

周期ポテンシャルに入射した波束の時間発展

前田大輝

2018 年 1 月 19 日

目次

1	概要	4
2	研究の背景と目的	4
3	方法	5
3.1	Kronig-Penny モデルについて	5
3.2	QSB について	6
4	QSB のクラス構成	7
4.1	Mesh(x_min: Real, x_max: Real, dx: Real, t_min: Real, t_max, dt: Real)	7
4.2	Context(x_min: Real, x_max: Real, dx: Real, t_min: Real, t_max, dt: Real)	8
4.3	Field(arg:Union[iterable, function])	9
4.4	Potential(arg:Union[iterable, function])	10
4.5	state(arg:Union[iterable, function])	12
4.6	Hamiltonian(potential:Potential)	12
4.7	Schroedinger	13
5	QSB の妥当性検証	14
5.1	Zhang の研究結果の再現	14
6	条件設定	15
6.1	物理的状況の考察	15
7	結果	16
7.1	x-t パターン	16
7.2	全反射	16
7.3	高次反射波	17
7.4	高次進行波	18
7.5	分裂状態	19
7.6	屈折状態	20
7.7	トラップ状態	21
7.8	停滞状態	22
7.9	反射率と壁の厚さの関係	23
付録 A	Bloch の定理の簡単な証明	25
付録 B	Kronig-Penny モデルのエネルギーバンド構造の簡単な計算方法	25
付録 C	定常状態の計算	27

1 概要

Schrödinger 方程式に従う量子波束について、Kronig-Penny 型の副格子を持つステップ型ポテンシャルから作用を受けた場合の時間発展を数値計算によってシミュレートし多角的な解析を試みた。反射率と副格子の構造との関係について、壁が厚くなるごとに一定時間後の反射率が低下する場合が有ることをみいだした。また、確率分布の時空間的構造の解析から、非線形効果を取り入れていないにも関わらず、透過波束が二つ以上の波束に分裂する現象等をみいだした。

また、多様なポテンシャルについて統一的に解析を行うため、汎用ポテンシャルに対する数値計算を行うフレームワークを作成した。このフレームワークは多様な物理的状況について手軽にシミュレーションを行うことが出来る。

2 研究の背景と目的

周期ポテンシャルは結晶中によく現れる [1] ことから、物性物理学における重要な研究対象となっている。歴史的には Bloch [2] によって周期ポテンシャルが持つ並進対称性を利用した理論が作られ、結晶理論の基礎となっている。この理論の簡単で有効な実例として Kronig-Penny モデル [3] によるエネルギーバンド構造の説明が知られている。また、準結晶のように並進対称性が部分的に破れている場合は Anderson 局在 [4] を考慮に入れる理論がよく現実を説明できる近似として用いられている。さらに、真空中から結晶中に入射する場合のように並進対称性が大きく破れている場合は転送行列を直接用いることで付録 C のように反射率や透過率を求められる場合がある。

このように周期ポテンシャルにおける定常状態についてはかなり広い場合について理論的に基礎付けられている。非定常状態もこれらの定常状態を重ねあわせることで記述可能 [10] だが、波束のように局在化している場合は波動関数が周期性を持たない。よって、無限に長い波長を持つ固有状態にわたって積分する必要があり、解析的に解ける場合は限られている。

しかしながら、近年では上記の理論に直接基礎付けられていない方法での解析技術も生まれている。

例えば、実験的方法が可能となっている。BEC による物質波波束を構成する方法 [7] が確立されている。また、レーザ技術の発展によって周期ポテンシャルを構成する技術 [7] も確立されている。

ポテンシャル制御技術については、光格子時計のように高度に制御された場を作ることが可能になっている。これによって時間分解能の向上 [8] という時間発展の解析に必要な副次的技術も向上している。

また、理論的な研究も行われている。周期結晶や準結晶中の波束について、Zhang によるモデル計算 [9] が行われ、波束の存在確率の分散が時間の 3 乗程度に比例して増大する場合が発見されている。

通常の真空中における Gauss 型波束は時間の 2 乗に比例して分散が増大する [10] ため、真空中よりも高次の拡散 [11] が起こっている。一般にブラウン運動よりも高次の拡散は Heyperdiffusion として知られている。

Zhang の研究の様に量子波束の時間発展では興味深い現象が起こり得る。解析方法は数値計算で Schrödinger 方程式を直接解くという方法が取られている [12–16]。

本研究では真空中から Kronig-Penny 型の周期ポテンシャル中に入射する粒子という並進対称性が大きく崩れた場合を扱う。

このような場合にでも有効な数値計算法を確立すること、波束についての興味深い現象を発見すること

が本研究の目的となる。

3 方法

本研究では真空中から Kronig-Penny 型のポテンシャルに入射する波束の時間発展を数値計算によって解析した。数値計算には自作の数値計算フレームワークである QuantumSketchBook(QSB) [17] を作成し、使用した。本章では、使用したポテンシャル、パラメータおよびフレームワークを説明する。

3.1 Kronig-Penny モデルについて

非相対論的な電子の確率振幅 $\psi(x, t)$ は Schrödinger 方程式に従うことが知られている [10]。

$$\begin{aligned} i\frac{\partial}{\partial t}\psi(x, t) &= H\psi(x, t) \\ H &= -\frac{1}{2}\Delta + V(x) \end{aligned} \quad (3.1)$$

ただし原子単位系を用いて、 $\hbar = m = 1$ としている。Schrödinger 方程式は波動方程式型の微分方程式であり、物理的状況に応じてポテンシャル $V(x)$ と境界条件を与えることで解が決定する。

最も簡単なポテンシャルとしてステップ型、箱型、井戸型のポテンシャルが入門的な教科書 [10] に取り入れられている。これらの単純なポテンシャルであっても、トンネル効果等の重要な現象を予言することが出来るため、モデルとしても一定の価値を持っている。

結晶を構成する原子が極めて周期的に並んでいるという事実を反映させたものとして周期ポテンシャルが使われる。周期 l を持つポテンシャル V_{periodic} は以下の式に従う。

$$V_{\text{periodic}}(x) = V_{\text{periodic}}(x - l) \quad (3.2)$$

ポテンシャルが周期性を持つことから、系全体が並進対称性を持つことが言えるため、定常状態の確率振幅 $\phi(x)$ については Bloch の定理 [2] が成立する。

$$\phi(x) = u(x)e^{-iKx} \quad (3.3)$$

$u(x)$ はポテンシャルと同じ周期を持ち、系の構造に依存する関数で Bloch 関数と呼ばれる。 K はポテンシャルの周期に依存する定数で逆格子定数と呼ばれる。逆格子定数はポテンシャルの周期の間に以下の関係がある (詳しくは付録 A 参照)。

$$\begin{aligned} Kl &= 2n\pi \\ n &\in \mathbb{N} \end{aligned} \quad (3.4)$$

Bloch の定理が成立する場合について最も早くから定常解が知られているものの一つとして Kronig-Penny モデルがある。Kronig-Penny モデルは箱型ポテンシャルが周期的に並んだモデルとして定義される。ポテンシャルが高い部分のことを壁 (barrier) と呼び、ポテンシャルが低い部分のことを井戸 (well) と呼ぶ [7]。Kronig-Penny ポテンシャルの $(0, l]$ における定義は以下のようになる。

$$V_{KP}(x) = \begin{cases} V_0 & (0 < x \leq a) \\ 0 & (a < x \leq l) \end{cases} \quad (3.5)$$

ただし、壁の厚さを a 周期を l と置いている。自動的に井戸の幅は $b = l - a$ となる。(??) 式の条件を課して (3.5) 式の定義域を $(0, l]$ から \mathbb{R} に拡張できる。

$$V_{KP} = V_{KP}(x - l) \quad (3.6)$$

Kronig-Penny ポテンシャルは V_0, a, b の 3 つのパラメータによって特徴づけられる。

波動関数に C^1 級の制限を加えると Kronig-Penny モデルの定常解が得られる。(詳しくは付録 B 参照) 定常解は禁止帯を持つことから、エネルギーバンド構造を表現することができている。

周期的に並べることが出来るものの中で最も簡単なポテンシャルである箱型ポテンシャルを周期的に並べただけという点が Kronig-Penny モデルの重要な特徴といえる。

この単純さのお陰で、最も単純にエネルギーバンドを説明できるモデルの一つとしての教育的な価値が認められている [18]。

また、箱型ポテンシャルは完全に局在化しているため、反射、透過の定義が有限の位置で可能であるという特質も持っている。Kronig-Penny モデルのポテンシャルにおいても同様のことが言える。

3.2 QSB について

QSB は Schrödinger 方程式系の求解、可視化のためのフレームワークとして設計されている。使用の対象者として、数値計算に不慣れな物理学者を想定している。理解しやすくするために、データ構造に物理的な対応物を与えている点が特徴で、このような設計方法はオブジェクト指向 (Object Oriented Programming: OOP) と呼ばれている。

本質的ではない環境構築等の手間を少なくするため、python [19] での実装を行っている。python には以下のような性質があり、今回のようなフレームワーク作成に適している。

- 環境整備が簡単で無償 (mac 等ではプリインストールされている)
- マルチパラダイム言語 (手続き指向、オブジェクト指向、関数指向に対応している)
- 充実した科学計算系のライブラリ (numpy, scipy, matplotlib)

特に科学計算系のライブラリとして numpy および、それを内包した scipy [20] が線形代数、特殊関数、数値積分、微分、補間、並列計算等の計算を網羅的にサポートしている。

python の処理系は c, Fortran に比べて一般に低速だが、scipy は実質的な計算処理を高度に最適化された Fortran サブモジュールの BLASS, LAPACK 等に移譲することで高速な計算を可能としている。

ただし、機能を最大限に利用するためには python ネイティブのループを用いない等の最適化が必要で、本質的ではないノウハウが必要となる。

また、連続量の離散化など、定型的で退屈な処理はコードの可読性を悪化させる。

そこで、ある程度の最適化が施されたコードを物理的概念に対応させ、定型的な処理を隠蔽したフレームワークを作成した。

隠蔽をしたデータには物理学者にとって機能を想像しやすい命名を行った。(詳しくは 4 参照)

QSB では、Schrödinger 方程式の初期値問題を 10 行以内で記述出来る様になっている。最も簡単なユースケースを以下に示す。

Listing 1 samplecode.py

```
1 import QuantumSketchBook as QSB
```

```

2 with QSB.Context(-5, 5, 0.01, 0, 5, 0.01) as mesh:
3     my_potential = QSB.potential(lambda x: 1 / 2 * x ** 2)
4     my_state = QSB.gaussian_state(0, 2, 0.5)
5     my_hamiltonian = QSB.Hamiltonian(my_potential)
6     my_schroedinger = QSB.Schroedinger(my_hamiltonian, my_state)
7     QSB.plot(my_schroedinger, False, True, "sample_code")

```

7行で二乗に比例するポテンシャル内の Gauss 型波束の時間発展を記述できる。このコードを実行すると、以下のような $x-t$ パターンとポテンシャルのグラフが得られる。

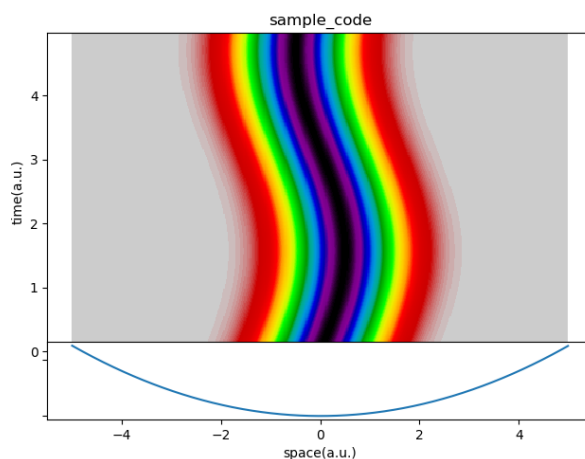


図1 サンプルコード実行結果

4 QSB のクラス構成

QuantumSketchBook のデータ構造はクラスとしてまとめられている。以下に主要なクラスを示す。

- Mesh : 離散化のデータを保持する
- Context : Mesh を配布する
- Field : 空間に依存する物理量を表す
- Potential : ポテンシャルを表す
- State : 初期状態を表す
- Hamiltonian : ハミルトニアンを表す
- Schrodinger : 初期値問題を計算する

また、QSB 独自概念として `plottable` という概念がある。これは QSB 付属の `plot()` 関数によって適切なグラフが出力される性質を持つことを指す。

4.1 Mesh(x_min: Real, x_max: Real, dx: Real, t_min: Real, t_max, dt: Real)

時空間の離散化に関するデータを保持する。

Mesh の構成過程で物理的に不合理な引数が設定された場合、`MeshError` が送出される。

様々なオブジェクトが離散化の際にこのクラスを利用するため、自動で Mesh にアクセスできるよう、Context クラスが提供されている。不慣れな利用者はこちらを使用すべきで、直接のインスタンス化 (具体的なデータを入力してデータ構造を生成すること) は推奨されない。異なった Mesh に依存した計算が行われそうになった場合、MeshError が送出される。それを避けるためにも後述の Context を通しての使用を推奨する。

Mesh は immutable(一度生成されると変更不可能) という性質を持っている。よって、Mesh を変更したい場合は 1 から新しい Mesh を生成しなければならない。しかし、この影響で python によって実行時に効率化されたメモリ配置がなされる。この性質によって、巨大なメッシュを貧弱な計算機でも動かすことができる。また、辞書のキーや集合の要素としても使用することができる。

プロパティ	説明
x_min	空間メッシュの最大値
x_max	空間メッシュの最小値
dx	空間メッシュの刻み
x_vector	空間メッシュの全要素
x_num	空間メッシュの全要素数
t_min	時間メッシュの最大値
t_max	時間メッシュの最小値
dt	時間メッシュの刻み
t_vector	時間メッシュの全要素
t_num	時間メッシュの全要素数

4.2 Context(x_min: Real, x_max: Real, dx: Real, t_min: Real, t_max, dt: Real)

Mesh の配布を行う。QSB のほとんどのクラスはこの Context が自動生成した Mesh を自動で受け取ることが出来るように作成されているため、利用者はこの Context をインスタンス化するだけで、離散化に関する処理を考える必要がなくなる。

複数の Context が存在する場合、最後にインスタンス化された Context の情報が優先されるが、既に生成されてしまったオブジェクトの情報は更新されないため、Mesh を複数使う必要がある場合は異なる Mesh に依存したオブジェクトが混在することになる。そこで、Context の有効範囲を明示化出来るよう with 構文によるコンテキストマネージャとしての使用法がサポートされている。with 構文は初期化処理と終了処理を隠蔽することができる構文で、Context の場合は with 構文を抜けたときに自動で最新の Context の情報を消去する終了処理を行っている。そのため、with 構文の内側で定義された、Mesh に依存するオブジェクトは必ず互いに同じ Mesh に依存していることが保証される。

Context.__enter__() メソッド (コンテキストマネージャとして使用された際に python 内部で呼ばれるメソッド) は Mesh を返す。as 節で束縛することで、Mesh を利用することができる。

Listing 2 Context の使用例

```

1 import QuantumSketchBook as QSB
2
3
4 with QSB.Context(-10, 10, 0.1, 0, 10, 0.1) as mesh:
5     print(mesh.__class__)
6     # --> <class 'QuantumSketchBook.mesh.Mesh'>
7
8     print(mesh.x_num)
9     # --> 200
10
11    print(mesh.t_num)
12    # --> 100
13
14    print(mesh.x_vector)
15    # --> [-1.00000000e+01 -9.90000000e+00 -9.80000000e+00 -9.70000000e+00 ...]

```

4.3 Field(arg:Union[iterable, function])

空間に依存する物理量を表す。引数には iterable(要素を次々と生成できるデータ構造 ex: list, tuple, generator) と一引数の関数のどちらも受け取ることができる。ただし、iterable の要素数と空間メッシュの数が一致しない場合 ValueError を送出する。

Field 内部で各位置での値の情報を Mesh と結びつけて保持する。Mesh が引数として与えられない場合、Context から自動的に情報を受け取る。Context が存在しない場合は MeshContextError が送出される。

抽象クラスとして設計されているため、このクラスを直接インスタンス化することは推奨されない。__plot__() メソッドが定義されているが、plottable ではない。

プロパティ	説明
vector	各空間メッシュにおける値
mesh	依存している Mesh

Listing 3 Field のサンプルコード

```

1 import QuantumSketchBook as QSB
2 from QuantumSketchBook.field import Field
3
4
5 with QSB.Context(0, 2, 1, 0, 2, 1) as mesh:
6     print(mesh.x_vector)
7     # --> [0 1]
8
9     function_field = Field(lambda x: x + 1) # 引数が関数の場合
10    print(function_field.vector)
11    # --> [1 2]
12
13    list_field = Field([1, 2]) # 引数がlistの場合

```

```

14     print(list_field.vector)
15     # --> [1 2]
16
17     generator_field = Field(x + 1 for x in range(2)) # iterableならば受け取ることができる
18     print(generator_field.vector)
19     # --> [1 2]
20
21     wrong_length = Field([1, 2, 3]) # 要素数が合わない場合はValueErrorが送出される
22     # --> ValueError: Length of input iterable must be equal to mesh.num

```

4.4 Potential(arg:Union[iterable, function])

ポテンシャルの情報を保持する。Field のサブクラスのため、配列と関数のどちらも受け取ることが出来る。また、具体的な Potential を構成するための関数 (box(), step(), kp() など) がいくつか用意されている。これらの基本的なポテンシャルを組み合わせても任意のポテンシャルを作ることができる。

このクラスではスカラー倍とポテンシャル同士の加減がサポートされている。スカラー倍ではすべての位置におけるポテンシャルがスカラー倍される。ポテンシャル同士の加減では、各位置におけるポテンシャルの値の和差がとられる。

plottable の性質を持つため、plot(some_potential) とすれば、簡単にグラフを作成できる。

matrix() メソッドは疎行列の形で行列要素を返すため、巨大な Mesh であっても $O(n)$ のオーダーでメモリを使うことができる。

プロパティ	説明
matrix()	行列要素を返す

Listing 4 Potential の使用例

```

1 import QuantumSketchBook as QSB
2
3
4 with QSB.Context(-10, 10, 0.01, 0, 2, 1):
5     step_potential = QSB.step(5, 0)
6     QSB.plot(step_potential, title="STEP") # ステップポテンシャル
7
8     box_potential = QSB.box(2, 1, 4)
9     QSB.plot(box_potential, title="BOX") # 箱型ポテンシャル
10
11     complex_potential = step_potential + box_potential
12     QSB.plot(complex_potential, title="COMPLEX") # ポテンシャルの足し合わせ
13
14     square_potential = QSB.Potential(lambda x: x ** 2)
15     QSB.plot(square_potential, title="SQUARE") # 2次関数
16
17     QSB.plot(QSB.kp(1, 2, 1), title="Kronig-Penny") # Kronig-Pennyポテンシャル
18     QSB.plot(QSB.vacuum_kp(1, 2, 1), title="vacuum_and_Kronig-Penny") # 真空とKronig-Pennyポテンシャル

```

以下のグラフが出力される。

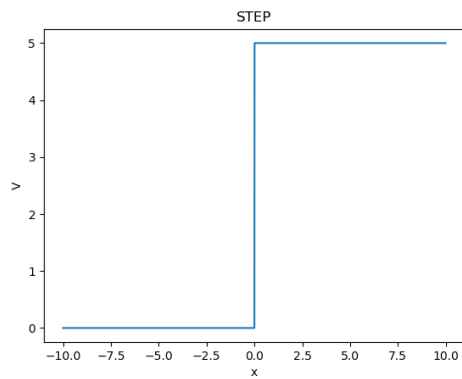


図2 ステップポテンシャル

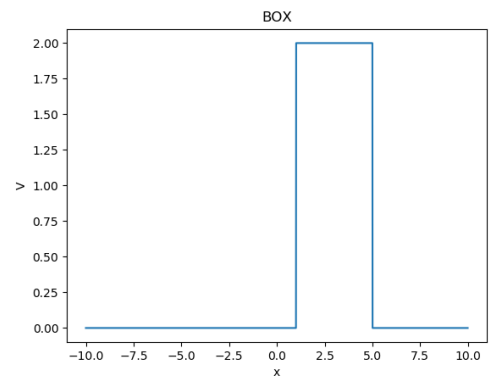


図3 箱型ポテンシャル

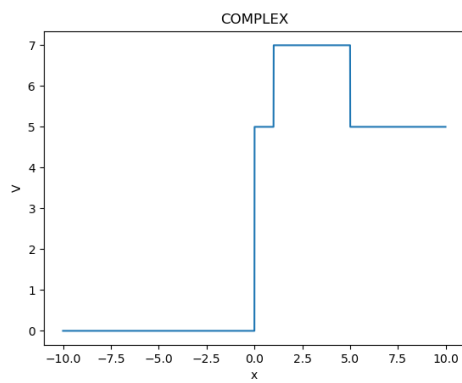


図4 ポテンシャルの足し合わせ

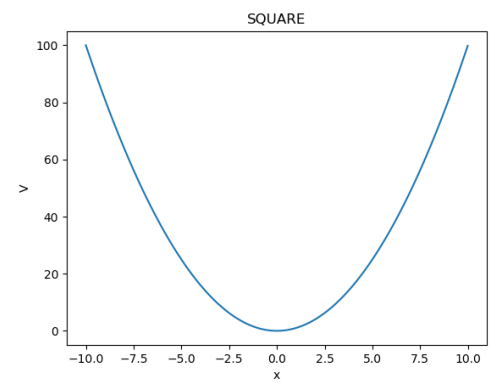


図5 二次関数

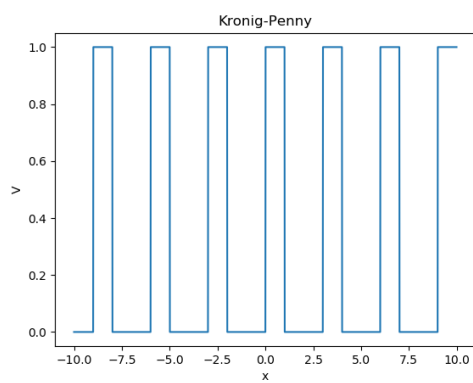


図6 Kronig-Penny ポテンシャル

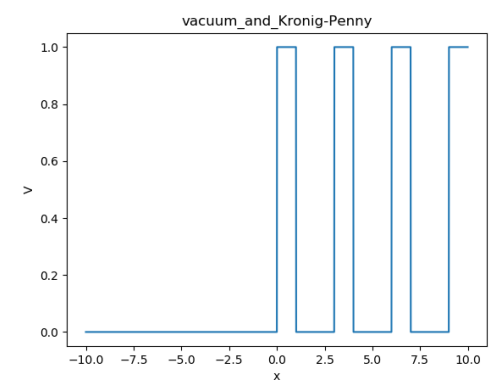


図7 真空と Kronig-Penny ポテンシャル

4.5 state(arg:Union[iterable, function])

初期状態に関する情報を保持する。Field のサブクラスのため、配列と関数を受け取ることが出来る。

random_values(n) メソッドを使うと State の確率密度分布に従う n 個の乱数を生成することが出来る。これは Nelson の確率過程を生成する際の初期値を得るために使われる。乱数生成には von Neumann の棄却法を用いている。

plottable であるため、plot(some_state) とすれば、グラフを作成できる。

Gauss 型波束の State を構成するための関数 gaussian_state() が用意されている。

Listing 5 State の使用例

```
1 import QuantumSketchBook as QSB
2
3
4 with QSB.Context(-10, 10, 0.01, 0, 2, 1):
5     gaussian = QSB.gaussian_state(0, 3, 0) # 平均0, 波動関数の標準偏差3, 波数0 のGauss型波束
6     QSB.plot(gaussian, title="GAUSS")
7
8     print(gaussian.random_values(5))
9     # --> [-1.6 -0.15 1.1 0.18 -1.4 ]
```

使用例を実行すると以下のグラフが得られる。プロットされているものが確率密度分布だということには注意が必要だと考えられる。波動関数の標準偏差が3のとき、絶対値2乗をとった確率密度分布の標準偏差は1.5になる。

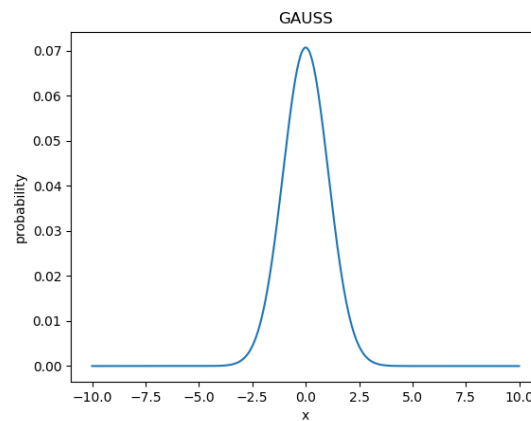


図 8 Gauss 型波束

4.6 Hamiltonian(potential:Potential)

Potential を受け取って、ハミルトニアン演算子を保持する。

行列要素は疎行列によって保持されているため、メモリ効率、時間効率ともに効率化されている。

境界条件の指定はこのクラスによって行われる。物理的には時空を管理する Mesh に保持させるべき

情報だが、同様の Mesh を異なった境界条件で計算する可能性が高いことから、全てのデータの基礎となっている Mesh ではなく、比較的に変更しやすい Hamiltonian に保持させる形になっている。

4.7 Schroedinger

初期値問題の求解を行う。Hamiltonian と State を受け取って Schrödinger 方程式の数値解を保持する。

数値解は初めて Schroedinger.solution() メソッドが呼び出されたときに計算され、キャッシュされる。よって、二回目以降の呼び出しは高速になる。また、Schroedinger は iterable であり、各ステップでの解を順々に返すことが出来る。この機能は Mesh が巨大すぎてメモリ消費が激しいときに有効な方法として設計されている。

plottable であり、plot すると x-t パターンを生成する。

5 QSB の妥当性検証

QSB の妥当性検証のため、既知の状況について QSB を用いた計算を行った。

5.1 Zhang の研究結果の再現

Zhang の研究では主格子の内部に副格子が存在している構造のポテンシャルを使用している。

主格子は箱状で、格子番号を i としたとき $m \in [-L, L]$ となるように設定されている。今回は $L = 50$ としている。

主格子内のポテンシャル構造は副格子の構造によって決定される。今回は Zhang が行った計算の中でも周期的なポテンシャルに相当する $V = V_0(-1)^i$ を使用した。ここで V_0 はポテンシャルの高さを決定する因子で、1.0, 1.5, 1.9, 2.0 が Zhang の論文では発表されている。

初期条件 ϕ_0 は Kronecker のデルタ型の波束 $\delta_{0,i}$ を使用した。

ハミルトニアン内に含まれるラプラシアンについては以下の式を使用した。

$$\Delta\phi_i := \frac{\phi_{i-1} - 2\phi_i + \phi_{i+1}}{\Delta x^2} \quad (5.7)$$

この式は $O(\Delta x)$ の精度がある [21]。

QSB で上記のラプラシアンを用いた $i \in [0, 10^4]$ について自由端境界条件をつけて $t < 10^4$ ステップに渡り時間発展を計算した。

分散は下記の図のように時間発展した。

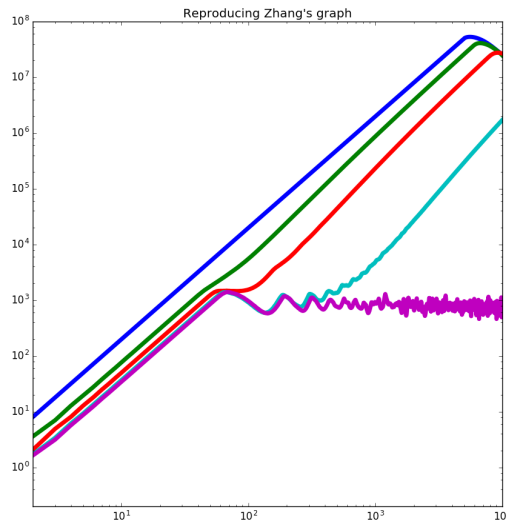


図 9 波束の分散の時間推移

特徴として分散が t^α ($\alpha > 2$) に比例して増加する現象が見られた。これは Zhang が指摘した Heyperdiffusion

の特徴と一致している。長時間での分散の値が増加していないのは境界に達した波動関数が反射した影響だと考えられる。

同じ縮尺のグラフをコンピュータ上で重ねたところ、境界条件の影響を除けば、Zhang の作成したグラフの線の上に QSB のグラフの線が重なることを定性的に確認できた。

また、最小二乗法フイッティングにより、Zhang の報告している α と一致する値が定量的に得られた。各パターンでの α を以下に示す。

V	QSB	Zang
0.0	2.00	2
1.0	2.23	2.2
1.5	2.39	2.4
1.9	2.58	2.6

6 条件設定

数値計算におけるパラメータを以下のように設定して QSB による計算を行った。

空間の離散化のパラメータは $x \in [-60, 60]$ で、 $\Delta x = 0.05$ のメッシュに自由端境界条件を採用した。時間の離散化パラメータは $t \in [0, 30]$ で $\Delta t = 0.05$ とした。

初期条件は Gauss 型波束 $\text{std}(x_0, \sigma, x)\exp[ikx]$ を使用した。パラメータは平均値 $x_0 = -15.00$ 標準偏差 $\sigma = 6.00$ 平均波数 $k = 2.00$ に設定した。

周期ポテンシャルは以下のように設定した。

$$V(x) = \begin{cases} 0 & (x < 0) \\ V_{KP}(x) & (x > 0) \end{cases} \quad (6.8)$$

$V_{KP}(x)$ には 3 つのパラメータ a, b, V_0 があるため、本研究で使用するポテンシャルもこれらの 3 つのパラメータによって完全に決定される。本研究の計算パターンはポテンシャルにのみ依存するため、これらの 3 つのパラメータのパターンによって各計算パターンは特徴づけられる。

6.1 物理的状況の考察

上記の設定は $x < 0$ の真空領域から、 $x > 0$ の周期ポテンシャル領域に平均波数 2.00 で入射する波束を想定している。

この平均波数は真空において 2.00 単位エネルギーを持つ状態で、丸め誤差の範囲内で de Broglie 波長 π に対応している。

$V_0 = 0$ の状況では $t = 30$ の時点で波束中心は $x = 45.00$ にあり、約片側 2σ が計算範囲内に入る。

$v_0 > 0$ の状況では真空の状況よりも波束の伝搬が遅くなるため、一般に $x = 60$ の境界で反射する粒子は 2.5% を上回らない。

また、境界で反射した粒子がもう一度 $x < 0$ の領域に侵入する確率は 10^{-10} を下回ると推定される。

7 結果

7.1 $x-t$ パターン

波束の時間発展を $x-t$ パターンによって図示した。 $x-t$ パターンは各時刻における各位置の存在確率を色で表したもので、古典力学で用いられる $x-t$ グラフの拡張として捉えることができる。軌跡の傾きは古典的な速度に対応するが、量子力学では量子が確率的な広がりを持つため、軌跡を大まかにしか見取ることとはできない。

縦軸は時刻、横軸は位置を表している。紫の領域の確率密度が最も高く、青緑黄赤の順に低くなっていく。半透明の水色で塗られた領域は壁を意味するが、色は全て同じで、ポテンシャルの高さとは対応していない。

以下に挙げられる特徴的なパターンを図示する。

- 全反射
- 高次反射波
- 高次進行波
- 分裂状態
- 屈折状態
- トラップ状態
- 停滞状態

7.2 全反射

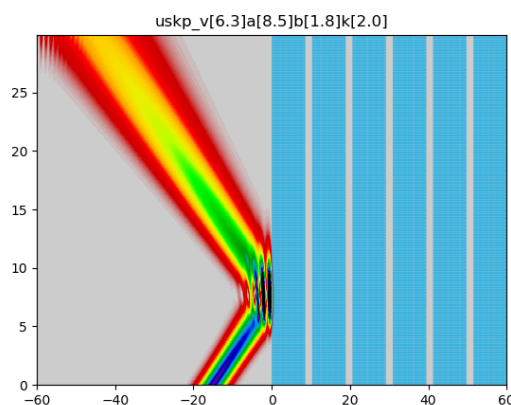


図 10 全反射の例 $V_0=6.3$, $a=8.5$, $b=1.8$ の場合

壁が厚すぎたり、ポテンシャルが高すぎたりする場合にみられる。ほとんどの波が反射する。

7.3 高次反射波

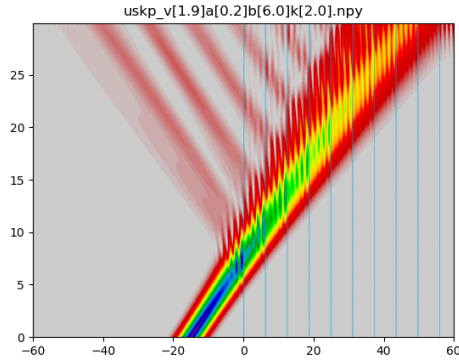


図 11 高次反射波の例:壁が薄めのもの $v_0=1.0$, $a=2.7$, $b=5.8$

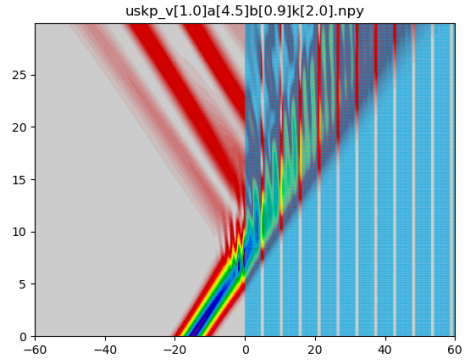


図 12 高次反射波の例:壁が厚めのもの。停滞状態の影響がみられる $v_0=1.0$, $a=4.5$, $b=0.9$

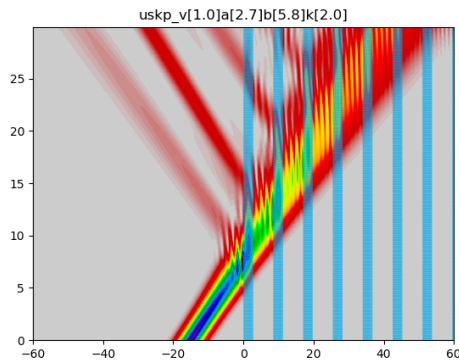


図 13 高次反射波の例:2 次の反射波が卓越している $v_0=1.9$, $a=0.2$, $b=6.0$

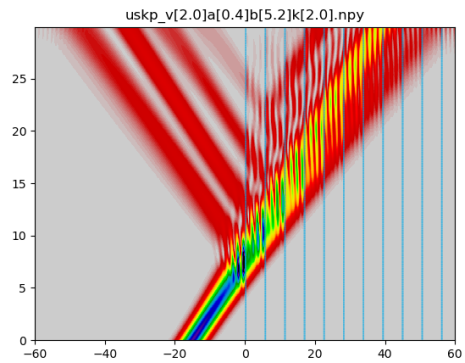


図 14 高次反射波の例:2 次進行波も同時に見られる珍しい例 $v_0=2.0$, $a=0.4$, $b=5.2$

ポテンシャルが 2.0 よりも低く、壁が井戸に比べて薄い場合によくみられる。ポテンシャル表層だけでなく 2 層目や 3 層目でも反射波束が現れる。ほとんどが 2 回以上反射しない。

繰り返しをさけるため、 n 個目の反射波のことを n 次反射波と呼ぶことにする。同様に n 個目の進行波のことを n 次進行波と呼ぶことにする。

高次反射波が表れる場合では 1 次反射波よりも 2 次反射波のほうが卓越している場合がみられる。図 11 以外の例ではポテンシャル 1 層目内部で反射した 2 次反射波が最も卓越している。

これは 1 次反射波と 2 次反射波の反射プロセスが違うためだと考えられる。まず、すべての例で 1 次反射波は真空-壁の境界付近で反射が起こっている。一方、図 12 ではポテンシャル 1 層目で停滞する波が壁-井戸の境界で反射し、再び真空へ侵入したものが 2 次反射波となる様子が観察できる。

二次反射波のほうが卓越しているパターンは壁が厚いものによくみられる。逆に壁が薄い図 11 では 1 次反射波と 2 次反射波との間に大きな違いは見られない。

7.4 高次進行波

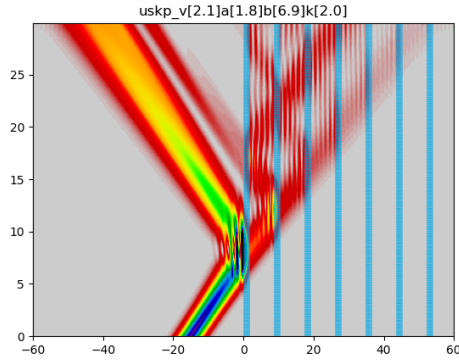


図 15 高次進行波の例:典型的なもの $V_0=2.1$, $a=1.8$, $b=6.9$

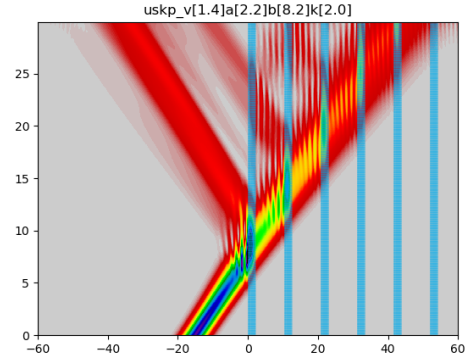


図 16 高次進行波の例:ひし形の模様がかなり大きく見える $V_0=1.4$, $a=2.2$, $b=8.2$

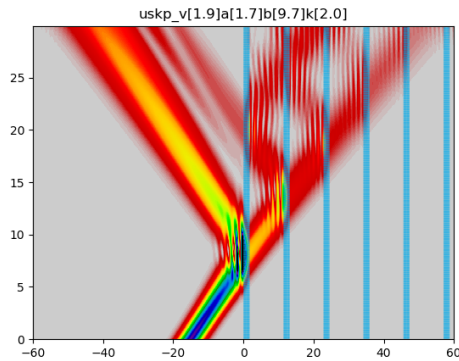


図 17 高次進行波の例:2 次進行波が次第に卓越する様子が見られる $V_0=1.9$, $a=1.7$, $b=9.7$

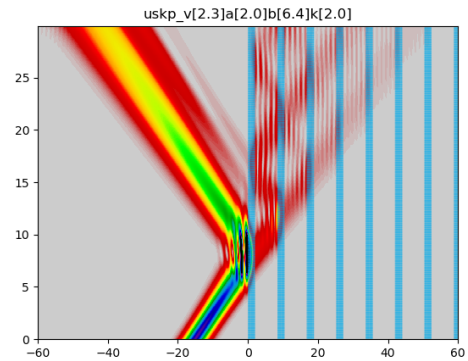


図 18 高次進行波の例:3 次進行波も見られる珍しい例 $V_0=2.3$, $a=2.0$, $b=6.4$

$V_0=2$ 前後、 $a=2$ 前後、 b が a の数倍のときにみられる。二層目以降の壁の左側で反射した波束がひとつ前の層の右側でもう一度反射して、はじめと同じ進行方向へ伝搬する。

図 15 が最も分かりやすいが、1 次進行波は時間とともに減衰し、2 次進行波やそれ以降の波束が卓越していく。1 次進行波が減衰する理由は、全ての壁を透過する波が時間とともに減少する一方で、新しく 1 次進行波に加わる波束が存在しないことがあげられる。一方で、2 次進行波は 1 次進行波から 2 回反射してきた波束が時間とともに加わってくる影響と、2 次進行波自身が壁で反射し減少する影響とがせめぎあうため、単調に減衰するとはいえない。図 15 では 20 単位時間以降で 2 次進行波が卓越している。

進行波同士の時間的間隔は井戸の幅に影響を受ける。井戸の幅が広いと、反射から反射までの時間的間隔が大きくなるため、進行波同士の間隔も大きくなる。しかし、波束がポテンシャル内で停滞する影響で、反射そのものにかかる時間が存在する [13, 15] ため、一概に井戸の幅だけに時間的間隔が依存しているとは言い切れない。実際に図 16 と図 17 を比較すると図 17 のほうが井戸が広い。しかし、図 17 が 3 次反射波まで観察できる一方、図 16 では 2 次進行波までしか観察できない。これは単位時間当たりの反射回数が

図 17 のほうが多いことを意味している。ポテンシャルが厚くなると波束がポテンシャルに滞在する時間が伸びることの影響が表れていると考えられる。

7.5 分裂状態

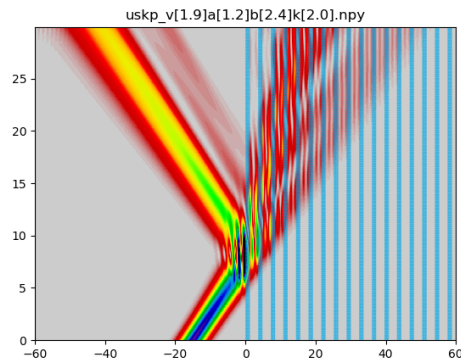


図 19 ステップポテンシャル

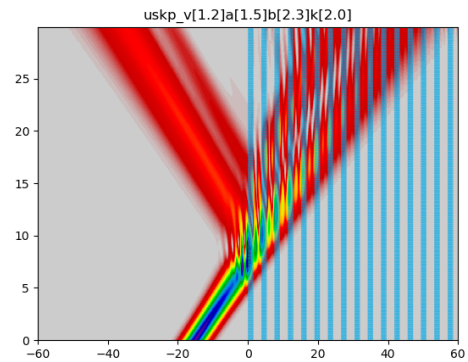


図 20 箱型ポテンシャル

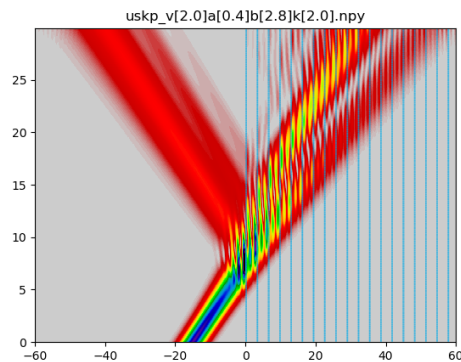


図 21 ポテンシャルの足し合わせ

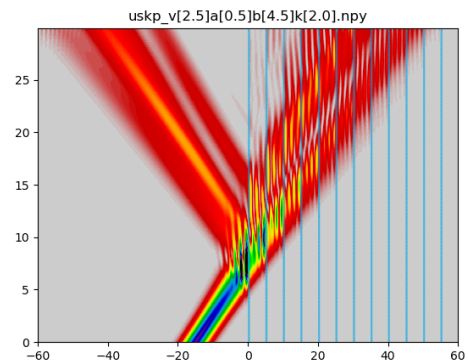


図 22 二次関数

井戸に対して壁の厚さが同程度の大きさで、周期が5を超えない程度の場合によくみられる。ポテンシャルが高すぎるとみられなくなる。入射波と反射波が重なり合っているため、1次、2次といった進行波の区別は付けられないが、時刻で区切った際に波束が2つ以上の山に分かれる。

7.6 屈折状態

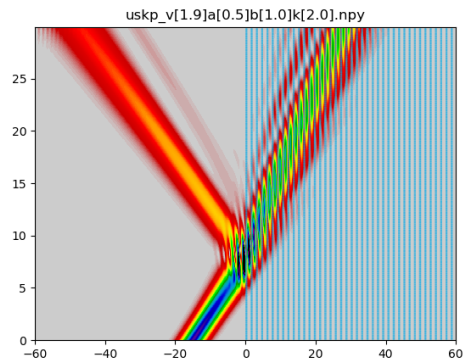


図 23 ステップポテンシャル

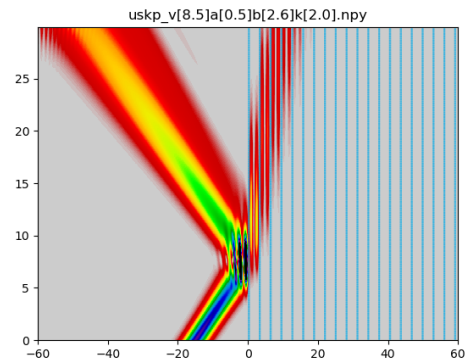


図 24 箱型ポテンシャル

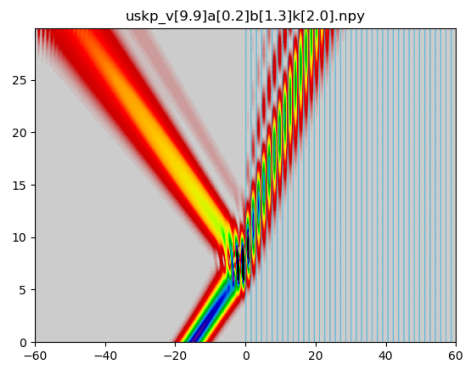


図 25 ポテンシャルの足し合わせ

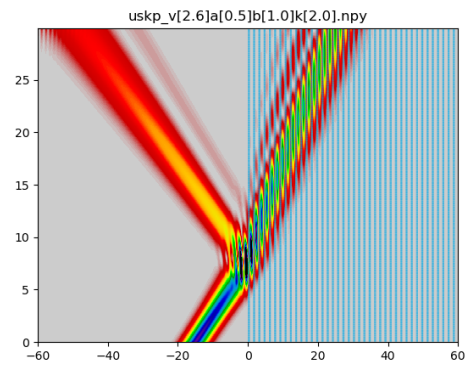


図 26 二次関数

周期が 1.5, 3.2 などの場合にみられる。ポテンシャルの高さには関係せず、 $v=1.9$ から $v=10.0$ の場合にまでみられた。波束がほぼ完全に一体となってポテンシャル中に入射している。

7.7 トラップ状態

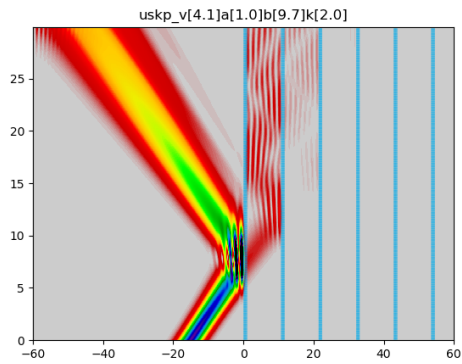


図 27 ステップポテンシャル

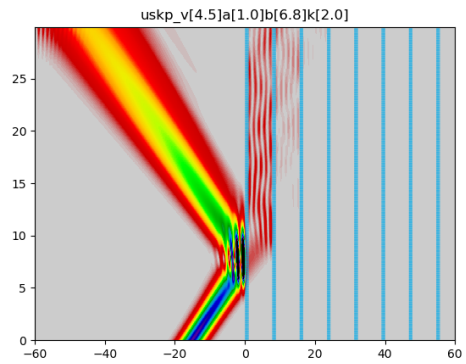


図 28 箱型ポテンシャル

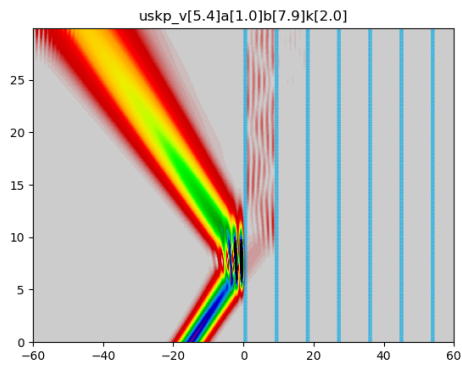


図 29 ポテンシャルの足し合わせ

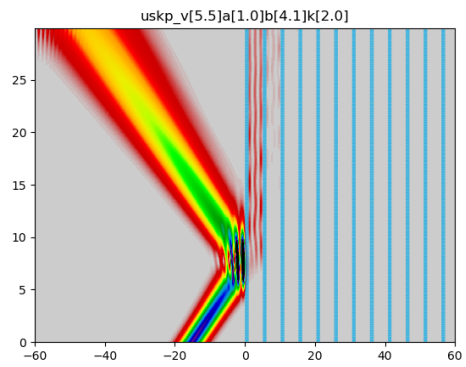


図 30 二次関数

ポテンシャルが2よりも高く、井戸と比較して壁が薄い場合にみられる。1層と2層の間に波束が入り、出られなくなる。

7.8 停滞状態

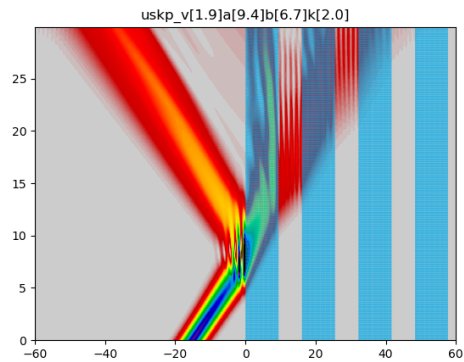


図 31 ステップポテンシャル

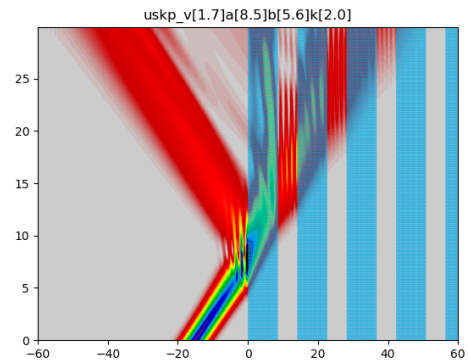


図 32 箱型ポテンシャル

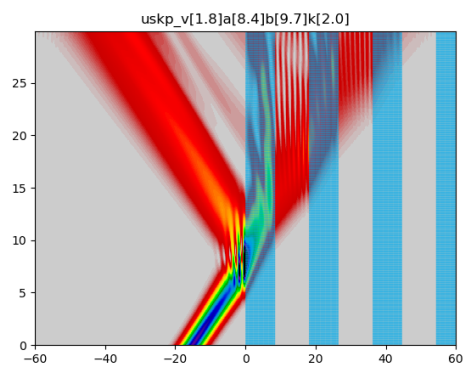


図 33 ポテンシャルの足し合わせ

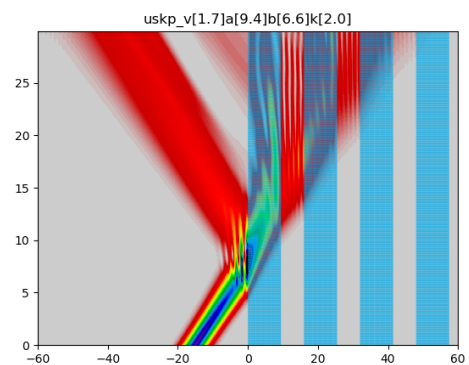


図 34 二次関数

ポテンシャルが2より低く、壁が5よりも厚い場合によくみられる。例の一層目が顕著だが、ポテンシャル内にとどまり続ける波が存在する。ポテンシャルにとどまる波はポテンシャル表面で反射する場合よりも時間をかけて反射していく。これは、ポテンシャル内の伝搬速度がほぼ0になることで起こる現象だと考えられる。

7.9 反射率と壁の厚さの関係

$t=30$ の時点で真空領域に滞在している確率を波束の反射率として様々なパターンの反射率を計算した。結果を以下に示す。

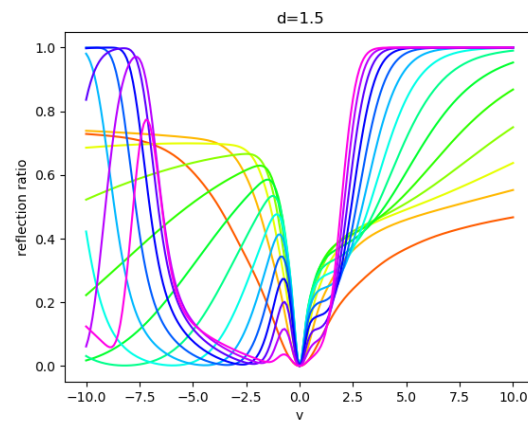


図 35 周期 1.5 のポテンシャルの反射率

曲線の色は 1 周期における壁の割合を示している。赤から紫にわたって虹のスペクトルと同じ順番で壁の割合が高くなっている。

注目すべきは $V=1.9$ 周辺で、反射率が最も高いものが緑色（壁の割合が 50% 程度）であり、赤色（壁の割合が 3%）や紫（壁の割合が 97%）はそれよりも低い。

古典的な発想ではポテンシャルが高いほど反射率も高くなりそうだが、そのようにはならない。

紫の反射率が低くなる理由は、ポテンシャルの高さが 2.0 程度と伝搬速度がほぼ 0 になる場合と近いことだと考えられる。停滞状態のパターンのようにポテンシャル内から出てこなくなっている粒子が一定数存在すると、有限時間で反射率は低く見積もられる。

ポテンシャル領域の滞在時間が高くなることの間接的証拠として、箱型ポテンシャルの場合を示す。箱型ポテンシャルの定常状態における真空領域とポテンシャル領域の確率密度の比は以下ようになる。

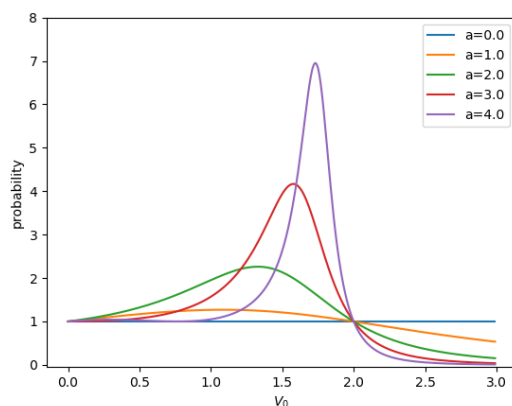


図 36 真空領域の滞在確率/ポテンシャル領域の滞在確率

横軸はポテンシャルの高さ、縦軸は真空領域の確率密度/ポテンシャル領域の確率密度を表している。縦軸が 1 を超えた場合、粒子が見いだされる確率密度はポテンシャル領域のほうが高いことを意味している。

系のエネルギーはシミュレーションと同様に 2 に設定している。

$V_0 < 2$ の領域ではポテンシャル内のほうが確率密度が高くなっていることがわかる。また、 V_0 が 2 よりも少しだけ小さいときに現れるピークは a が大きくなるほど鋭くなることがわかる。

よって、入射エネルギーよりも少しだけ低いポテンシャルでは、 a の値が大きくなるごとに滞在時間が増えていくことがいえる。停滞状態のパターンから、Kronig-Penny ポテンシャルでも同様の傾向があると考えられる。

付録A Blochの定理の簡単な証明

ハミルトニアン H に以下の条件を課す。 T_d は移動距離 d の並進演算子。

$$[T_d, H] = 0 \quad (\text{付録 A.9})$$

これは並進対称性を表している。これにより、固有状態 ϕ は d の整数倍の周期を持ち、 H と T_d の同時固有状態になる。並進演算子の固有状態は 1 の整数乗根になる。

$$\begin{aligned} T_d \phi &= A \phi \\ (T_d)^n \phi &= A^n \phi = \phi \\ A &= e^{i2\pi/n} = e^{iKd} \\ K &= \frac{2\pi}{nd} \end{aligned} \quad (\text{付録 A.10})$$

ϕ の周期を nd とした。ここで以下の関数 u を考える。

$$u(x) = e^{iKx} \phi(x) \quad (\text{付録 A.11})$$

$$\begin{aligned} T_d u(x) &= e^{iK(x-d)} e^{iKd} \phi(x) \\ T_d u(x) &= e^{iKx} \phi(x) = u(x) \end{aligned} \quad (\text{付録 A.12})$$

よって $u(x)$ は周期 d を持つ周期関数だということが分かる。
 $u(x)$ の定義を変形すれば Bloch の定理を証明したことになる。

$$\phi(x) = e^{-iKx} u(x) \quad (\text{付録 A.13})$$

K を三次元に拡張したものは逆格子ベクトルと呼ばれる。

付録B Kronig-Penny モデルのエネルギーバンド構造の簡単な計算方法

井戸と壁の Schrödinger 方程式は単純な単振動になるため、エネルギーを ε としたとき以下のような解になる。

$$\begin{aligned} \phi_{well}(x) &= W^+ e^{+ikx} + W^- e^{-ikx} \\ \phi_{barrier}(x) &= B^+ e^{+iqx} + B^- e^{-iqx} \\ k &= \sqrt{2m\varepsilon} \\ q &= \sqrt{2m(\varepsilon - V_0)} \end{aligned} \quad (\text{付録 B.14})$$

W^+, W^-, B^+, B^- は振幅を表す定数。

$\phi(x)$ に一回微分までの連続性を課すと、井戸と壁の境界で以下の条件が必要となる。

$$\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} W^+ \\ W^- \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ \rho & -\rho \end{pmatrix} \begin{pmatrix} B^+ \\ B^- \end{pmatrix} \quad (\text{付録 B.15})$$

$$\begin{pmatrix} e^{ika} & e^{-ika} \\ e^{ika} & -e^{-ika} \end{pmatrix} \begin{pmatrix} W^+ \\ W^- \end{pmatrix} = \begin{pmatrix} e^{iqa} & e^{-iqa} \\ \rho e^{iqa} & -\rho e^{-iqa} \end{pmatrix} \begin{pmatrix} B^+ \\ B^- \end{pmatrix} \quad (\text{付録 B.16})$$

ここで $\rho = q/k_0$ 。

Bloch の定理を上式の左辺に使うと以下の置き換えをすることが出来る。

$$e^{ika} \rightarrow e^{ikb} e^{-iKd} \quad (\text{付録 B.17})$$

$$e^{-ika} \rightarrow e^{-ikb} e^{-iKd} \quad (\text{付録 B.18})$$

簡略のために以下の置き換えをする。

$$\begin{aligned} H &= \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \\ \alpha &= \begin{pmatrix} 1 & 0 \\ 0 & \rho \end{pmatrix} \\ K_x &= \begin{pmatrix} e^{ikx} & 0 \\ 0 & e^{-ikx} \end{pmatrix} \\ Q_x &= \begin{pmatrix} e^{iqx} & 0 \\ 0 & e^{-iqx} \end{pmatrix} \\ \mathbf{W} &= \begin{pmatrix} W^+ \\ W^- \end{pmatrix} \\ \mathbf{B} &= \begin{pmatrix} B^+ \\ B^- \end{pmatrix} \end{aligned} \quad (\text{付録 B.19})$$

ちなみに、H は Hadamard 行列として知られている。置き換えをすると以下のような式が得られる。

$$\begin{aligned} H\mathbf{W} - H\alpha\mathbf{B} &= 0 \\ e^{-iKd} K_b H\mathbf{W} - Q_a H\alpha\mathbf{B} &= 0 \end{aligned} \quad (\text{付録 B.20})$$

整理すると以下のような方程式が得られる。

$$\begin{pmatrix} E & \alpha \\ E & e^{iKd} H^{-1} K_b^{-1} Q_a H \alpha \end{pmatrix} \begin{pmatrix} \mathbf{W} \\ \mathbf{B} \end{pmatrix} = 0 \quad (\text{付録 B.21})$$

ここで E は 2 次の単位行列。トリビアルでない解は行列部分が 0 となるときに現れる。4 行 4 列の行列式を求める手続きは煩雑だが、余因子展開を繰り返せば必ず展開できる。幸い、今回は単位行列が現れることを利用すれば、実質的に現れる項は 4 項だけとなる。

付録 C 定常状態の計算

数値計算で用いたポテンシャルにおける定常解は転送行列によって扱うことが出来る。

$$V(x) = \begin{cases} 0 & (x < 0) \\ V_{KP}(x) & (x > 0) \end{cases} \quad (\text{付録 C.22})$$

Schrödinger 方程式の解は Kronig-Penny モデルの計算と同じ形になる。連続の条件を課すと、以下の境界条件が成立する。

$$\begin{aligned} K_{dn} H \mathbf{W}_n &= Q_{dn} H \alpha \mathbf{B}_n \\ K_a K_{dn} H \mathbf{W}_{n+1} &= Q_a Q_{dn} H \alpha \mathbf{B}_n \end{aligned} \quad (\text{付録 C.23})$$

前章で用いた置き換えを使った。 W_n, b_n は 0 から数えて原点から n 周期目の井戸、壁における振幅を表している。

上式から井戸の振幅についての漸化式を作ると以下のようなになる。

$$\mathbf{W}_{n+1} = H^{-1} K_{dn}^{-1} K_a^{-1} Q_a K_{dn} H \mathbf{W}_n \quad (\text{付録 C.24})$$

ちなみに、一般項は以下のようなになる。

$$\mathbf{W}_n = H^{-1} (K_b Q_a)^n H \mathbf{W}_0 \quad (\text{付録 C.25})$$

初項 W_0 を漸化式の係数部分 $A = H^{-1} K_{dn}^{-1} K_a^{-1} Q_a K_{dn} H$ の固有ベクトル A_1, A_2 で展開する。ここで、固有値 a_1, a_2 には $|a_1| < |a_2|$ の関係があることにする。

$$\begin{aligned} \mathbf{W}_n &= c_1 A_1 + c_2 A_2 \\ W_0 &= A^{-n} \mathbf{W}_n = c_1 a_1^{-n} A_1 + c_2 a_2^{-n} A_2 \end{aligned} \quad (\text{付録 C.26})$$

ここで、 $n \rightarrow \infty$ の場合を考えると、 $|a_1|^{-n} \gg |a_2|^{-n}$ となり、 W_0 の主成分は A_1 となる。

A_1 における進行波と後退波の比をとれば、反射率を得ることが出来る。

付録 D QSB のソースコード

QSB ソースコードはいくつかのテストコードとともにインターネット上でも MIT ライセンスで公開されている [17]。

Listing 6 __init__.py

```
1 from QuantumSketchBook.context import Context
2 from QuantumSketchBook.state import State, gaussian_state
3 from QuantumSketchBook.potential import Potential, potential, free, box, kp, step, vacuum_kp,
   kp_vacuum
4 from QuantumSketchBook.hamiltonian import Hamiltonian
```

```

5 from QuantumSketchBook.schroedinger import Schroedinger
6
7
8 def plot(plotted, show=True, save=False, title="no_title", *args, **kwargs):
9     return plotted._plot_(show, save, title, *args, **kwargs)

```

Listing 7 mesh.py

```

1 from scipy import arange
2 from numbers import Real
3 import QuantumSketchBook.meta as meta
4
5
6 class Mesh(meta.Mesh):
7
8     def __new__(cls, x_min: Real, x_max: Real, dx: Real,
9                 t_min: Real, t_max: Real, dt: Real)-> "Mesh":
10         if not all(isinstance(x, Real) for x in (x_min, x_max, dx, t_min, t_max, dt)):
11             raise TypeError("input should be a number")
12         if not (x_min < x_max and t_min < t_max):
13             raise ValueError("The min should be smaller than The max.")
14         if dx <= 0 or dt <= 0:
15             raise ValueError("The dx and dt should be positive.")
16         if not (x_max - x_min > dx and t_max - t_min > dt):
17             raise ValueError("too big dx or dt to make mesh.")
18         return super().__new__(cls, x_min, x_max, dx, t_min, t_max, dt)
19
20     @property
21     def param(self):
22         return self.x_min, self.x_max, self.dx, self.t_min, self.t_max, self.dt
23
24     @property
25     def x_vector(self):
26         return arange(self.x_min, self.x_max, self.dx)
27
28     @property
29     def t_vector(self):
30         return arange(self.t_min, self.t_max, self.dt)
31
32     @property
33     def x_num(self):
34         return len(self.x_vector)
35
36     @property
37     def t_num(self):
38         return len(self.t_vector)
39
40     def __str__(self):
41         x_string = "{}<x<{} (dx={})".format(self.x_min, self.x_max, self.dx)
42         t_string = "{}<t<{} (dt={})".format(self.t_min, self.t_max, self.dt)

```

```

43         return "\n".join((x_string, t_string))
44
45     def __eq__(self, other):
46         return self.param == other.param
47
48
49 def my_mesh(x_min=-10, x_max=10, dx=0.1, t_min=0, t_max=10, dt=0.1):
50     return Mesh(x_min, x_max, dx, t_min, t_max, dt)
51
52
53 if __name__ == "__main__":
54     test1 = my_mesh()
55     assert test1.x_num == 200
56     assert test1.t_num == 100

```

Listing 8 context.py

```

1  from QuantumSketchBook.mesh import Mesh
2
3
4  class MeshContextError(ValueError):
5      pass
6
7
8  class MeshContext:
9
10     _has_mesh = False
11     _mesh = None
12     _context = None
13     _observer = list()
14     _in_context = False
15
16     def __new__(cls, mesh=None):
17         if not cls._has_mesh:
18             if cls._in_context:
19                 raise MeshContextError("MeshContext should not be overwriten in 'with_block'.")
20
21             if isinstance(mesh, Mesh):
22                 cls._has_mesh = True
23                 cls.update_mesh(mesh)
24                 cls._context = super().__new__(cls)
25             else:
26                 raise MeshContextError("invalid argument {} should {}".format(mesh.__class__,
27                                         Mesh.__class__))
28         return cls._context
29
30     @classmethod
31     def get_mesh(cls):
32         if cls._mesh is not None:
33             return cls._mesh

```

```

33         else:
34             raise MeshContextError("not_found_any_mesh_in_Context")
35
36     @classmethod
37     def update_mesh(cls, mesh):
38         if isinstance(mesh, Mesh):
39             cls._mesh = mesh
40         for osv in cls._observer:
41             osv.update_mesh()
42         return cls()
43
44     @classmethod
45     def has_instance(cls):
46         return cls._has_mesh
47
48     @classmethod
49     def clean(cls):
50         cls._has_mesh = False
51         cls._mesh = None
52         cls._context = None
53         cls._observer = list()
54
55     @classmethod
56     def add_observer(cls, observer):
57         if hasattr(observer, "update_mesh"):
58             cls._observer.append(observer)
59         else:
60             raise ValueError("observer_should_have_the_update_mesh().")
61
62     @classmethod
63     def remove_observer(cls, observer):
64         if observer in cls._observer:
65             cls._observer.remove(observer)
66         else:
67             raise ValueError("{}is_not_a_observer_of_MeshContext.".format(observer))
68
69     @classmethod
70     def is_observer(cls, item):
71         return item in cls._observer
72
73     @classmethod
74     def set_context_on(cls):
75         cls._in_context = True
76
77     @classmethod
78     def set_context_off(cls):
79         cls._in_context = False
80
81

```

```

82 class Context:
83
84     def __init__(self, x_min, x_max, dx, t_min, t_max, dt):
85         self.mesh = Mesh(x_min, x_max, dx, t_min, t_max, dt)
86         self._pre_mesh = None
87         self._pre_observer = list()
88
89     def __enter__(self):
90         if MeshContext.has_instance():
91             self._pre_mesh = MeshContext.get_mesh()
92             MeshContext(self.mesh)
93             MeshContext.set_context_on()
94             return self.mesh
95
96     def __exit__(self, exc_type, vallue, traceback):
97         MeshContext().set_context_off()
98         MeshContext.clean()
99         if self._pre_mesh is not None:
100             MeshContext(self._pre_mesh)
101
102
103 if __name__ == "__main__":
104     mesh1 = Mesh(-10, 10, 1, 0, 10, 1)
105     mesh2 = Mesh(0, 10, 1, 0, 10, 1)
106     a = MeshContext(mesh1)
107     assert a.has_instance()
108     assert a.get_mesh() == mesh1
109
110     b = MeshContext()
111     assert a.get_mesh() == b.get_mesh()
112     assert a is b
113
114     c = MeshContext.update_mesh(mesh2)
115     assert a.get_mesh() == c.get_mesh()
116     assert a is c
117
118     MeshContext.clean()
119     assert not a.has_instance()
120
121     try:
122         a.get_mesh()
123         error = False
124     except MeshContextError:
125         error = True
126     assert error
127
128     def error_check(*args):
129         try:
130             MeshContext(*args)

```

```

131         except MeshContextError:
132             return True
133         return False
134
135     assert error_check()
136     assert error_check(3)
137     assert not error_check(mesh2)
138     MeshContext().clean()
139     with Context(-10, 0, 1, -10, 0, 1) as mm:
140         assert mm == Mesh(-10, 0, 1, -10, 0, 1)
141
142     try:
143         MeshContext.get_mesh()
144     except MeshContextError as e:
145         assert e.args[0] == "not_found_any_mesh_in_Context"

```

Listing 9 field.py

```

1 from QuantumSketchBook.quantized import Quantized
2 from scipy import array
3 from typing import TYPE_CHECKING, Optional
4 from matplotlib.pyplot import figure, axes
5 if TYPE_CHECKING:
6     from QuantumSketchBook import Mesh
7
8
9 class Field(Quantized):
10
11     def __init__(self, arg, mesh: Optional["Mesh"]=None):
12         super().__init__(mesh=mesh)
13         if hasattr(arg, "__iter__"):
14             vec = array(tuple(arg))
15             if not vec.size == self.mesh.x_num:
16                 raise ValueError("Length_of_input_iterable_must_be_equal_to_mesh.num")
17             self.vector = vec
18         elif callable(arg):
19             self.vector = arg(self.mesh.x_vector)
20         else:
21             raise TypeError("the_first_argument_should_be_iterable_or_callable")
22
23     def __plot__(self, show, save, title, ylabel, values, *args, **kwargs):
24         fig = figure()
25         ax = axes()
26         ax.plot(self.mesh.x_vector, values)
27         ax.set_title(title)
28         ax.set_xlabel("x")
29         ax.set_ylabel(ylabel)
30         fig.add_axes(ax)
31
32         if save:

```



```

33         fig.savefig(title + ".png")
34
35     if show:
36         fig.show()
37     return fig
38
39
40 if __name__ == "__main__":
41     import QuantumSketchBook as QSB
42     from scipy import ndarray
43     with QSB.Context(0, 10, 1, 0, 10, 1):
44         assert Field(range(10)).vector.__class__ == ndarray
45         assert Field(lambda x: x).vector.__class__ == ndarray

```

Listing 10 potential.py

```

1 from QuantumSketchBook.field import Field
2 from QuantumSketchBook.context import MeshContext
3 from scipy.sparse import dia_matrix
4 from scipy import array
5 from numbers import Real
6 from math import ceil, floor
7
8
9 class Potential(Field):
10
11     def matrix(self):
12         offset = [0]
13         n = self.mesh.x_num
14         mat = dia_matrix((self.vector, offset), shape=(n, n), dtype=complex)
15         return mat.tocsr()
16
17     def __add__(self, other):
18         if not self.mesh == other.mesh:
19             raise ValueError("meshes on {} and {} are not the same.".format(
20                 self.mesh, other.mesh))
21         return Potential(self.vector + other.vector, mesh=self.mesh)
22
23     def __mul__(self, other):
24         if not isinstance(other, Real):
25             raise TypeError("unsupported operand type(s) for *: 'Potential' and '{}'".format(
26                 type(other)))
27         return Potential(other * self.vector, mesh=self.mesh)
28
29     def __sub__(self, other):
30         return self.__add__(other.__mul__(-1))
31
32     def __rmul__(self, other):
33         return self.__mul__(other)
34
35     def __eq__(self, other):

```

```

34         return self.mesh == other.mesh and self.vector == other.vector
35
36     def __plot__(self, show, save, title, *args, **kwargs):
37         return super().__plot__(show, save, title, "V", self.vector)
38
39
40     def potential(arg):
41         return Potential(arg)
42
43
44     def free():
45         return potential(lambda x: 0)
46
47
48     def step(height: Real, distance: Real):
49         if not all(isinstance(x, Real) for x in (height, distance)):
50             raise TypeError
51         v = array([height if distance <= x else 0 for x in MeshContext.get_mesh().x_vector])
52         return potential(v)
53
54
55     def box(height: Real, distance: Real, barrier: Real):
56         if 0 >= barrier:
57             raise ValueError("barrier should be positive.")
58         near = step(height, distance)
59         far = step(height, distance + barrier)
60         return near - far
61
62
63     def vacuum_kp(height, well, barrier):
64         x_max = MeshContext.get_mesh().x_max
65         period = well + barrier
66         cycle = ceil(x_max / period) if x_max > 0 else 1
67         gen = (box(height, i * period, barrier) for i in range(cycle))
68         return sum(gen, free())
69
70
71     def kp_vacuum(height, well, barrier):
72         x_min = MeshContext.get_mesh().x_min
73         period = well + barrier
74         cycle = -floor(x_min / period) if x_min < 0 else 1
75         gen = (box(height, -(i + 1) * period, barrier) for i in range(cycle))
76         return sum(gen, free())
77
78
79     def kp(height, well, barrier):
80         v1 = vacuum_kp(height, well, barrier)
81         v2 = kp_vacuum(height, well, barrier)
82         return v1 + v2

```

```

83
84
85 if __name__ == "__main__":
86     import QuantumSketchBook as QSB
87     with QSB.Context(0, 10, 1, 0, 10, 1) as mesh:
88         test1 = step(3, 1).vector
89         test2 = step(2, 2).vector
90         test3 = vacuum_kp(1, 1, 1).vector
91         assert all(a == 0 for i, a in enumerate(test1) if i < 1)
92         assert all(a == 3 for i, a in enumerate(test1) if i >= 1)
93         assert all(b == 0 for i, b in enumerate(test2) if i < 2)
94         assert all(b == 2 for i, b in enumerate(test2) if i >= 2)
95         assert all(c == 0 for i, c in enumerate(test3) if i % 2 == 1)
96         assert all(c == 1 for i, c in enumerate(test3) if i % 2 == 0)
97         QSB.plot(step(3, 1), False, True)

```

Listing 11 stste.py

```

1 from QuantumSketchBook.field import Field
2 from QuantumSketchBook.context import MeshContext
3 from scipy.stats import norm
4 from scipy import array, exp, absolute, ones, random
5
6
7 class State(Field):
8
9     def random_values(self, n):
10         probability = absolute(self.vector) ** 2
11         probability = probability * 10000 # 有効数字を3ケタ以上取るために10000倍する
12         probability_naturalized = probability.astype(int)
13         target = []
14         for i in range(self.mesh.x_num):
15             weight = probability_naturalized[i]
16             if not weight == 0:
17                 weight_array = i * ones(weight, dtype=int)
18                 target.extend(weight_array.tolist())
19         max_random = len(target)
20         target_index = random.randint(0, max_random, n)
21         random_index = array(target)[target_index]
22         return self.mesh.x_vector[random_index]
23
24     def __plot__(self, show, save, title, *args, **kwargs):
25         super().__plot__(show, save, title, "probability", absolute(self.vector) ** 2)
26
27
28 def gaussian_state(mean, sd, wave_number):
29     x = MeshContext.get_mesh().x_vector
30     pdf = norm.pdf(x, mean, sd / 2)
31     wav = exp(1j * wave_number * x)
32     return State(array(pdf * wav, dtype=complex))

```

```

33
34
35 if __name__ == "__main__":
36     import QuantumSketchBook as QSB
37     with QSB.Context(0, 10, 1, 0, 10, 1):
38         test1 = absolute(gaussian_state(5, 1, 0).vector)
39         assert max(test1) == test1[5]
40         assert test1[4] == test1[6]
41         QSB.plot(gaussian_state(0, 3, 1), False, True)

```

Listing 12 schroedinger.py

```

1 from QuantumSketchBook.state import State
2 from QuantumSketchBook.nelson import Nelson
3 from scipy import zeros, meshgrid, absolute
4 from scipy.integrate import ode
5 from matplotlib.pyplot import figure
6
7
8 class Schroedinger:
9
10     def __init__(self, hamiltonian, x0state: State):
11         self.mesh = hamiltonian.mesh
12         self.potential = hamiltonian.potential
13         self._operator = -1j * hamiltonian.matrix
14         self.x0state = x0state
15         self._sol = None
16
17         self.ode = ode(self.equation)
18         self.ode.set_integrator('zvode', nsteps=1000000)
19         self.ode.set_initial_value(self.x0state.vector)
20
21     def equation(self, t, phi0): # noqa
22         return self._operator.dot(phi0)
23
24     def __iter__(self):
25         yield self.x0state.vector
26         index = 1
27         print("now solving", end="_")
28         while self.ode.successful() and index < self.mesh.t_num:
29             fin = (index + 1) / self.mesh.t_num
30             print('\rSchrodinger have solved {:.3.2%}!_'.format(fin), end='_ ' if not fin == 1 else
31                   "\n", flush=True)
32             yield self.ode.integrate(self.mesh.t_vector[index])
33             index += 1
34
35     def solution(self):
36         if self._sol is None:
37             self._sol = zeros([self.mesh.t_num, self.mesh.x_num], dtype=complex)
38             for i, s in enumerate(self):

```

```

38         self._sol[i] = s
39     return self._sol
40
41     def nelson(self, n: int, micro_steps=10):
42         return Nelson(self, n, micro_steps)
43
44     def __plot__(self, show=True, save=False, title="no_title", max_pix=(1000, 1000), limit=()):
45         x_step = int(self.mesh.x_num / max_pix[0]) + 1
46         t_step = int(self.mesh.x_num / max_pix[1]) + 1
47         x_grid, t_grid = meshgrid(self.mesh.x_vector[::x_step], self.mesh.t_vector[::t_step])
48         phi2 = absolute(self.solution()[1::t_step, ::x_step]) ** 2
49         potential_ = self.potential.vector
50         x = self.mesh.x_vector
51         fig = figure()
52         above = fig.add_axes((0.06, 0.25, 0.9, 0.7))
53         under = fig.add_axes((0.06, 0.1, 0.9, 0.17), sharex=above)
54
55         above.set_title(title)
56         above.tick_params(labelbottom="off")
57         above.set_ylabel("time(a.u.)")
58         under.tick_params(labelleft="off")
59         under.set_xlabel("space(a.u.)")
60
61         above.pcolormesh(x_grid, t_grid, phi2, cmap="nipy_spectral_r")
62         under.plot(x, potential_)
63
64         if limit is not ():
65             above.set_xlim(*limit)
66             under.set_xlim(*limit)
67
68         if save:
69             fig.savefig(title + ".png")
70
71         if show:
72             fig.show()
73         return fig

```

Listing 13 hamiltonian.py

```

1 from QuantumSketchBook.laplasian import Laplasian
2 from QuantumSketchBook.quantized import Quantized
3
4
5 class Hamiltonian(Quantized):
6
7     def __init__(self, potential, mass=1, boundary="free", mesh=None):
8         super().__init__(mesh=mesh)
9         self.mass = mass
10        self.potential = potential
11        lap = Laplasian(self.mesh).matrix(boundary=boundary)

```

```

12     pot = potential.matrix()
13     self.matrix = -1 / (2 * self.mass) * lap + pot

```

Listing 14 laplasian.py

```

1  from scipy import arange, array, zeros
2  from scipy.sparse import dia_matrix, csr_matrix, coo_matrix
3  from QuantumSketchBook.mesh import Mesh
4
5
6  class Laplasian:
7
8      def __init__(self, mesh: Mesh):
9          self.dx = mesh.dx
10         self.x_num = mesh.x_num
11
12     def _core_matrix(self) -> csr_matrix:
13         det = 1 / (360 * self.dx ** 2)
14         coeff = array([[4], [-54], [540], [-980], [540], [-54], [4]])
15         std = det * coeff
16         data = std.repeat(self.x_num + 1, axis=1)
17         pad = arange(-3, 3 + 1)
18         dia = dia_matrix((data, pad), shape=(self.x_num, self.x_num), dtype=complex)
19         csr = dia.tocsr()
20         return csr
21
22     def matrix(self, boundary="free") -> csr_matrix:
23         det = 1 / (360 * self.dx ** 2)
24         csr = self._core_matrix()
25
26         if boundary == "free":
27             data = csr.data
28             bound = array([-980, 1080, -108, 8, 540, -1034, 544, -54, 4, -54, 544]) * det
29             data[:11] = bound
30             data[-11:] = bound[::-1]
31             matrix = csr_matrix((data, csr.indices, csr.indptr), dtype=complex)
32             return matrix
33
34         elif boundary == "fix":
35             data = csr.data
36             bound = array([-980, 0, 0, 0, 540, -926, 536, -54, 4, -54, 536]) * det
37             data[:11] = bound
38             data[-11:] = bound[::-1]
39             matrix = csr_matrix((data, csr.indices, csr.indptr), dtype=complex)
40             return matrix
41
42         elif boundary == "period":
43             n = self.x_num
44             data = array([4, -54, 540, 4, -54, 4, 4, -54, 4, 540, -54, 4]) * det
45             row = array([0, 0, 0, 1, 1, 2, n-3, n-2, n-2, n-1, n-1, n-1])

```

```

46         col = array([n-3, n-2, n-1, n-2, n-1, n-1, 0, 0, 1, 0, 1, 2])
47         coo = coo_matrix((data, (row, col)), shape=(n, n), dtype=complex)
48         bound = coo.tocsr()
49         matrix = csr + bound
50         return matrix
51
52     elif boundary == "absorb":
53         data = csr.data
54         bound = array([-980, 1080, -108, 8, 540, -1034, 544, -54, 4, -54, 544]) * det
55         data[:11] = bound
56         data[-11:] = bound[::-1]
57         free = csr_matrix((data, csr.indices, csr.indptr), dtype=complex)
58         data2 = zeros(self.x_num, dtype=complex)
59         data2[-40:] = 2 ** -6 * (1j + 1) * arange(40) ** 2
60         data2 = data2 + data2[::-1]
61         data.transpose()
62         dia = dia_matrix((data2, zeros(1)), shape=(self.x_num, self.x_num)).tocsr()
63         return free + dia
64
65     elif boundary == "poor":
66         det = 1 / (self.dx ** 2)
67         coeff = array([[1], [-2], [1]])
68         std = det * coeff
69         data = std.repeat(self.x_num + 1, axis=1)
70         pad = arange(-1, 1 + 1)
71         dia = dia_matrix((data, pad), shape=(self.x_num, self.x_num), dtype=complex)
72         csr = dia.tocsr()
73         return csr

```

Listing 15 quantized.py

```

1  from QuantumSketchBook.context import MeshContext
2  from typing import Optional, TYPE_CHECKING
3  if TYPE_CHECKING:
4      from QuantumSketchBook import Mesh
5
6
7  class Quantized:
8      def __init__(self, mesh: Optional["Mesh"]=None):
9          if mesh is not None:
10             self.mesh: Mesh = mesh
11          else:
12             self.mesh: Mesh = MeshContext.get_mesh()
13             MeshContext.add_observer(self)
14
15      def update_mesh(self):
16          self.mesh = MeshContext.get_mesh()
17
18
19  if __name__ == "__main__":

```

```

20     mod = __import__("QuantumSketchBook.mesh")
21     mesh1 = mod.my_mesh()
22     mesh2 = mod.my_mesh(x_min=0)
23     MeshContext(mesh1)
24     a = Quantized()
25     assert a.mesh == mesh1
26
27     MeshContext.update_mesh(mesh2)
28     assert Quantized().mesh == mesh2
29     assert not a.mesh == mesh1
30     assert a.mesh == mesh2

```

参考文献

- [1] Kittel, C. (1976). Introduction to solid state physics. New York: Wiley, 1976, 5th ed., 1.
- [2] Bloch, F. (1929). Über die quantenmechanik der elektronen in kristallgittern. Zeitschrift für physik, 52(7-8), 555-600.
- [3] Kronig, R. D. L., & Penney, W. G. (1931, February). Quantum mechanics of electrons in crystal lattices. In Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences (Vol. 130, No. 814, pp. 499-513). The Royal Society.
- [4] 長岡洋介. (1985). アンダーソン局在. 日本物理学会誌, 40(7), 489-498.
- [5] Esaki, L., & Tsu, R. (1970). Superlattice and negative differential conductivity in semiconductors. I B M J RES DEVELOP, 14(1), 61-65.
- [6] Szmulowicz, F. (1997). New eigenvalue equation for the Kronig – Penney problem. American Journal of Physics, 65(10), 1009-1014.
- [7] Anker, T., Albiez, M., Eiermann, B., Taglieber, M., & Oberthaler, M. K. (2004). Linear and non-linear dynamics of matter wave packets in periodic potentials. Optics express, 12(1), 11-18.
- [8] Yamanaka, K., Ohmae, N., Ushijima, I., Takamoto, M., & Katori, H. (2015). Frequency Ratio of Hg 199 and Sr 87 Optical Lattice Clocks beyond the SI Limit. Physical review letters, 114(23), 230801.
- [9] Zhang, Z., Tong, P., Gong, J., & Li, B. (2012). Quantum hyperdiffusion in one-dimensional tight-binding lattices. Physical review letters, 108(7), 070603.
- [10] 小出昭一郎,(2008), 量子力学 I 改訂版, 裳華房, 基礎物理学選書,5A,50 版 など
- [11] Hufnagel, L., Ketzmerick, R., Kottos, T., & Geisel, T. (2001). Superballistic spreading of wave packets. Physical Review E, 64(1), 012301.
- [12] Moyer, C. A. (2004). Numerov extension of transparent boundary conditions for the Schrödinger equation in one dimension. American Journal of Physics, 72(3), 351-358.
- [13] 前田雅人 (2014). 量子波束の跳ね返しシミュレーション. 卒業論文
- [14] 保坂あゆみ, (2011). 2次元ポテンシャル障壁を透過する波束の運動と量子軌道. 卒業論文
- [15] 後藤敬佑 (2015). 確率力学に基づいた量子跳ね返し時間のシミュレーション. 卒業論文
- [16] 太箸良介 (2015). 相対論的粒子の量子跳ね返し時間. 卒業論文
- [17] https://github.com/tmaeda11235/schroedinger_solver で開発版のソースコードを公開している

- [18] Pavelich, R. L., & Marsiglio, F. (2015). The Kronig-Penney model extended to arbitrary potentials via numerical matrix mechanics. *American Journal of Physics*, 83(9), 773-781.
- [19] python org, (2018/1/19 閲覧) Python 3.6.3 ドキュメント, <https://docs.python.jp/3/index.html>
- [20] scipy org. (2018/1/19 閲覧), Scipy Lecture Notes, <http://www.scipy-lectures.org/>
- [21] Carnahan, B. & Luther, H. A., Wilkes, J. O., 藤田宏他訳 (1982). 計算機による数値計算法, 日本コンピュータ協会, コンピュータ・サイエンス研究書シリーズ / 日本コンピュータ協会編, 19