$, can be transformed to a given base α_1, α_2 by

$$\alpha_2' = a\alpha_2 + b\alpha_1 \tag{3.1}$$
$$\alpha_1' = c\alpha_2 + d\alpha_1 \tag{3.2}$$

where

$$a, b, c, d \in \mathbb{Z},\ ad - bc = 1, \tag{3.3}$$

then the lattice generated by α_1' and α_2',

$$\Lambda' = \{j_1\alpha_1' + j_2\alpha_2' :\ j_1, j_2 \in \mathbb{Z}\} \tag{3.4}$$

is the same as the lattice Λ in (2.1), i.e. $\Lambda' = \Lambda$. Since $\tau = \frac{\alpha_2}{\alpha_1}$ and $\tau' = \frac{\alpha_2'}{\alpha_1'}$, we have

$$\tau' = \frac{a\tau + b}{c\tau + d}. \tag{3.5}$$

These matrices form a group, called the modular group denoted by Γ,

$$\Gamma = \left\{ \begin{bmatrix} a & b \\ c & d \end{bmatrix} :\ a, b, c, d \in \mathbb{Z},\ ad - bc = 1 \right\}. \tag{3.6}$$

It acts on the upper half complex plane

$$\mathbb{H} = \{\tau \in \mathbb{C} :\ \mathrm{Im}(\tau) > 0\} \tag{3.7}$$

by (3.5):

$$\tau \to g\tau = \frac{a\tau + b}{c\tau + d} \ \text{ for all } \tau \in \mathbb{H}, \ \text{ where } g = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \in \Gamma. \tag{3.8}$$

Under this group action, the images of a single point in $\mathbb{H}$ form an orbit; namely, if $\tau \in \mathbb{H}$, then

$$\{g\tau \in \mathbb{H} :\ g \in \Gamma\} \tag{3.9}$$

is an orbit that includes the point τ. The following theorem tells more about this group action.

Theorem 3.1. *Let R be a subset of $\mathbb{H}$ given by*

$$R = \left\{ \tau \in \mathbb{H} :\ -\frac{1}{2} < \mathrm{Re}(\tau) \le \frac{1}{2},\ |\tau| \ge 1,\ \mathrm{Re}(\tau) \ge 0 \text{ if } |\tau| = 1 \right\} \tag{3.10}$$

Then R contains exactly one point from each of the orbits of Γ.

The region R is called a fundamental region of the group action Γ on $\mathbb{H}$. A proof of this theorem can be found in [1]. When searching for the minimal energy lattice, it suffices to restrict the search to this fundamental domain.

4 Critical point distribution

In this section we take $\alpha_1 = 1$ and $\alpha_2 = \tau$ so that the Green's function can be expressed in terms of Jacobi's theta function. This choice of basis does not satisfy the condition $|\mathbb{C}/\Lambda| = 1$. One can later scale this lattice by a multiple $\sqrt{\mathrm{Im}\,\tau}$ so that $\alpha_1 = \frac{1}{\sqrt{\mathrm{Im}\,\tau}}$ and $\alpha_2 = \frac{\tau}{\sqrt{\mathrm{Im}\,\tau}}$.

Critical point	$\mathcal{K}_\Lambda(\Omega)$
0.5	0.01608
$0.5i$	0.01608
$0.5 + 0.5i$	0.01586

Table 1: Critical points for the square lattice, with the basis vectors $\alpha_1 = 1$ and $\alpha_2 = i$. The energies $\mathcal{K}_\Lambda(\Omega)$ are listed for each.

Critical point	$\mathcal{K}_\Lambda(\Omega)$
0.5373	0.01598
$0.5373 + 0.3102i$	0.01592
$0.2686 + 0.4653i$	0.01598
$0.8059 + 0.4653i$	0.01598
$1.0746 + 0.6205i$	0.01592

Table 2: Critical points for the hexagonal lattice, with the normalized basis vectors $\alpha_1 = 1.07457$ and $\alpha_2 = 0.53728 + 0.93060i$. The energies $\mathcal{K}_\Lambda(\Omega)$ are listed for each.

To determine the distribution of critical points in the τ-plane, an infinite lattice is modeled as a torus. The Green's function can be written in terms of elliptic functions as

$$G(z) = -\frac{1}{2\pi}\log|\vartheta_1(z)| + \frac{\mathrm{Im}(z)^2}{2\mathrm{Im}(\tau)} + C(\tau). \tag{4.1}$$

The critical point equation $\nabla G = 0$ can be written as

$$\frac{\partial G}{\partial z} = \frac{\vartheta_1'(z)}{\vartheta_1(z)} + 2\pi i\frac{\mathrm{Im}(z)}{\mathrm{Im}(\tau)} = 0. \tag{4.2}$$

Each lattice has either three or five solutions to this equation [3]. The theta function $\vartheta_1(z)$ is

$$\vartheta_1(z) = 2\sum_{n=0}^{\infty}(-1)^n q^{(n+\frac{1}{2})^2}\sin\left((2n+1)\pi z\right), \tag{4.3}$$

where $q = e^{i\pi\tau}$.

It suffices to focus on the fundamental domain, where numerical errors are minimized. Numerical issues are especially prevalent near the real line, due to the condition $\mathrm{Im}(\tau) > 0$. Figure 1 shows a numerically generated plot of the critical point distribution in a portion of the τ-plane. This plot is compared to the theoretical expectation in [2] and agrees quite well.

Two important assemblies are the hexagonal and square lattices. Critical point data for these lattices are included as examples. The square lattice has only three critical points. These are shown numerically in Table 1 and Figure 2. The minimum point is $\frac{\alpha_1+\alpha_2}{2}$. The other two half periods are saddle points.

The hexagonal lattice has five critical points, which are listed in Table 2 and plotted in Figure 3. The three half periods are saddle points, and the two extra critical points are minima. The extra critical points can be expressed in terms of the basis vectors as $\frac{\alpha_1+\alpha_2}{3}$ and $\frac{2(\alpha_1+\alpha_2)}{3}$.

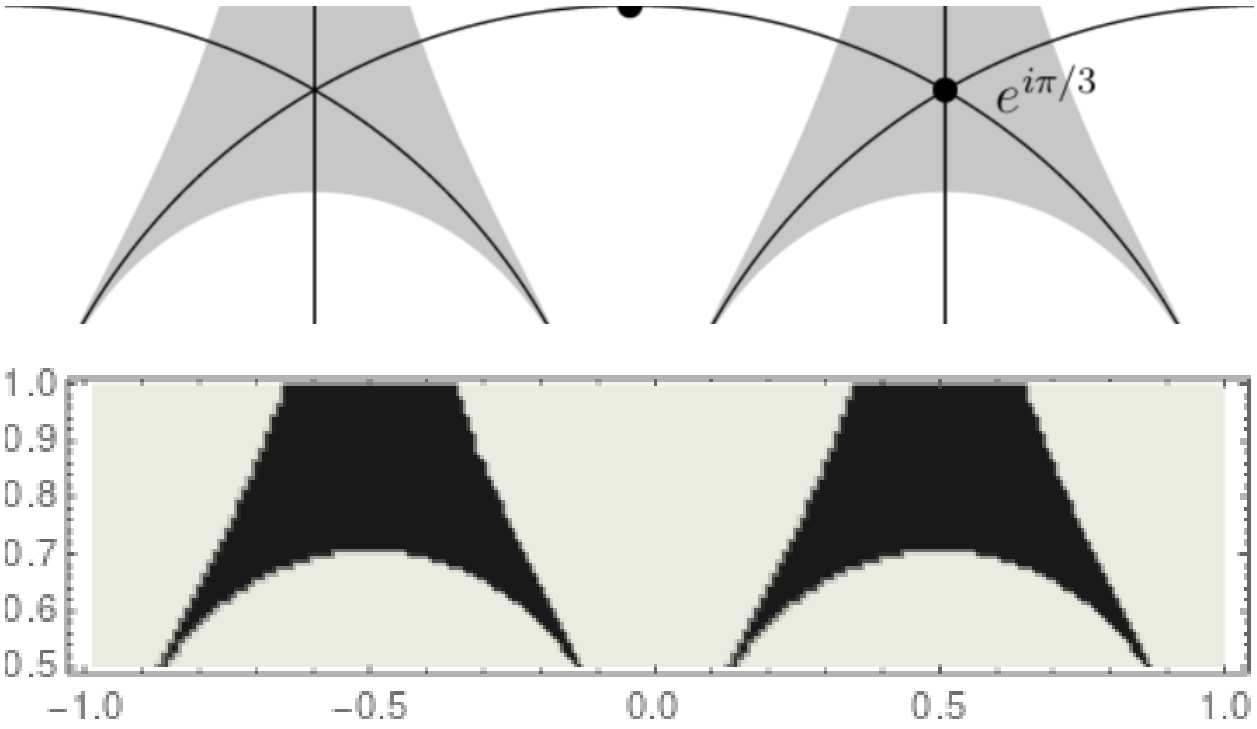

Figure 1: The figure in (a) is a section of the τ-plane as found in [2]. Gray indicates regions with five critical points and white designates lattices with three critical points. The plot in (b) is a numerical reproduction of (a).

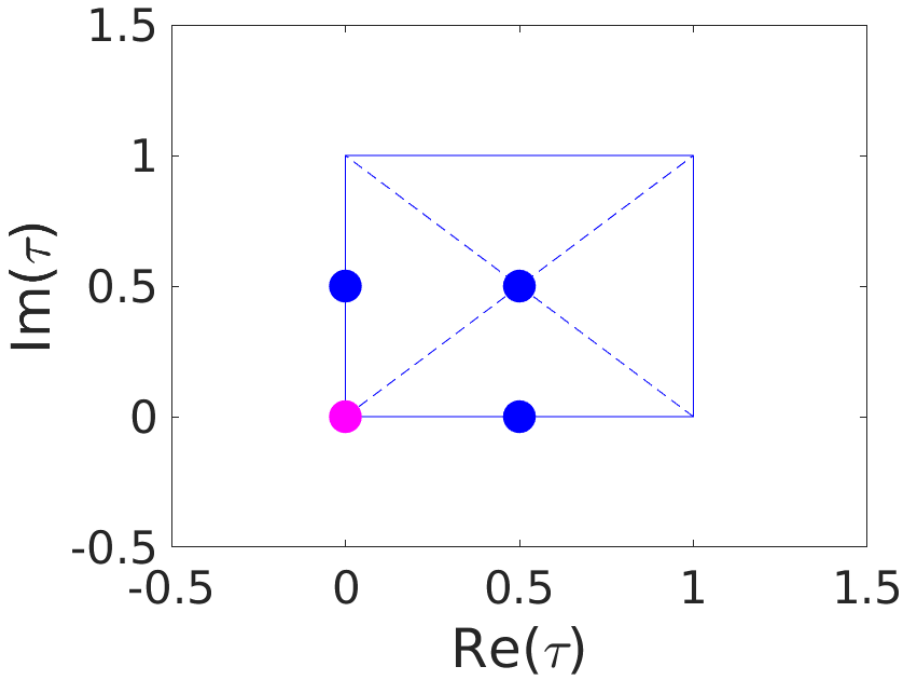

Figure 2: Graphical representation of the square lattice with two discs. The first disc is shown in pink and fixed at a lattice point. Blue dots represent the critical points and are possible locations for the second disc.

5 One disc configurations

Here we take Ω to be a single disc B of radius ρ in $\mathbb{C}/\Lambda$. Due to translation invariance, the center can be placed anywhere. Then

$$\mathcal{K}_\Lambda(B) = (2\pi\rho)^{2/3}\Big(\pi^2\rho^4 H_\Lambda(0) - \frac{\pi\rho^4}{2}\log\frac{2\pi\rho}{\sqrt{|\Lambda|}} + \frac{\pi\rho^4}{8} + \frac{\pi^2\rho^6}{4|\Lambda|}\Big)^{1/3}, \quad |\Lambda| = 1. \tag{5.1}$$

With ρ, the radius of B, fixed, $\mathcal{K}_\Lambda(B)$ can be minimized with respect to Λ, with $|\Lambda| = 1$. Then it suffices to minimize $H_\Lambda(0)$ with respect to Λ, with $|\Lambda| = 1$. This problem was studied by Chen and Oshita and it was shown that the minimum is attained at the hexagonal lattice [5]. Here we present a numerical verification of their result.

To determine the lattice with minimal energy, the τ-plane was scanned with a step size of 0.005. For each lattice, the energy was computed using (5.1) and plotted in Figure 4. Table 3 lists a sample of the computed energies. The lattice with the least energy is attained at $\tau = 0.5 + 0.865i \approx 0.5 + \frac{\sqrt{3}}{2}i$. This corresponds to the hexagonal lattice, as predicted by Chen and Oshita.

6 Two disc configurations: half-period case

Now we take Ω to be the union of two discs $B(\xi_1) \cup B(\xi_2)$ both of radius ρ. Because of the translation invariance of the Laplace operator, $\mathcal{K}_\Lambda(B(\xi_1) \cup B(\xi_2))$ depends on ξ_1 and ξ_2 via the relative displacement $\xi_2 - \xi_1$. Let

$$\zeta = \xi_2 - \xi_1. \tag{6.1}$$

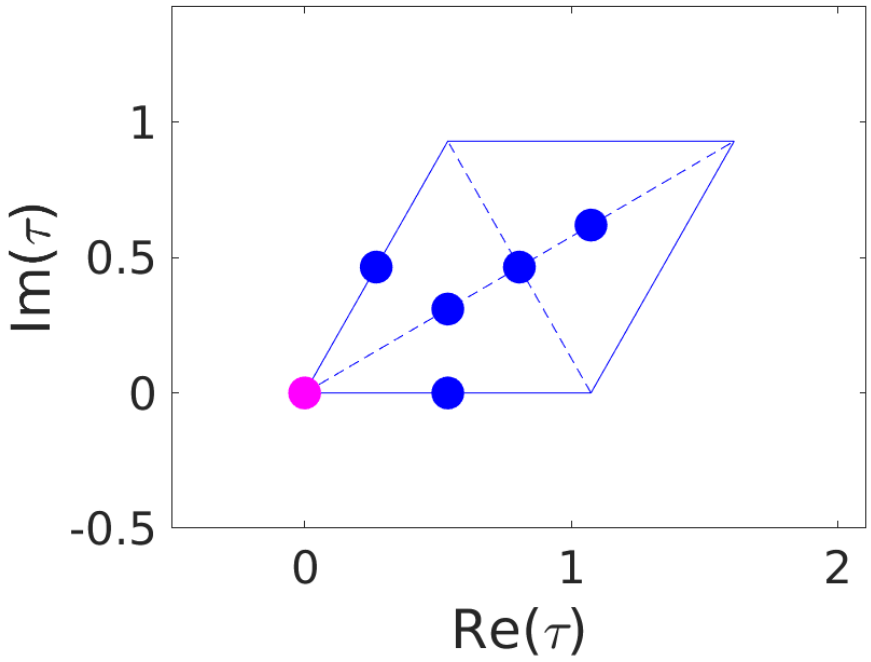

Figure 3: Graphical representation of the hexagonal lattice with two discs. The first disc is shown in pink and fixed at a lattice point. Blue dots represent the critical points and are possible locations for the second disc.

$\mathrm{Re}(\tau)$	$\mathrm{Im}(\tau)$	$\mathcal{K}_\Lambda(\Omega)$
-1.	0.005	0.0591286419
-0.95	0.005	0.015840356
-0.9	0.005	0.0158546804
-0.85	0.005	0.0154615308
-0.8	0.005	0.020709752
-0.75	0.005	0.0236024719
-0.7	0.005	0.0158546597
-0.65	0.005	0.0157387287
-0.6	0.005	0.020709752
-0.55	0.005	0.0153902278
-0.5	0.005	0.0370920524
-0.45	0.005	0.0153902278
-0.4	0.005	0.020709752
-0.35	0.005	0.0157387287
-0.3	0.005	0.0158546597
-0.25	0.005	0.0236024719
-0.2	0.005	0.020709752
-0.15	0.005	0.0154615308
-0.1	0.005	0.0158546804
-0.05	0.005	0.015840356
0.	0.005	0.0591286419
0.05	0.005	0.015840356
0.1	0.005	0.0158546804
0.15	0.005	0.0154615308
0.2	0.005	0.020709752
0.25	0.005	0.0236024719
0.3	0.005	0.0158546597
0.35	0.005	0.0157387287
0.4	0.005	0.020709752
0.45	0.005	0.0153902278
0.5	0.005	0.0370920524
0.55	0.005	0.0153902278
0.6	0.005	0.020709752
0.65	0.005	0.0157387287
0.7	0.005	0.0158546597
0.75	0.005	0.0236024719
0.8	0.005	0.020709752
0.85	0.005	0.0154615308
0.9	0.005	0.0158546804
0.95	0.005	0.015840356
1.	0.005	0.0591286419

Table 3: A sample of data for the one disc configuration.

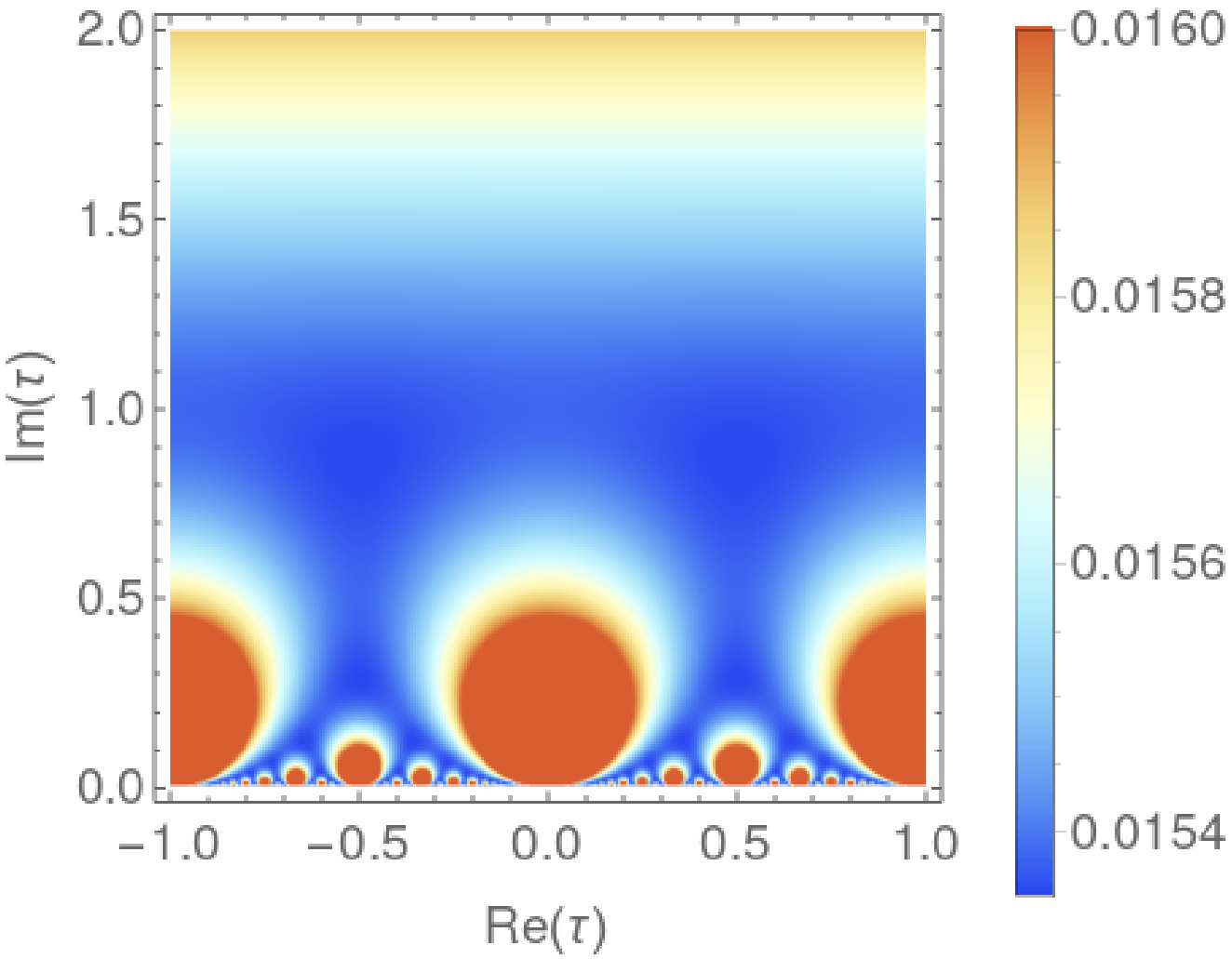

Figure 4: Plot of the τ-plane for the one disc configuration. Lower energies are in dark blue. The plane was scanned with a step size of 0.005 in both the real and imaginary directions. The minimum is attained numerically at $\tau = 0.5 + 0.865i \approx 0.5 + \frac{\sqrt{3}}{2}i$. This corresponds to the hexagonal lattice.

Then

$$\mathcal{P}_\lambda\big(B(\xi_1)\cup B(\xi_2)\big) = 4\pi\rho \tag{6.2}$$

$$\mathcal{I}_\Lambda\big(B(\xi_1)\cup B(\xi_2)\big) = 2\pi^2\rho^4 G_\Lambda(\zeta) + 2\Big(\pi^2\rho^4 H_\Lambda(0) - \frac{\pi\rho^4}{2}\log\frac{2\pi\rho}{\sqrt{|\Lambda|}} + \frac{\pi\rho^4}{8}\Big) + \frac{\pi^2\rho^6}{|\Lambda|} \tag{6.3}$$

from which $\mathcal{K}_\Lambda(B(\xi_1)\cup B(\xi_2))$ follows.

When Λ and ρ are fixed, $\mathcal{K}_\Lambda(B(\xi_1)\cup B(\xi_2))$ depends on ζ via $G_\Lambda(\zeta)$. We would like to choose ζ that minimizes $G_\Lambda(\zeta)$. Minimizing G_Λ turns out to be a difficult problem. Lin and Wang have shown that there are two cases, depending on Λ [3]. For many lattices, G_Λ has three critical points in $\mathbb{C}/\Lambda$ where one is a minimum and two are saddle points. If we represent $\mathbb{C}/\Lambda$ by a period parallelogram D_α, then the critical points are the half periods:

$$\frac{\alpha_1}{2},\quad \frac{\alpha_2}{2},\quad \frac{\alpha_1+\alpha_2}{2}. \tag{6.4}$$

However, for other Λ, G_Λ has five critical points. The three half periods are still critical points, but they are saddle points. The two extra critical points are minima.

In this section we take ζ to be a half-period, compute $\mathcal{K}_\Lambda(B(\xi_1)\cup B(\xi_2))$ with this ζ, and determine the lattice Λ with the lowest energy $\mathcal{K}_\Lambda(B(\xi_1)\cup B(\xi_2))$. The τ-plane is scanned with a step size of 0.005. For each lattice, the half period which minimizes $\mathcal{K}_\Lambda(B(\xi_1)\cup B(\xi_2))$ is determined, and the resulting energy is plotted in Figure 5. Table 4 lists a sample of the computed energies. The minimum is attained at $\tau = i$, which corresponds to the square lattice.

7 Two disc configurations: non-half-period case

Now we numerically find the extra critical points and take ζ to be one of them. Then we minimize $\mathcal{K}_\Lambda(B(\xi_1)\cup B(\xi_2))$ with respect to Λ. Two situations are considered. First, we consider only lattices that have five critical points and determine which of these has the minimal energy. Next, we consider all lattices, including those with only three critical points, and determine where the energy is minimized.

Figure 6 shows a portion of the fundamental domain and includes only lattices that have five critical points. The minimum lattice in this region is $\tau = 0.265 + 0.995i$.

When additionally considering lattices with three critical points, the minimum is once again attained at $\tau = i$, the square lattice. Figure 7 shows a plot of the fundamental domain for the τ-plane, which looks qualitatively similar to the half-period case shown in Figure 5. Table 5 lists a sample of the computed energies.

References

[1] L. Ahlfors. *Complex Analysis.* McGraw-Hill, 1979.

[2] W. Bergweiler and A. Eremenko. Green's function and anti-holomorphic dynamics on a torus. *Proc. Amer. Math. Soc.* 144 (2016): 2911-2922, 2015.

[3] C.S. Lin and C.L. Wang. Elliptic functions, Green functions and the mean field equations on tori. *Annals of Mathematics* 172 (2010): 911–954, 2010.

[4] E.J. Mueller and T.L. Ho. Two-component Bose-Einstein condensates with large number of vortices. *Phys. Rev. Lett.* 88 (18): 180403, 2002.

[5] X. Chen and Y. Oshita. An application of the modular function in nonlocal variational problems. *Arch. Rat. Mech. Anal.* 186 (1): 109–132, 2007.

$\mathrm{Re}(\tau)$	$\mathrm{Im}(\tau)$	$\mathcal{K}_\Lambda(\Omega)$
-0.495	0.87	0.0159462361
-0.475	1.36	0.0159199162
-0.455	1.9	0.0160401905
-0.43	1.38	0.0159169828
-0.405	0.915	0.015925165
-0.385	1.585	0.0159484352
-0.36	1.215	0.0158958933
-0.34	1.955	0.0160574387
-0.315	1.67	0.0159663973
-0.29	1.43	0.0159121029
-0.265	1.22	0.0158836429
-0.24	1.05	0.0158737833
-0.22	1.935	0.0160485588
-0.195	1.815	0.0160068462
-0.17	1.72	0.0159767301
-0.145	1.645	0.0159548218
-0.12	1.595	0.0159410547
-0.095	1.55	0.0159293398
-0.07	1.525	0.0159229791
-0.045	1.5	0.0159169179
-0.02	1.475	0.015911184
0.005	1.45	0.0159058088
0.03	1.425	0.0159008263
0.055	1.4	0.0158962741
0.08	1.375	0.015892193
0.105	1.34	0.0158869958
0.13	1.29	0.0158805546
0.155	1.225	0.0158740997
0.18	1.14	0.0158690712
0.205	1.03	0.0158687144
0.225	1.925	0.0160450144
0.25	1.765	0.0159919811
0.275	1.575	0.0159409049
0.3	1.345	0.0159000957
0.325	1.08	0.0158885102
0.345	1.835	0.016016249
0.37	1.485	0.0159272919
0.395	1.085	0.0159033142
0.415	1.725	0.0159849539
0.44	1.23	0.015908888
0.46	1.79	0.0160051697
0.485	1.175	0.0159181115

Table 4: A sample of data for the two disc, half-period configuration.

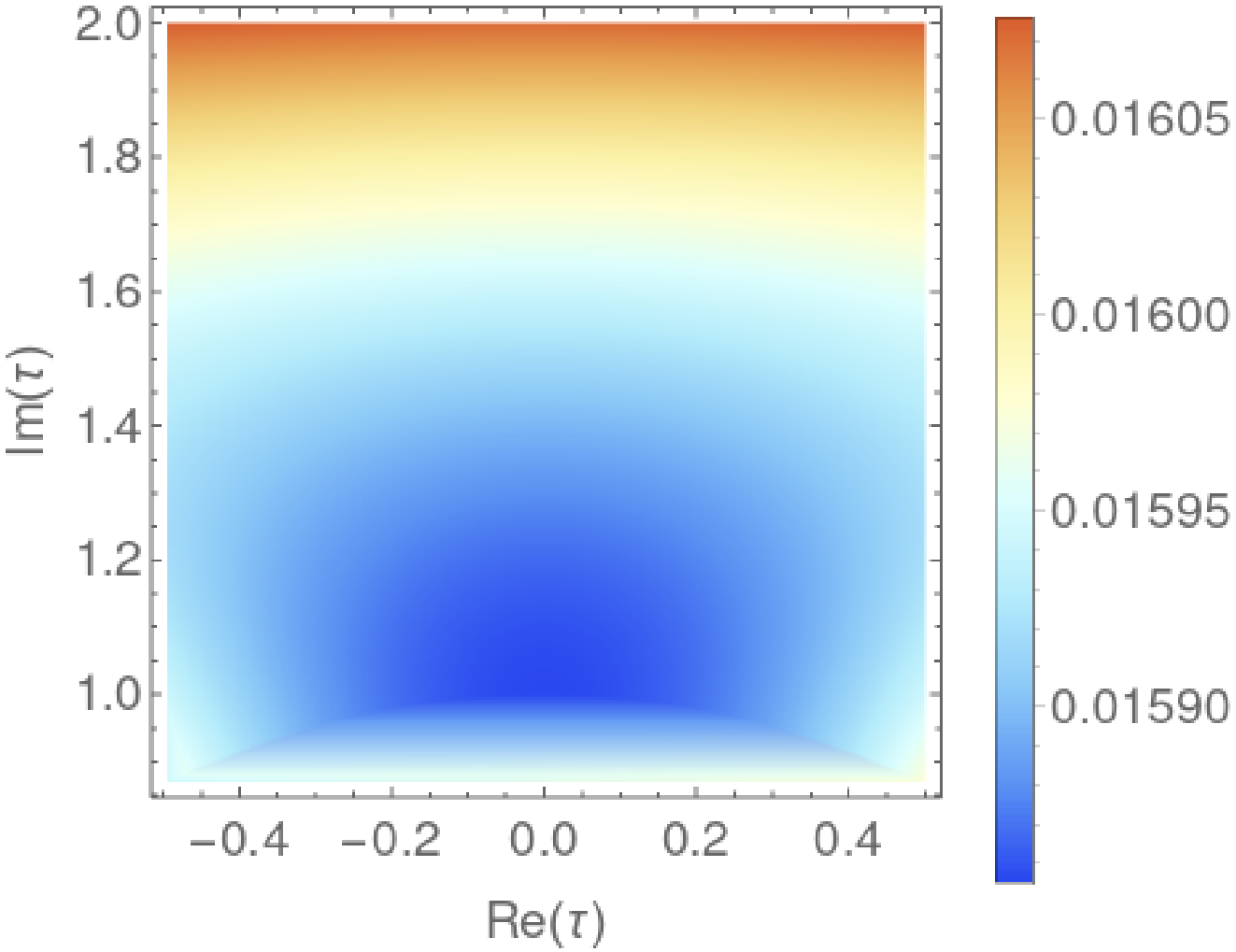

Figure 5: Plot of the τ-plane for the two disc configuration, considering only half-periods. Lower energies are in dark blue. The plane was scanned with a step size of 0.005 in both the real and imaginary directions. The minimum is attained numerically at $\tau = i$. This corresponds to the square lattice.

$\mathrm{Re}(\tau)$	$\mathrm{Im}(\tau)$	$\mathcal{K}_\Lambda(\Omega)$
-0.495	0.87	0.0159216791
-0.475	1.36	0.0159183553
-0.455	1.9	0.0160401905
-0.43	1.38	0.0159169828
-0.405	0.915	0.015914692
-0.385	1.585	0.0159484352
-0.36	1.215	0.0158958933
-0.34	1.955	0.0160574387
-0.315	1.67	0.0159663973
-0.29	1.43	0.0159121029
-0.265	1.22	0.0158836429
-0.24	1.05	0.0158737833
-0.22	1.935	0.0160485588
-0.195	1.815	0.0160068462
-0.17	1.72	0.0159767301
-0.145	1.645	0.0159548218
-0.12	1.595	0.0159410547
-0.095	1.55	0.0159293398
-0.07	1.525	0.0159229791
-0.045	1.5	0.0159169179
-0.02	1.475	0.015911184
0.005	1.45	0.0159058088
0.03	1.425	0.0159008263
0.055	1.4	0.0158962741
0.08	1.375	0.015892193
0.105	1.34	0.0158869958
0.13	1.29	0.0158805546
0.155	1.225	0.0158740997
0.18	1.14	0.0158690712
0.205	1.03	0.0158687144
0.225	1.925	0.0160450144
0.25	1.765	0.0159919811
0.275	1.575	0.0159409049
0.3	1.345	0.0159000957
0.325	1.08	0.0158885102
0.345	1.835	0.016016249
0.37	1.485	0.0159272919
0.395	1.085	0.0159028114
0.415	1.725	0.0159849539
0.44	1.23	0.0159079375
0.46	1.79	0.0160051697
0.485	1.175	0.015910182

Table 5: A sample of data for the two disc, non-half-period configuration.

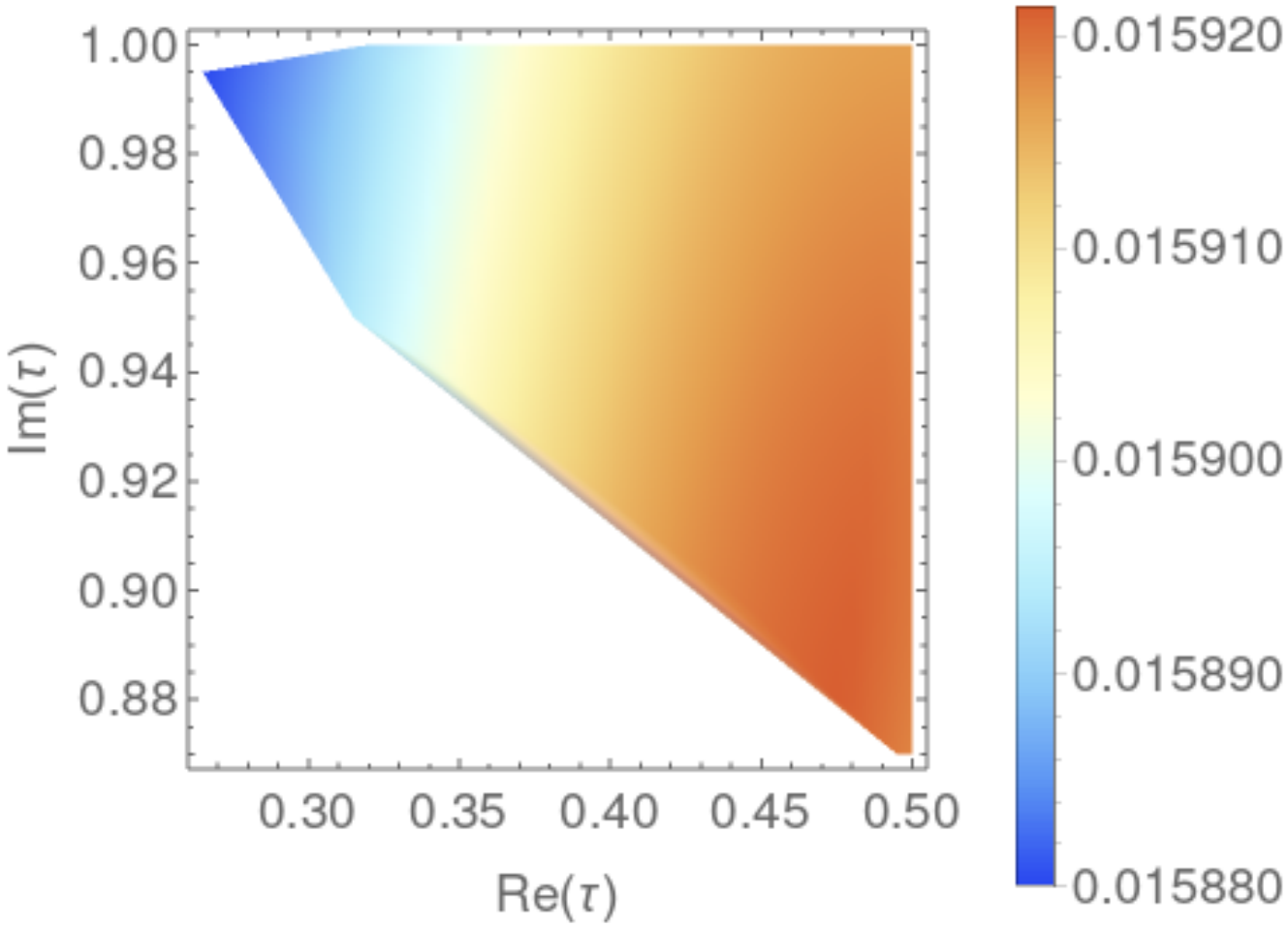

Figure 6: Plot of part of the fundamental domain, showing only lattices that have five critical points. The minimum is attained at $\tau = 0.265 + 0.995i$.

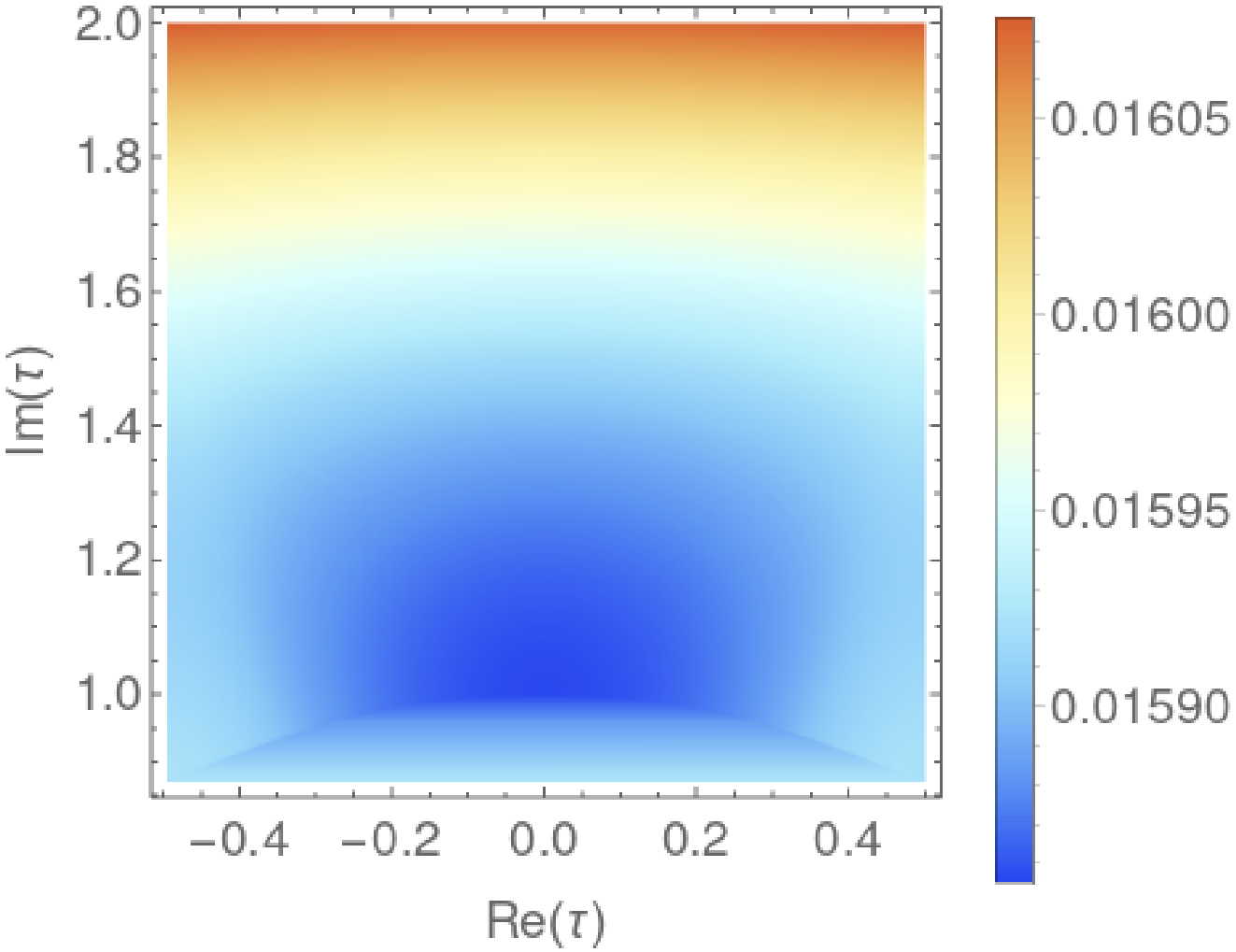

Figure 7: Plot of the τ-plane for the two disc configuration, considering all critical points. Lower energies are in dark blue. The plane was scanned with a step size of 0.005 in both the real and imaginary directions. The minimum is attained numerically at $\tau = i$. This corresponds to the square lattice.

A Appendix

The code used in determining the minimal lattice is split into four functions and displayed below. The first function, `Green`, calculates the Green's function at a point. The `FindRoots` function finds all critical points for a given lattice. This routine is used in the `MinLattice` function to choose the optimal location of the second disc in the two disc configuration problems. `MinLattice` also has an option to determine the minimal lattice for the one disc configuration. The routine `MinEnergy` is used to calculate the energy of a lattice.

```
function g = Green(z, a1, a2)

    function H = infProduct(z)
        H = 1;
        for n=1:1000
            H = H * (1 - exp(2*pi*1i*(n*(a2/a1)+(z/a1)))) ...
                * (1 - exp(2*pi*1i*(n*(a2/a1)-(z/a1))));
        end

        H = log(abs(H));
    end

    Lambda = det([real(a1) real(a2) ; imag(a1) imag(a2)]);

    infProd = infProduct(z);

    g = (abs(z)^2 / (abs(Lambda) * 4)) - (1 / (2 * pi)) * ...
        log(abs(exp(2*pi*1i*((z^2 * conj(a1) / (4i * ...
        abs(Lambda) * a1)) - (z / (2 * a1)) + (a2 / (12 * a1)))) * ...
        (1 - exp(2*pi*1i*(mod(real(z), 1) + 1i*imag(z))/a1)))) - ...
        (1 / (2 * pi)) * infProd;
end
```

```
function roots = FindRoots(a1, a2)

    tau = a2 / a1;

    % When searching for critical points, a1 must equal 1 and
    % a2 must equal tau. To then get the critical points for the normalized
    % basis vectors, multiply the critical points by the factor
    % 1. / sqrt(imag(tau))

    warning('off', 'MATLAB:singularMatrix')
    warning('off', 'MATLAB:nearlySingularMatrix')

    cpSearch(a1, a2);


    function cpSearch(a1, a2)
        a1Re = real(a1);
        a1Im = imag(a1);
        a2Re = real(a2);
        a2Im = imag(a2);

        % Preallocate space for array of roots
        roots = zeros(1, 500);
        index = 1;

        % Set step sizes
        stepReal = real(a1 + a2) / 10.;
        stepImag = imag(a1 + a2) / 10.;

        % The for loops iterate over a rectangular region.
```

```
        % However, the `inLattice` function used below prevents points
        % from outside the parallelogram from being considered.
        for j = min([0, a1Re, a2Re]):stepReal:max([(a1Re+a2Re), a1Im, a2Im])
            for k = min([0, a1Im, a2Im]):stepImag:max([(a1Im+a2Im), a1Im, a2Im])
                z = complex(j, k);

                % If the theta function equals 0, the gradient of the
                % Green's function will be infinite, so in this case,
                % skip the current loop iteration.
                if round(theta(z), 5) == 0
                    continue
                end

                    lastwarn('')

                    % Uses `cxroot` program available on MathWorks File Exchange
                    [root, ssq, ¬] = cxroot(@gradG, z, 'FunTol', 1e-10, ...
                                'XTol', 1e-10, 'MaxIter', 30);

                    root = round(root, 4);

                    % Ignores results from singular matrices
                    warnMsg = lastwarn;
                    if ¬isempty(warnMsg)
                        continue
                    end

                    % If root has not already been found,
                    % add root to list.
                    if isempty(find(roots == root, 1))

                        % Checks that the root that has been found
                        % is inside the parallelogram cell.
                        if inLattice(root, a1, a2)

                            % A large value of the sum of squares
                            % indicates that the point returned by
                            % `cxroot` is not actually a root.
                            if ssq > 1e-15
                                continue
                            end

                            roots(index) = root;
                            index = index + 1;
                        end
                    end
            end
        end

        roots = roots(1:index-1);
    end

    % Checks if point `z` is in the parallelogram cell
    function in = inLattice(z, a1, a2)

        x1 = [0, 0];
        x2 = [real(a1), imag(a1)];
        x3 = [real(a1+a2), imag(a1+a2)];
        x4 = [real(a2), imag(a2)];

        x = [x1(1), x2(1), x3(1), x4(1)];
        y = [x1(2), x2(2), x3(2), x4(2)];

        in = inpolygon(real(z), imag(z), x, y);

```

```
96          % Function checks if `point` is on the line between
97          % points `p1` and `p2`.
98          function on = onLine(p1, p2, point)
99              on = false;
100
101             % Calculate normal along the line connecting p1 and p2
102             p12 = p2 - p1;
103             l12 = sqrt(p12 * p12');
104             n12 = p12 / l12;
105
106             % Calculate normal along the line connecting p1 and point
107             p1p = point - p1;
108             l1p = sqrt(p1p * p1p');
109             n1p = p1p / l1p;
110
111             % If the normals are parallel, then the point is on the line
112             if round(dot(n12, n1p), 3) == 1
113                 on = true;
114             end
115         end
116
117         % Note that points on the right and top edges of the cell
118         % are considered to be in other cells.
119
120         % Checks whether `z` is on the right edge of the cell.
121         if onLine(x2, x3, [real(z), imag(z)])
122             in = false;
123         end
124
125         % Checks whether `z` is on the top edge of the cell.
126         if onLine(x4, x3, [real(z), imag(z)])
127             in = false;
128         end
129     end
130
131     % Gradient of the Green's function
132     function dGdz = gradG(z)
133         dGdz = (thetaPrime(z) / theta(z)) ...
134             + (2 * pi * 1i * imag(z) / imag(tau));
135     end
136
137     % Theta function, based on Lin and Wang 2010
138     function t = theta(z)
139         q = exp(1i * pi * tau);
140         t = 0;
141
142         for n = 0:30
143             t = t + ((-1)^n * q^((n+0.5)^2) * sin((2*n+1)*pi*z));
144         end
145
146         t = t * 2;
147     end
148
149     % Derivative of the theta function
150     function tp = thetaPrime(z)
151         q = exp(1i * pi * tau);
152         tp = 0;
153
154         for n = 0:30
155             tp = tp + ((-1)^n * q^((n+0.5)^2) * (2*n+1)*pi * cos((2*n+1)*pi*z));
156         end
157
158         tp = tp * 2;
159     end
160 end
```

```
function minEnergy = MinEnergy(K, a1, a2, point, G)

    % Local term
    function P = Pfunc(K, pho)
        P = K * 2 * pi * pho;
    end

    % Nonlocal term
    function I = Ifunc(K, pho, a1, a2, G)
        % Consider only lattices of size 1
        Lambda = 1.;

        % Value of the harmonic function at 0
        function h = hZero()

            infSum = 0;
            for n=1:1000
                infSum = infSum + log(abs((1 - exp(2*pi*1i*n*(a2/a1)))^2));
            end

            h = (-1/(2*pi)) * (log(abs(sqrt(imag(a2/a1)) * exp(2*pi*1i*(a2/a1)/12))) + ...
                infSum);
        end

        if K > 1
            Gterm = (K - 1) * pi^2 * pho^4 * G(point, a1, a2);

        else
            Gterm = 0.;
        end

        I = Gterm + K * ((pi^2 * pho^4 * hZero()) + ...
            (pi * pho^4 / 8) - (pi * pho^4 * ...
            log(2*pi*pho/sqrt(abs(Lambda))) / 2)) + ...
            (K^2 * pi^2 * pho^6 / (4 * abs(Lambda)));
    end

    % omega must be in (0, 1)
    omega = 0.01;
    pho = sqrt(omega / (K * pi));

    % Computes the K value
    minEnergy = (Pfunc(K, pho) ^ (2/3)) * ...
                (Ifunc(K, pho, a1, a2, G) ^ (1/3));
end
```

```
function MinLattice(K, filePath, G, cases)

    % The parameter `cases` can assume one of three values
    % (all referring to the two disc configuration):
    %    1) 'half' - place second disc at the minimal half period
    %    2) 'five' - consider only lattices with 5 critical points
    %    3) 'all'  - consider all lattices and place the second disc
    %                at the minimal critical point


    file = fopen(filePath, 'w');

    % Iterate over the fundamental domain in the tau plane
```

```
    for j = -0.495:0.005:0.5
        for k = 0.5:0.005:1
            tau = complex(j, k);

            % Skip points outside the fundamental domain
            if abs(tau) < 1
                continue
            end

            % Normalize the basis vectors
            a1 = 1 / sqrt(imag(tau));
            a2 = tau / sqrt(imag(tau));

            switch K
                case 1
                    % The `point` parameter of MinEnergy is not used for K=1.
                    % 0 is passed for simplicity but any value can be used.
                    energy = MinEnergy(K, a1, a2, 0, G);

                case 2
                    switch cases
                        case 'half'
                            % Determine the energy for each of the half periods,
                            % and choose the minimum.
                            energy1 = MinEnergy(K, a1, a2, a1/2, G);
                            energy2 = MinEnergy(K, a1, a2, a2/2, G);
                            energy12 = MinEnergy(K, a1, a2, (a1+a2)/2, G);

                            energy = min([energy1, energy2, energy12]);

                        case 'five'
                            % When using the FindRoots routine, we require that
                            % the basis vectors be 1 and tau.
                            w1 = 1.;
                            w2 = tau;

                            % Search for roots and normalize them
                            roots = FindRoots(w1, w2) / sqrt(imag(tau));

                            %  Skip lattices with 3 critical points
                            if length(roots) < 5
                                continue
                            end

                            gList = zeros(1, length(roots));
                            index = 1;
                            for root = roots
                                gList(index) = Green(root, a1, a2);
                                index = index + 1;
                            end

                            % Determine the critical point that minimizes
                            % the Green's function (there should be two
                            % minima, so take the smaller one).
                            point = min(roots(find(gList == min(gList))));

                            energy = MinEnergy(K, a1, a2, point, G);

                        case 'all'
                            % When using the FindRoots routine, we require that
                            % the basis vectors be 1 and tau.
                            w1 = 1.;
                            w2 = tau;

                            % Search for roots and normalize them
```

```
                                roots = FindRoots(w1, w2) / sqrt(imag(tau));

                                gList = zeros(1, length(roots));
                                index = 1;
                                for root = roots
                                    gList(index) = Green(root, a1, a2);
                                    index = index + 1;
                                end

                                % Determine the critical point that minimizes
                                % the Green's function (there should be two
                                % minima, so take the smaller one).
                                point = min(roots(find(gList == min(gList))));

                                energy = MinEnergy(K, a1, a2, point, G);
                        end
                end

                fprintf(file, '%.3f %.3f %.10f\n', j, k, energy);
            end
        end

        fclose(file);
    end
```

The following routine computes the Green's function in terms of elliptic functions, as found in [3]. It serves as a check that the code in **Green** correctly calculates the Green's function. However, this function is not suitable for use in the minimization routines as a replacement for **Green** since the determination of the constant $C(\tau)$ by an integral over the parallelogram cell is computationally intensive.

```
function g = GreenTheta(z, a1, a2)
    tau = a2 / a1;

    % Calculate the constant in the Green's function
    function C = GreenConstant()

        C = Square();

        function Ctau = Square()
            Gfunc = @(x, y) IntegrateG(x, y);

            % The constant is the negative of the integral over the cell
            Ctau = -integral2(Gfunc, 0, 1, 0, 1);

            function G = IntegrateG(w1, w2)

                % Change coordinates to integrate over [0,1]x[0,1]

                % Transformation matrix
                A = [real(a1) real(a2); imag(a1) imag(a2)];

                % integral2 sends in a matrix of values,
                % this code multiplies each value in the matrix by A
                x = zeros(size(w1));
                y = zeros(size(w2));
                for i = 1 : size(w1, 1)
                    for j = 1 : size(w1, 2)
                        Z = A * [w1(i, j); w2(i, j)];
                        x(i, j) = Z(1);
                        y(i, j) = Z(2);
                    end
                end

```

```
34                  G = (-1 / (2 * pi)) * log(abs(theta(complex(x, y)))) ...
35                      + (y.^2 / (2 * imag(tau)));
36              end
37          end
38      end
39
40      % Theta function, based on Lin and Wang 2010
41      function t = theta(z)
42          q = exp(1i * pi * tau);
43          t = 0;
44
45          for n = 0:50
46              t = t + ((-1)^n * q^((n+0.5)^2) * sin((2*n+1)*pi*z));
47          end
48
49          t = t * 2;
50      end
51
52      g = (-1 / (2 * pi)) * log(abs(theta(z))) ...
53          + (imag(z) ^ 2 / (2 * imag(tau))) + GreenConstant();
54  end
```