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 21 日

目次

1 概要

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た。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, Kronig-Penny モデルは単なる教育的モデルではなく, 半導体超格子 [5] のような微細構造の基礎として扱われたり, ナノリソグラフィーの基盤形状の一つとして使われたりする [6] 実用的側面も持っている。半導体超格子においてトンネル効果を実証した功績によって 1975 年に Esaki はノーベル物理学賞を受けている [8]。

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。Anderson 局在によって, 金属等の導体における不純物の影響を考察することが可能となった。Anderson はこの功績により 1977 年にノーベル物理学賞を受けている [7]。

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研究されており, 周期ポテンシャル関連分野の物理学における重要性が理解できる。

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 [12] 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 [9] 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 [9] 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 [10] 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 [11] 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 [12] ため, 真空中よりも高次の拡散が起こっている。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。ちなみに Heyperdiffusion が発生するメカニズムについては Point-source モデルによる説明 [13], Langevin 方程式系とみなすことによる説明 [27] など様々な方向からの説明が試みられている。

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。波束は古典的な質点に対応しているが, 古典的に考えて Heyperdiffusion のような状況が起こることは予測できない。

しかし, 時間はオブザーバブルではないため, 波動関数の状態から定性的な考察を行うことは難しい。その解析方法は専ら数値計算で Schrödinger 方程式を直接解くという方法が取られている [11, 14–18]。その際のアルゴリズムは Crank-Nicolson 法 [11], Numerov 法 [14], FFT 法 [17, 18] とケースバイケースで様々な方法が使われており, 過去のコード資源の再利用が効率的に行われているようではない。

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。

数値計算法の確立のために Schrödinger 方程式を解くためのフレームワークである QuantumSketchBook(QSB) [19] を作成した。QSB は高速化のために scipy, numpy といった数値計算系の python パッケージに依存しており, これらの性能を最大限引き出すことができるように最適化が施されている。また, オブジェクト指向 (Object Oriented Programing: OOP) を取り入れ, 数値計算初心者でも最適化が施されたプログラムを使えるように設計されている。

QSB を実際に使用し, 真空中から Kronig-Penny ポテンシャルに入射する電子波束の時間発展を計算した。その結果, 特徴的なパターンを 7 パターン得ることができた。また, 一定時間後の反射率が壁の厚さに関して単調な関係にならないことを見いだすことができた。

数値計算の結果は全体で 5000 ケース以上得られており, 最適化が施されたコードを再利用することの恩恵を十分に受けることができたと考えられる。

2　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) [19] 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。

2.1　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 [12]。

$$
\begin{aligned}
i\frac{\partial}{\partial t}\psi(x,t) &= H\psi(x,t) \\
H &= -\frac{1}{2}\Delta + V(x)
\end{aligned}
\tag{2.1}
$$

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 [12] 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できるため, 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。

$$
V_{periodic}(x) = V_{periodic}(x-l)
\tag{2.2}
$$

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。

$$
\phi(x) = u(x)\mathrm{e}^{-iKx}
\tag{2.3}
$$

$u(x)$ 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)。

$$
\begin{aligned}
Kl &= 2n\pi \\
n &\in \mathbb{N}
\end{aligned}
\tag{2.4}
$$

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 [9]。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。

$$
V_{KP}(x) = \begin{cases} V_0 & (0 < x \leq a) \\ 0 & (a < x \leq l) \end{cases}
\tag{2.5}
$$

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。(2.6) 式の条件を課して (2.5) 式の定義域を $(0, l]$ から $\mathbb{R}$ に拡張できる。

$$V_{KP} = V_{KP}(x - l) \tag{2.6}$$

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でき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 [20]。

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。

2.2 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。処理の内容を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 [21] 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。

- 平易な文法とバッテリー同梱思想 (たった 8 個の構文, 60 以上の組み込み関数, 100 を優に超える標準モジュール)
- 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)

python はプログラミング初心者でも中級者以上でも同じようなコードを書くことができるように言語が設計されているため, 習得は容易い。文法的には動的型付け言語であり, コンパイル言語にありがちなデータ型に関するストレスから初心者を開放している。一方, 型ヒントによってコンパイル言語のような取り扱いも可能なため, パッケージを始めとする中規模から大規模なプログラムを作る際にも使うことができる。

また, 初心者が最も苦労する過程の一つである環境整備についても, python の場合は多くの UNIX 系 OS でプリインストールされており, 容易な部類に入る。ただし, 互換性の無いバージョンの古いものが含まれる場合もあるため注意が必要ではある。QSB は python3 上で動作するように設計されているため python2 上では動作しない。

更に, マルチパラダイム言語であることは初心者向けパッケージを作る上で重要な性質である。初心者向けのプログラミングの教科書の多くが手続き指向のコードの書き方を最初に紹介している。そのため, プログラムを部品化しやすいオブジェクト指向や関数指向のプログラミングは比較的敷居が高い。そこで, 初心者が使用する際は手続き指向のコードを受け付けながら, パッケージ作成者が使用する際はオブジェクト指向を受け付けるという性質はかなり都合がよいといえる。

最後に, 高機能な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 [22] 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 の処理系は動的型付けのため, c, Fortran に比べて一般に低速である。これは, ループを実行する際に毎回の型チェックが必要であったり, 機械語に翻訳する際に c のプログラムを経由する必要があったりすることなどによる。

しかし, 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 BLAS(Basic Liner Algebra Subprograms), LAPACK(Liner Algebra PACKage) 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。内部の挙動がある程度理解できている人間でなければ, むしろ低速なコードを書いてしまうこともあり得る。

また, 連続量の離散化や, 密行列の疎行列化など, 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。(詳しくはセクション 3 参照)

QSB では, Schrödinger 方程式の初期値問題を 10 行以内で記述できる様になっている。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 (セクション 7.1 参照) とポテンシャルのグラフが得られる。

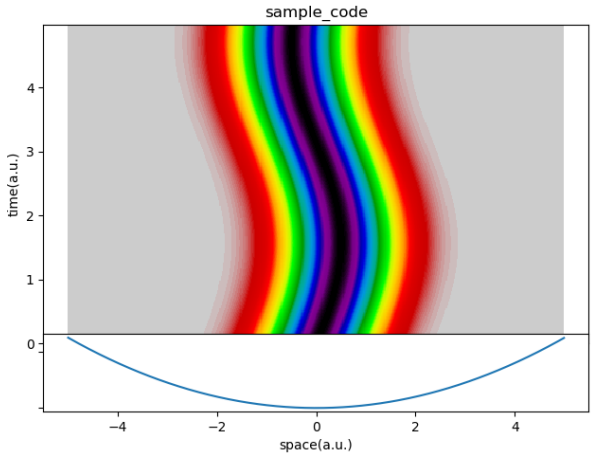

図 1　サンプルコード実行結果

3　QSB のクラス構成

QuantumSketchBook のオブジェクト (データの構造体) はクラス (メソッドやプロパティのかたまり) 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。

python には PEP8 [23] によって成文化されたコーディングの指針が存在する。QSB ではそれに準じてクラスの頭文字は大文字, 関数, メソッド, プロパティの頭文字は小文字としている。

また, PEP484 [24] によってタイプヒントというマナーが設定されている（あくまでマナーであるため, 実行時には何の影響も及ぼさない)。これによって, 引数名の後にコロン (:) をつけると, それ以降はデータ型 (文字列 str, 整数 int, 実数 numbers.Real などデータの種類を表すもの) として認識される。QSB でもこのマナーに則り, 引数と返り値の型を明記している。これによって気の利いたテキストエディタを使えば, コンパイル型の言語 (c や java に代表されるプログラムを実行する前に全てのコードを機械語へ翻訳する形式の言語) のように実行前にエラーを発見することができる。

このセクションにおける次章以降の命名規則を, python ドキュメントの形式にならって” オブジェクト名 (引数名:引数の型, ...)” と定める。デフォルト値 (引数が入力されなかった場合に代わりに使用される値) が設定されている引数は” 引数名=デフォルト値” と表記する。

例えば Mesh(x_min: Real, x_max: Real, dx: Real, t_min: Real, t_max, dt: Real) という章は, Mesh という名前で引数として x_min などの Real 型の 6 つの引数があるオブジェクトの説明をする章であることを示している。また Mesh の頭文字が大文字であることから, このオブジェクトがクラスだということも示している。

4　plot(arg: Plottable, show=True, save=False, title=”no_title”)

plottable なオブジェクト受け取り, 適切な形にプロットされた matplotlib.Figure を返す。返り値を束縛すれば, 通常の Figure と同様の処理をすることができる。プロットした後のグラフの体裁を変更するための最も一般的な方法という理由で, この返り値の形式が採用されている。

プロットする画像を見る必要がない場合や, 画面にグラフを表示する前にグラフの体裁を変更したい場合には引数 show を利用すればよい。引数 show が False の場合, この関数は matplotlib.Figure オブジェクトを構成するだけで, 画面にグラフを表示する副作用は発生しない。

また, save が True の場合, プロットされたグラフが title と同名の png ファイルとして保存される。title

はファイル名だけでなく, グラフの上部に表示されるグラフタイトルとしても使われる。

matplotlib の仕様によって, 一度作成した Figure は, clear() メソッドを呼び出してメモリを解法しない限り生き残り続ける。一度に大量のグラフを生成する場合は少し注意が必要である。一定以上の Figure が開放されることなく生成されると matplotlib は RuntimeWarning を送出する。

plot() 関数は内部的に arg の___plot___() メソッドを呼び出している。これは python 組み込みの len(), next(), iter() などと同様の形式である。よって, plottable なクラスを自作したい場合に何か特定のクラスを継承する必要はない。イテレータ (要素を次々と生成できるデータ構造) を自作する場合と同様に, ___plot___() メソッドを定義するだけでそのクラスは plottable の性質を持つ。

ただし, plottable なクラスを自作する場合は, ___plot___() メソッドは必ず Figure を返さなければならない。これは, 使用者の見えないところで Figure をみだりに生成しないために必要な規則である。一見, いちいち Figure を clear することは手間であり, ___plot___関数内で clear を呼び出す処理をした方が効率的に見える。しかし, そのような仕様では仕様者がグラフの体裁を変更する方法が失われる。また, 見えないところで Figure が生成され, 使用者にとって意図しない効果が起こるよりは clear を明示的に呼び出す方がましなことが多い。これは zen of python で知られる PEP20 [25] の”Explicit is better than implicit.” を意識した設計になっている。

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るべきで, 直接のインスタンス化 (具体的なデータを入力してテ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でき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できるよう 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

```
import QuantumSketchBook as QSB


with QSB.Context(-10, 10, 0.1, 0, 10, 0.1) as mesh:
    print(mesh.__class__)
    # --> <class 'QuantumSketchBook.mesh.Mesh'>

    print(mesh.x_num)
    # --> 200

    print(mesh.t_num)
    # --> 100

    print(mesh.x_vector)
    # -->[ -1.00000000e+01 -9.90000000e+00 -9.80000000e+00 -9.70000000e+00 ...]
```

4.3　Field(arg: Union[Iterable, Callable])

空間に依存する物理量を表す。引数には iterable(イテレータを生成するデータ構造。または, その性質のこと。例 : list, set, 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 (具体的なクラスの基本的な性質だけをまとめたもの) として設計さているため, 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, values という追加の引数から値の解釈を与えなければ ValueError を送出する。

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

```
import QuantumSketchBook as QSB
from QuantumSketchBook.field import Field


with QSB.Context(0, 2, 1, 0, 2, 1) as mesh:
    print(mesh.x_vector)
    # --> [0 1]

    function_field = Field(lambda x: x + 1) # 引数が関数の場合
    print(function_field.vector)
```

```
    # --> [1 2]

    list_field = Field([1, 2]) # 引数がlistの場合
    print(list_field.vector)
    # --> [1 2]

    generator_field = Field(x + 1 for x in range(2)) # iterableならば受け取ることができる
    print(generator_field.vector)
    # --> [1 2]

    wrong_length = Field([1, 2, 3]) # 要素数が合わない場合はValueErrorが送出される
    # --> 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 (親となるクラスのメソッドやプロパティを受け継いだクラス。あるクラス A, B が”A is a B. ” という関係のとき, A がサブクラス, B が親クラスに当たる) のため, Field と同様に配列と関数のどちらも受け取ることができる。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表す Potential クラスのインスタンス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。素直な実装では $O(n^2)$ の保存領域を使うため, すぐにメモリを使いきってしまう。

プロパティ	説明
matrix()	ポテンシャルの行列要素を返す

Listing 4　Potential の使用例

```
import QuantumSketchBook as QSB


with QSB.Context(-10, 10, 0.01, 0, 2, 1):
    step_potential = QSB.step(5, 0)
    QSB.plot(step_potential, title="STEP") # ステップポテンシャル

    box_potential = QSB.box(2, 1, 4)
    QSB.plot(box_potential, title="BOX") # 箱型ポテンシャル

    complex_potential = step_potential + box_potential
    QSB.plot(complex_potential, title="COMPLEX") # ポテンシャルの足し合わせ

    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
```

使用例を実行すると以下のグラフが出力される。

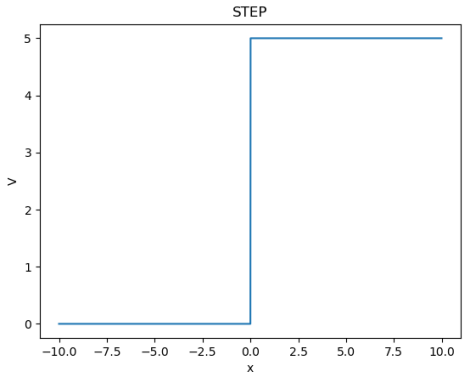

図 2　ステップポテンシャル

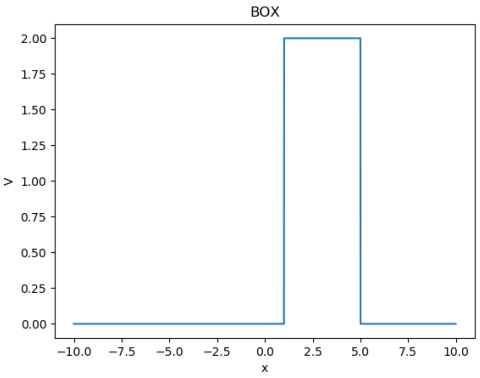

図 3　箱型ポテンシャル

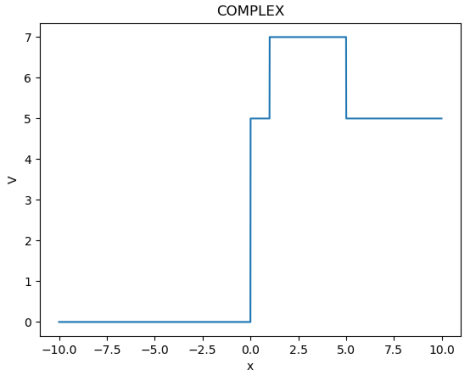

図 4　ポテンシャルの足し合わせ

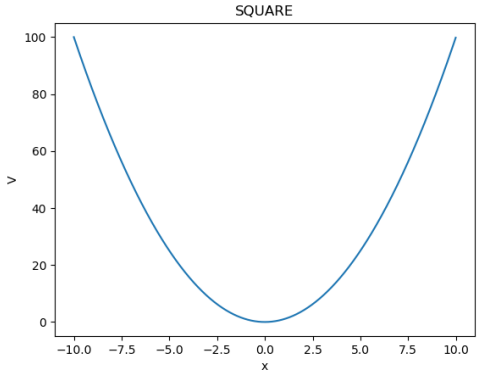

図 5　二次関数

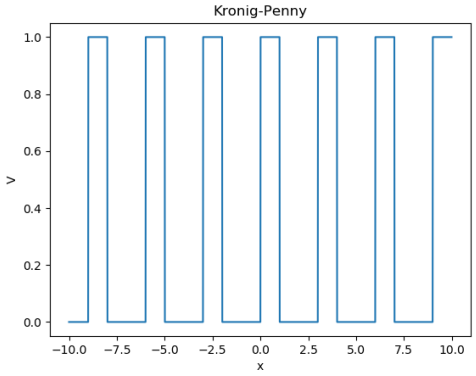

図 6　Kronig-Penny ポテンシャル

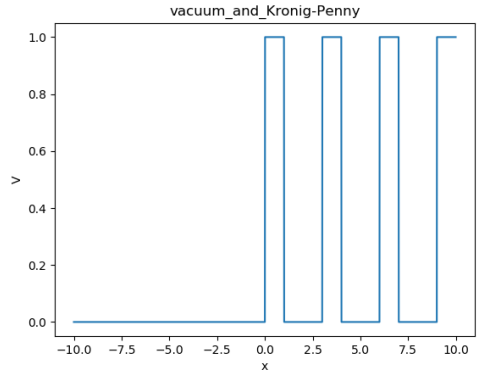

図 7　真空と Kronig-Penny ポテンシャル

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できる。

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できる。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

```
import QuantumSketchBook as QSB


with QSB.Context(-10, 10, 0.01, 0, 2, 1):
    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
    QSB.plot(gaussian, title="GAUSS")

    print(gaussian.random_values(5))
    # --> [-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。

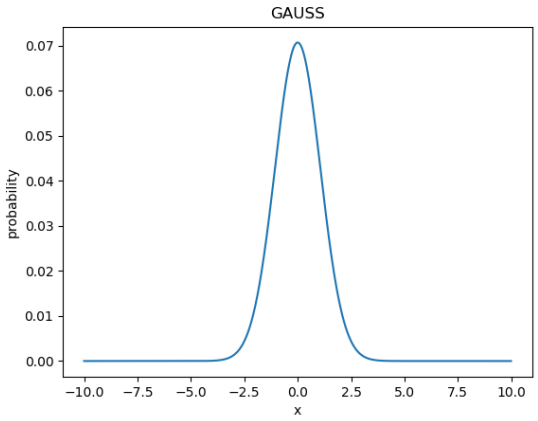

図 8　Gauss 型波束

4.6 Hamiltonian(potential: Potential, boundary="free")

Potential を受け取って, ハミルトニアン演算子を保持する。

境界条件の指定はこのクラスの引数によって行われる。指定可能な境界条件は挙動が安定している自由端境界条件"free", 固定端境界条件"fix", 周期境界条件"period" の 3 種類と, 試験的に導入されている吸収

境界条件”absorb” を合わせた 4 種類である。それ以外を指定した場合一部の例外を除いて, ValueError が送出される。デフォルト値は”free” に設定されている。

境界条件は, 本来の意味合い的には時空を管理する Mesh に保持させるべきものである。しかし, 同様の Mesh を使った異なる境界条件での計算を行う場面を考慮して, 変更不能で多くのデータの基礎となっている Mesh ではなく, 比較的変更しやすい Hamiltonian に保持させる形になっている。また, 物理的にも, Schrödinger 方程式の境界条件はハミルトニアンの行列要素に反映されるため, 大きく的を外した仕様ではない。

行列要素は疎行列によって保持されている。これによって, Potential の場合と同様に, 素直な実装では $O(n^2)$ のメモリを使うところを $O(n)$ に抑えている。また, 行列積の計算に疎行列用の効率化されたメソッドを使うことができるため, 計算時間も短縮される。よって, このクラスではメモリ, 時間ともに効率化されているといえる。

QSB を使用する分には気にする必要性はないが, scipy に付属されている微分方程式ソルバーは一変数関数の一階微分方程式しか解くことができない。一方, 波動関数は位置と時間とで二つの変数を持つため, そのままでは scipy を使って Schrödinger 方程式の求積計算をすることができない。そこで, QSB では各メッシュ点における波動関数をすべて独立な変数とみなし, 全ての変数について連立した微分方程式を作ることでこの問題を解決している。

Hamiltonian の実態は連立微分方程式の係数を保持する役割を担うものである。これは物理的には空間表示したハミルトニアンの行列要素を保持することに対応している。冒頭で述べたハミルトニアン演算子を保持するとは, 厳密にはハミルトニアンの行列要素を保持することを指している。

使用例は次章の Schroedinger の使用例で一緒に例示する。

プロパティ	説明
matrix	ハミルトニアンの行列要素を返す

4.7 Schroedinger(hamiltonan: Hamiltonian, state: State)

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。

積分器は scipy.integrate.ode を通して, Fortran サブモジュールの一つである zvode を指定している。注意として, zvode は scipy の仕様で 2 つ以上のインスタンスを同時に作ることはできないとされている。よって, QSB でも, 必ず一つの方程式を解き終えてから別の方程式を解かせるように使用しなければならない。zvode は non-stiff な問題に対して予測子修正子 Adams 法を使い, stiff な問題に対して後退差分法を使う。一般に予測子修正子 Adams 法のほうが精度が高く, 高速に計算できる。

計算量を節約しようとするときに計算範囲を狭くすることがあるが, 初期状態で波束の一部が計算範囲からはみ出してしまうと, 問題が stiff になり, zvode が後退差分法を適用してしまうため, 逆に計算量が増加してしまう場合がある。

計算範囲を広げすぎてしまうと余分な計算が増えるため, なるべく小さなメッシュを使用すべきなことに間違いはない。しかし, 波動関数がほとんどの場所で 0 になるような場合, 乗算のコストはあまり高くない。広範囲に広がらない波束のような状態の時間発展を計算する場合は, メッシュ範囲の大きさに敏感になる必要はあまりない。

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, Schrodeinger は iterable であり, 各ステップでの解を順々に返すことができる。この機能は Mesh が巨大すぎて全ての時間に渡る解を一度にメモリに乗せることができない場合の対抗策として設計されている。

Schroedinger が iterable として呼び出された場合は, 過去のステップで保持されたメモリが次のステップを呼び出すとすぐに開放される。そのため, 1 ステップ分のデータがメモリの上に乗ればどれだけ長時間に渡る計算であってもメモリエラーを起こさず実行することができる。

デフォルトのラプラシアンは (4.1) 式の形式を採用した。

$$\Delta := \frac{1}{360\Delta x^2}\left(4\phi_{i-3} - 54\phi_{i-2} + 540\phi_{i-1} - 980\phi_i + 540\phi_{i+1} - 54\phi i + 2 + 4\phi_{i+3}\right) \tag{4.1}$$

ただし ϕ_i はメッシュ番号 i における波動関数で, Δx は空間メッシュの間隔。境界値では境界条件に従って係数を変化させている。

(4.1) 式を tight-binding モデルで解釈すると, 滑らかな空間における 3 格子点先の飛び移りまでを考慮に入れることに対応している。

この形式は ϕ_i の左右 3 点と ϕ_i 自身を含む 7 点を Ragrange 補間し, 2 回微分することで得られる。7 点を与えた Ragrange 補間は ϕ_i 周辺を 6 次の精度で近似できるため, それを 2 階微分した (4.1) 式は $O(\Delta x^4)$ の精度がある。ただし, 等間隔メッシュで高次の Ragrange 補間を行うと, 近似した関数が激しく波打つ Runge 現象が起こるため, 安心して精度が高いと言えるのは ϕ_i のごく周辺だけである。

QSB では境界条件に”poor” と入力すると 1 次の精度がある公式と入れ替えられるように設計されている。

Hamiltonian は plottable であり, plot すると $x-t$ パターンとポテンシャルのグラフを生成する。プロットエリア上部に $x-t$ パターンが描写され, プロットエリア下部にポテンシャルのグラフが描画される。

プロパティ	説明
mesh	依存している Mesh
potential	依存している Potential
x0state	依存している State
ode	微分方程式のソルバー
equation(t, phi0)	ode によって解かれる方程式をメソッド化したもの
solution()	Schrödinger 方程式の解を返す。2 回め以降はキャッシュを用いるため高速
nelson()	Nelson の確率過程の標本を生成するための Nelson クラスオブジェクトを生成する

Listing 6　Hamiltonian と Schroedinger の使用例

```
import QuantumSketchBook as QSB


with QSB.Context(-40, 40, 0.05, 0, 60, 0.05):
    initial_state = QSB.gaussian_state(0, 8, 1)
    free_particle = QSB.potential(lambda x: 0 * x)

    free_boundary = QSB.Hamiltonian(free_particle, boundary="free")
    schroedinger = QSB.Schroedinger(free_boundary, initial_state)
    QSB.plot(schroedinger, title="free_boundary", save=True) # 自由端境界条件

```

```
12      fix_boundary = QSB.Hamiltonian(free_particle, boundary="fix")
13      schroedinger = QSB.Schroedinger(fix_boundary, initial_state)
14      QSB.plot(schroedinger, title="fix_boundary", save=True) # 固定端境界条件
15
16      periodic_boundary = QSB.Hamiltonian(free_particle, boundary="period")
17      schroedinger = QSB.Schroedinger(periodic_boundary, initial_state)
18      QSB.plot(schroedinger, title="periodic_boundary", save=True) # 周期境界条件
19
20      square_potential = QSB.potential(lambda x: 0.003 * x ** 2)
21      hamiltonian = QSB.Hamiltonian(square_potential)
22      schroedinger = QSB.Schroedinger(hamiltonian, initial_state)
23      QSB.plot(schroedinger, title="square_potential", save=True)
```

使用例を実行すると以下のグラフが出力され, 実行場所に png 形式のファイルが保存される。

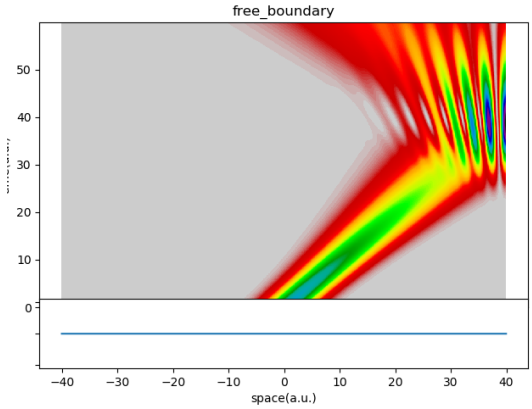

図 9　自由端境界条件

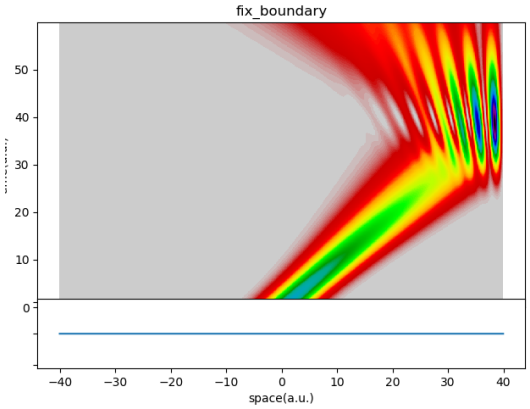

図 10　固定端境界条件

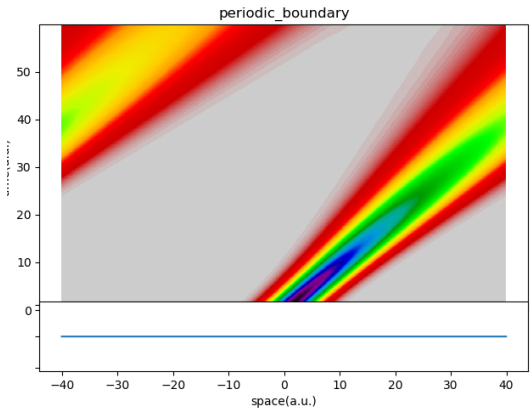

図 11　周期境界条件

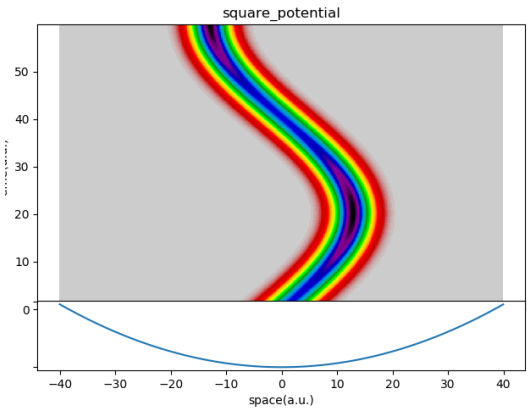

図 12　2 次関数

図 11 の境界条件による影響はその他の境界条件と区別しやすいが, 図 9 と図 10 も境界での反射の仕方が微妙に異なっている。具体的には, 図 9 は反射の際に必ず境界が着色される (確率が高いことを表す) が, 図 10 は絶対に境界が着色されない。

2 次関数ポテンシャルの例では波束がほとんど緩和しない。これは Schrödinger 方程式の定常解から考え

ても自然なことだと考えられる。また, 波束が左右に行ったり来たりする様子は, 古典的な質点の振る舞いとも一致する。これらの定性的特徴は QSB によって Schrödinger 方程式の求解ができていることを示す一つの証拠と言える。

5 QSB の妥当性検証

QSB の妥当性検証のため, Zhang の論文にある Heyperdiffusion の場合について QSB を用いた計算を行った。この章では Heyperdiffusion についての簡単な解説と Zhang の行った計算の設定を紹介し, 最後に QSB による計算結果を示す。

5.1 Heyperdiffusion

Heyperdiffusion は拡散現象の形態の一つで, 分散 σ^2 が時間 t に関して (5.1) 式の形で拡散することをいう。

$$\sigma^2 \propto t^\alpha \tag{5.1}$$

$$2 < \alpha \tag{5.2}$$

ブラウン運動的な拡散は $\alpha = 2$ で拡散するため, Heyperdiffusion はブラウン運動を越える拡散のことだといえる。

Schrödinger 方程式に従う真空中の Gauss 型波束も同様に t^2 に比例した速さで分散が増加する。よって, ポテンシャル中の量子波束が Heyperdiffusion を起こすということは真空中よりも高い次数に比例して拡散していくことと等しい。もちろん, 比例定数が異なるため, 単純に次数が大きい, 即ち速く拡散しているとはいえないが, 素朴な直感に反することは事実である。

この現象の理論的な説明の一つとして Hufnagel [13] の point-source モデルによる説明を紹介する。このモデルでは, x = 0 に集中した確率の源が, 指数関数的に減衰する様子を (5.3) 式の形で仮定する。

$$P(t) = \mathrm{e}^{-\Gamma t} \tag{5.3}$$

ここで $P(t)$ は, x=0 に粒子が存在する確率を表す。Γ は拡散のスケールを表す正の定数。x=0 の位置から失われた粒子が, 等速で周囲に拡散していくと仮定すると, (5.4) 式のように分散 $M_{PS}(t)$ を計算することができる。

$$M_{PS}(t) = \int_0^\infty dx\, x^2 \int_0^t dt'(-\dot{P}(t'))\delta(x - v(t-t')) \tag{5.4}$$

$$= v^2\Gamma \int_0^t dt' \mathrm{e}^{-\Gamma t}(t-t')^2$$

$$= v^2\left(t^2 - \frac{2}{\Gamma}t + \frac{2}{\Gamma^2} - \frac{2}{\Gamma^2}\mathrm{e}^{-\Gamma t}\right) \tag{5.5}$$

(5.5) 式の 1 項目は通常のブラウン運動的拡散を表すが, 2 項目以降の項はそれと異なる影響を表している。$t << \Gamma$ というゆっくりとした拡散の状況では 2 項目以降は打ち消し合うため, ブラウン運動と一致する。一方, $t \sim \Gamma$ の状況ではこれらの項が無視できない影響を与えるため, 異常な拡散の挙動を示す。Hufnagel はこの拡散を Heyperballistic な拡散と呼んだ。

5.2　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。主格子が Hufnagel の言うところの point-source に対応する。

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。Zhan の計算では主に $L = 50$ としているため, 今回の再現もそれにならった。

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。Zhang が行った周期的なポテンシャルにおける計算の中では $V = V_0(-1)^i$ を使用していた。ここで V_0 はポテンシャルの高さを決定する因子で, 0.0, 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 $\delta_{0,i}$ を使用していた。

ハミルトニアン内に含まれるラプラシアンについては (5.6) 式を使用している。

$$\Delta\phi_i := \frac{\phi_{i-1} - 2\phi_i + \phi_{i+1}}{\Delta x^2} \tag{5.6}$$

この式は $O(\Delta x)$ の精度がある [26]。

また, tight-binding モデルで解釈すると, 1 次の飛び移りまでを考慮することに対応する。格子間の飛び移りが QSB で再現する際は, Hamiltonian に一次精度のラプラシアンを使うように指定した。

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ところ, 分散は図 13 のように時間発展した。個別のケースの計算には各 2 時間程度を要した。

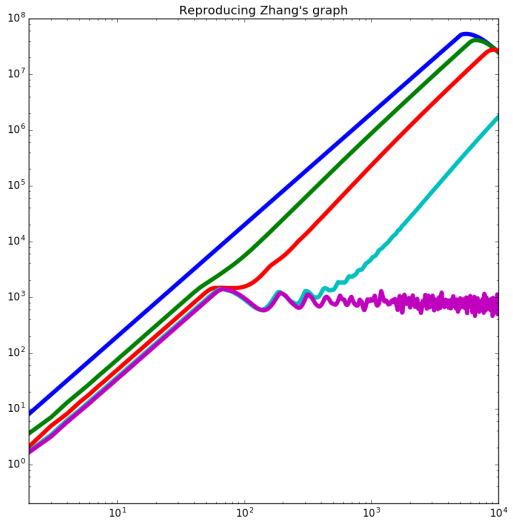

図 13　波束の分散の時間推移

線の色はそれぞれ上から V_0=0.0, 1.0, 1,5, 1,9, 2.0 に対応している。

縦軸は分散を表し, 横軸は時刻を表している。両対数軸をとっているため, グラフの傾きが多項式近似した際の次数に対応する。縦軸と横軸の比は 2:1 であるため, 通常のブラウン運動的拡散でのグラフは傾き 45

度の直線となる。Heyperdiffusion が起こるとグラフの傾きが 45 度を超えるものが現れる。

図 13 の特徴として, 傾きが 45 度を超え, 分散が $t^{\alpha}(\alpha > 2)$ に比例して増加する現象が見られた。これは 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, 最小二乗法フフィッティングにより, 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

これによって, Zhang の計算と QSB の計算が一致したということが確認できた。以降の計算も妥当であるという前提で解釈を行う。

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 $\mathrm{std}(x_0, \sigma, x)\exp[ikx]$ 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。標準偏差は波動関数のもので, 確率密度の標準偏差は 3.00 となる。

波束の状態を固定する理由は, 波束の拡散現象がスケール不変なことにある。波束の平均波数を変えても, ポテンシャルと時間, 空間のスケールを変換することで同じ状況に対応させられる。標準偏差を拡大することは, 空間のスケールを縮小し, 時間のスケールを拡大することに対応する。よって, 波束を固定し, ポテンシャルを変えるだけでポテンシャルと波束の対応関係を見る分には十分である。

ポテンシャルを固定して波束を変化させる方法を取らない理由は, 波数を大きくすると計算領域の境界にたどり着いてしまう可能性が高くなったり, 時間スケールを拡大するためにメッシュを変更することには余分なコストがかったりと QSB で数値計算する上でのデメリットが多いからである。

周期ポテンシャルは以下のように設定した。

$$V(x) = \begin{cases} 0 & (x < 0) \\ V_{KP}(x) & (x > 0) \end{cases} \tag{6.1}$$

実際の数値計算では, QSB の vacuum_kp() 関数を用いた。

$V_{KP}(x)$ 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。本研究の計算ケースはポテンシャルにのみ依存するため, これらの 3 つのパラメータの組み合わせ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。

実際の結晶表面はダングリングボンドや格子欠損のような表面特有の現象が起こるが, このモデルではその影響は含まれていない。また, 実際の結晶中分子が作るポテンシャルは Kronig-Penny ポテンシャルのような切り立った傾きで立ち上がっていない。飽くまで Kronig-Penny 型の周期ポテンシャルに入射する電子波束を想定している。

平均波数 2.00 は真空中の平面波において 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

QSB によって 5000 ケース以上のケースについて計算し, 得られた波束の時間発展を $x-t$ パターンの形式で図示した。

$x-t$ 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はできない。

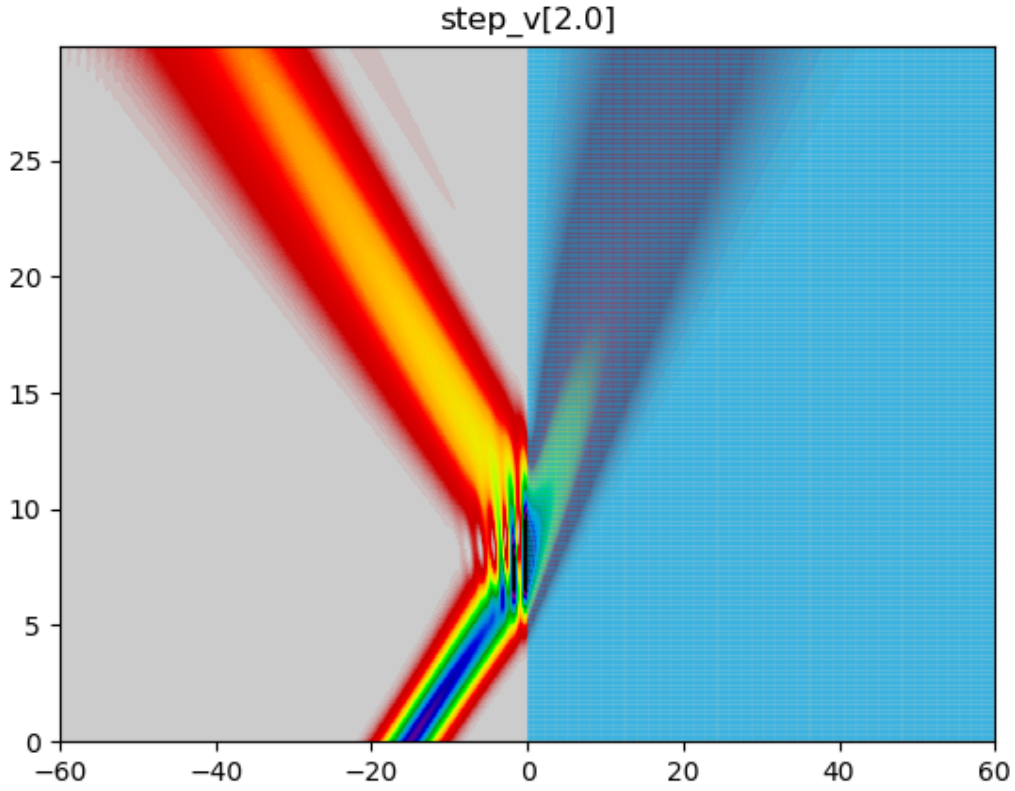

図 14　$x-t$ パターンの例:ステップポテンシャルに入射する波束をシミュレーションしたもの

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。灰色の領域は確率が 0 の領域を表している。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。

図 14 では右側のポテンシャル領域に入射した波束が 5 から 10 単位時間ごろにポテンシャル表面で反射し, 左側に伝搬していくものと, ポテンシャル領域に侵入していくものとに分かれる様子が確認できる。

ポテンシャルに侵入した進行波の軌跡の傾きが急になっていることから, 伝搬速度が遅くなっている様子

も確認できる。

反射波の方は時間経過と共に黄色の領域が少なくなり赤の領域が増していることから, 波束が緩和している様子を確認することができる。

$x-t$ パターンはこのように多くの情報をアニメーションに頼らずに得ることができるため, 量子力学的な現象であっても時間発展の様子を知るための道具として優れている。アニメーションは計算機での生成コストが高く, 大量に作ることが困難である。また, 人間がアニメーションを観る際には再生時間が必要で, 大量のデータを処理することにも向いていない。読解に少しコツが必要だが, 大量のケースについて読解する必要がある場合は $x-t$ パターンのほうが適切だと考えられる。

次章以降ではシミュレーションによって得られた以下の特徴的な $x-t$ パターンを例示する。

- 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

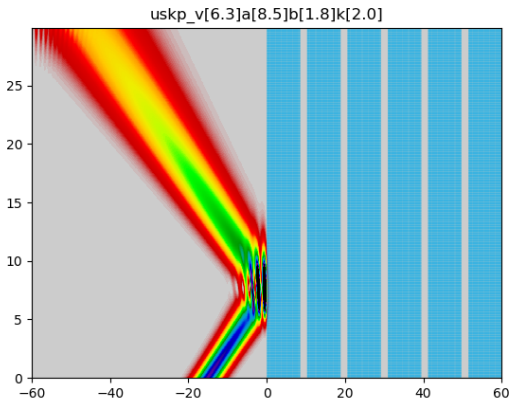

図 15　全反射の例 V_0=6.3, a=8.5, b=1.8 の場合

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。ポテンシャルの高さは 2 よりも大きくなければ現れない。逆にポテンシャルの高さが 5 を超えると, 全反射以外のパターンが現れることは稀になる。

$x-t$ パターン上では, どの個別ケースもほぼ同じような形状をしているため, 複数の例示はしない。

ポテンシャルの高さが 2 よりも小さい場合は入射エネルギーよりもポテンシャルのほうが低いため, ポテンシャルに侵入する波束が現れ, 全反射パターンは現れなくなる。その代わりに停滞状態が多くを占める。

7.3　停滞状態

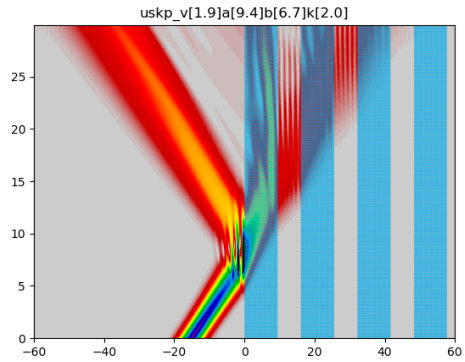

図 16　停滞状態の例:典型的なもの V_0=1.9, a=9.4, b=6.7

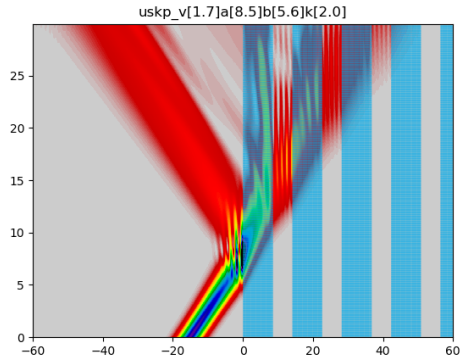

図 17　停滞状態の例:2 層目でも黄色い領域が見られる V_0=1.7, a=8.5, b=5.6

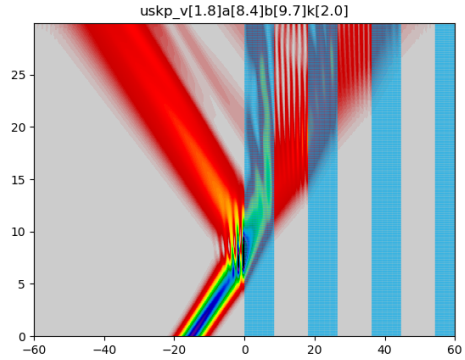

図 18　停滞状態の例:1 層目壁内部で 2 回真空領域に反射している V_0=1.8, a=8.4, b=9.7

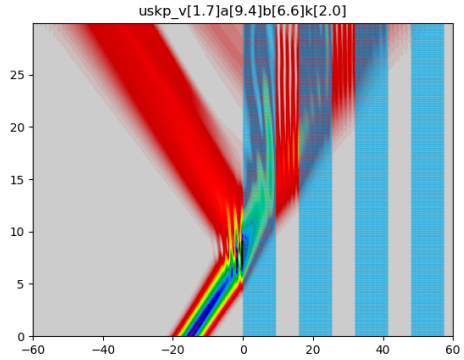

図 19　停滞状態の例:観測した中で停滞時間が最長のもの V_0=1.7, a=9.4, b=6.6

ポテンシャルが 2 より低く, 壁が 5 よりも厚い場合に分かりやすいものが現れる。井戸ではなく壁の領域に注目すると, 壁内部にとどまり続ける波が存在している。

ポテンシャルが 2 よりも低い設定では多かれ少なかれ必ずこの状態の影響が見られる。停滞状態はポテンシャルが 2 を超えると段々と見られなくなり, 3 を超えるとほとんどの影響が見られなくなる。壁が厚ければ厚いほど内部に波が留まりやすい。

素朴に考えるとポテンシャルが高い領域に存在する確率は低くなりそうだが, シミュレーション結果では井戸よりも壁に多く確率が集まる様子が確認できた。

ポテンシャルが高い領域に確率が集まる現象は, 後述するが, 箱型ポテンシャルにも見られる。

図 18 の時刻 20 単位時間ごろのポテンシャル境界付近に注目すると分かりやすいが, 壁にとどまる波の一部は真空方向へ反射する。しかし, 波束の軌跡は大きく弧を描くように曲がっており, 時間をかけて反射していく様子が見受けられる。

壁内部に停滞する波束は, かなり大きい時間スケールで周囲の真空や井戸に拡散していく。この影響で,

真空側の反射波は波束のようなまとまった形は取らず, 散発的に壁から放出されていくものが現れる。

一枚の壁からどの程度の時間をかけて放出されて行くのかを調べるため, 箱型ポテンシャルに同様の波束を入射したところ, 図 20 のような $x-t$ パターンを得た。

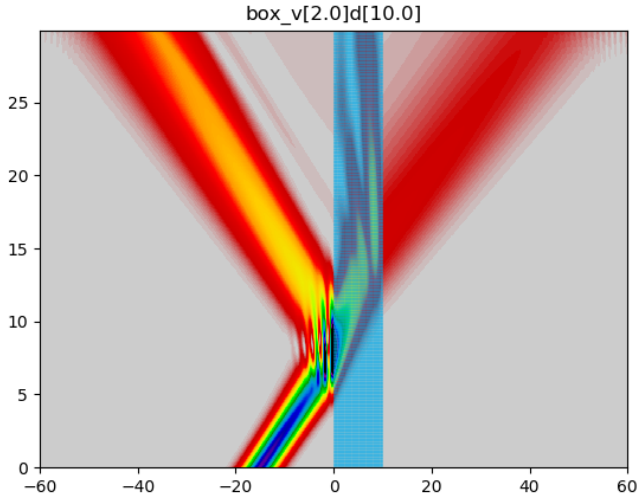

図 20　箱型ポテンシャルに入射した波束の時間発展 V=2.0

壁の内部にとどまり続ける波が観察できる。ゆっくりと弧を描くように反射する様子も, 周期的に並んでいる場合と同様の傾向が見受けられた。図 20 を更に 3 倍の時間について時間発展を計算すると図 21 が得られた。

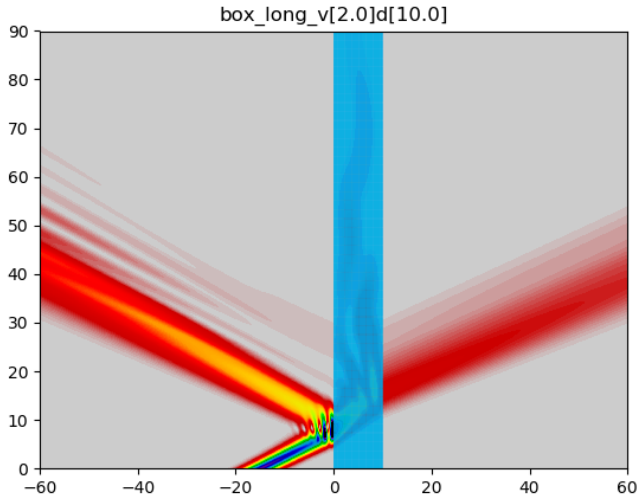

図 21　90 単位時間まで計算された箱型ポテンシャルにおける時間発展

境界の影響を排除するため, 画像には描かれていないが, 空間メッシュを 3 倍に広げて計算している。

図 21 では壁内部に 80 時間程度まで停滞する様子が見受けられる。着色する閾値以下ではさらに停滞していると考えられる。

このように壁内部に長時間滞在し, ゆっくりと時間をかけて外部に放出する影響で, 次章以降に示すパターンにはポテンシャル境界での反射では説明できないノイズのようなものが混入する。停滞の影響を受ける, 受けないで波束の振る舞いはかなり違ったものになるため, ポテンシャルが 2 より大きいか小さいかは波束の時間発展を決める上で重要な因子となる。

壁内部の滞在時間が長くなること理論的な間接証拠として, 箱型ポテンシャル定常状態における壁内部と真空の確率密度の比を図 22 に示す。

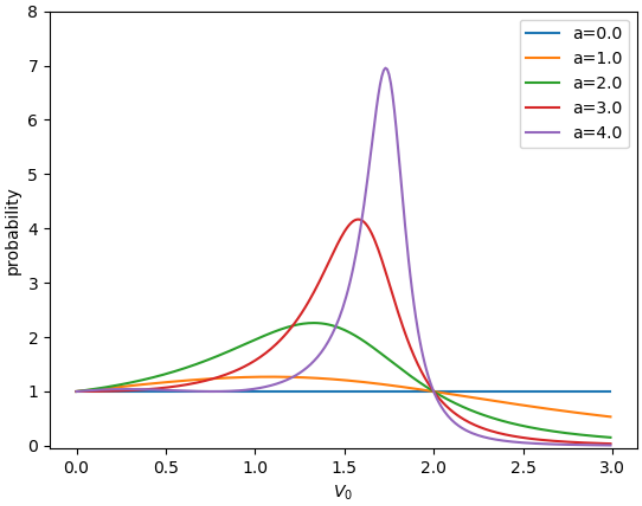

図 22　真空領域の滞在確率/壁内部の滞在確率

系のエネルギーはシミュレーションにおける平均エネルギーと等しい 2 に設定してある。

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。a は Kronig-Penny モデルと同様に壁の厚さを表している。

図 22 から, $V_0 < 2$ の領域では壁内部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壁の厚さを表す a が大きくなるほど鋭くなる。

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。この傾向はシミュレーションでポテンシャルを 2 周辺に設定したときに, 壁が厚くなるほど停滞し続けるという停滞状態の傾向と一致している。このことから, Kronig-Penny ポテンシャルにおけるシミュレーションでも同様の傾向があると考えられる。

また, 直感的な説明として壁内部の伝搬速度を使う説明が考えられる。箱型ポテンシャルの場合, 壁内部の位相速度は準波数に比例する。順波数はエネルギーを ε としたとき, 準波数は $\sqrt{2m(\varepsilon - V_0)}$ となるため, $\varepsilon \sim V_0$ の状況では, ほぼ 0 になる。伝搬速度もほぼ 0 になるため, 粒子は長時間壁内部に滞在し続けると説明できる。

ただし, 波束の場合は様々な波数の波が重なり合っているため, 壁内部で完全に停止する粒子は現れない。

7.4　高次反射波

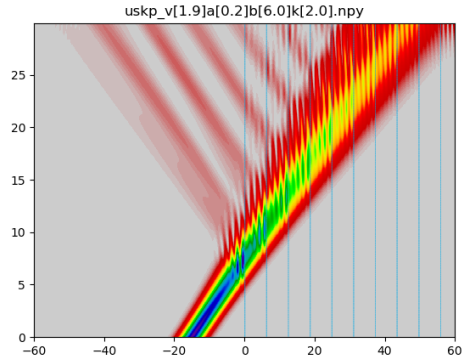

図 23　高次反射波の例:壁が薄めのもの V_0=1.0, a=2.7, b=5.8

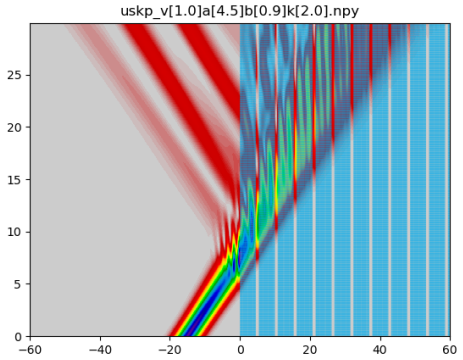

図 24　高次反射波の例:壁が厚めのもの。停滞状態の影響がみられる V_0=1.0, a=4.5, b=0.9

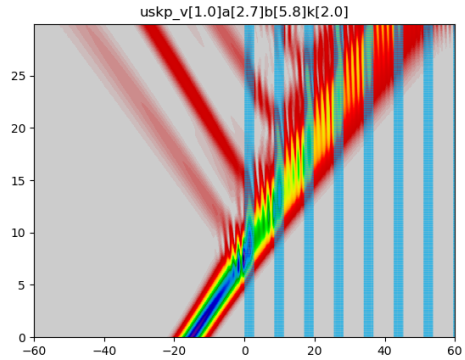

図 25　高次反射波の例:2 次の反射波が卓越している V_0=1.9, a=0.2, b=6.0

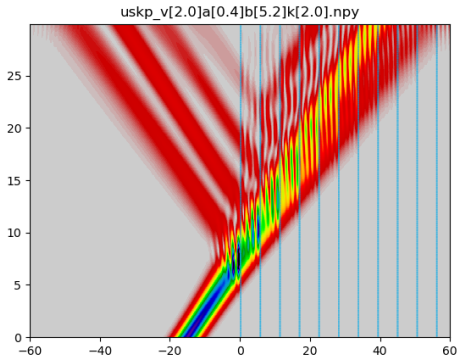

図 26　高次反射波の例:2 次進行波も同時に見られる珍しい例 V_0=2.0, a=0.4, b=5.2

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 23 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 24 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 23 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。

7.5　高次進行波

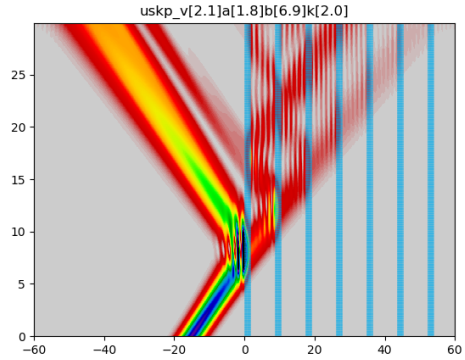

図 27　高次進行波の例:典型的なもの V_0=2.1, a=1.8, b=6.9

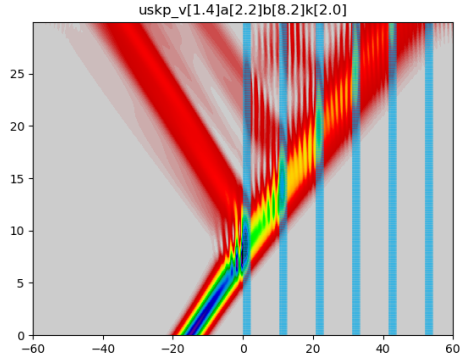

図 28　高次進行波の例:ひし形の模様がかなり大きく見える V_0=1.4, a=2.2, b=8.2

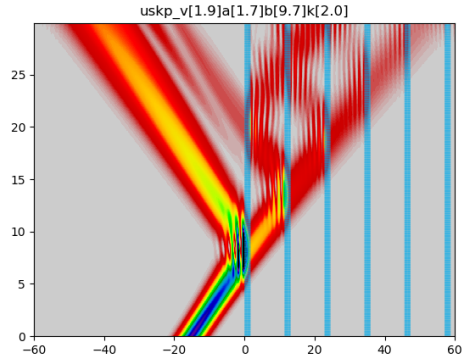

図 29　高次進行波の例:2 次進行波が次第に卓越する様子が見られる V_0=1.9, a=1.7, b=9.7

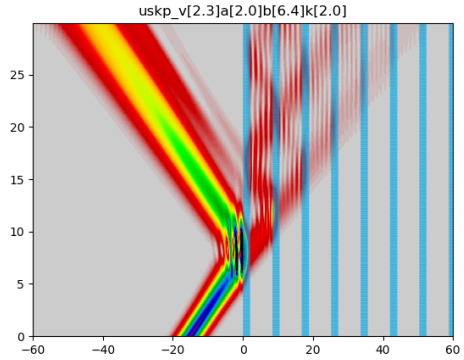

図 30　高次進行波の例:3 次進行波も見られる珍しい例 V_0=2.3, a=2.0, b=6.4

V_0=2 前後, 壁の厚さが 2 程度, 井戸の幅が壁の倍以上のときにみられる。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 27 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合わさるため, 単調に減衰するとはいえない。図 27 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 [17] 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 28 と図 29 を比較すると図 29 のほうが井戸が広い。しかし, 図 29 が 3 次反射波まで観察できる一方, 図 28 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 29

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。

7.6　分裂状態

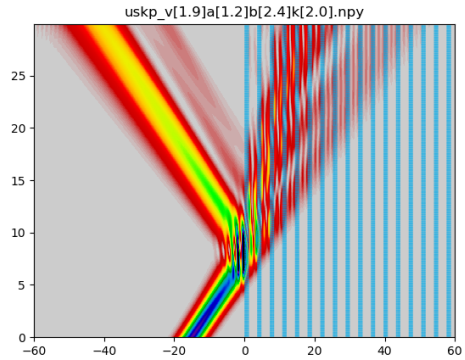

図 31　分裂状態の例:分裂がいくつも起こっている V_0=1.9, a=1.2, b=2.4

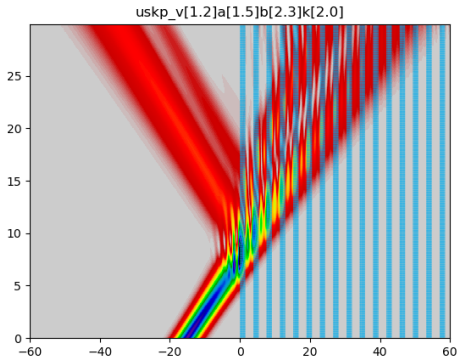

図 32　分裂状態の例:壁が厚いもの V_0=1.2, a=1.5, b=2.3

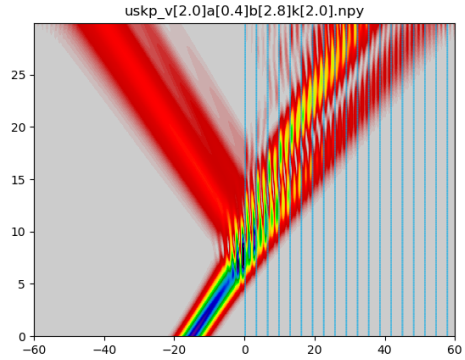

図 33　分裂状態の例:2 次反射波が卓越している v_0=2.0, a=0.4, b=2.8

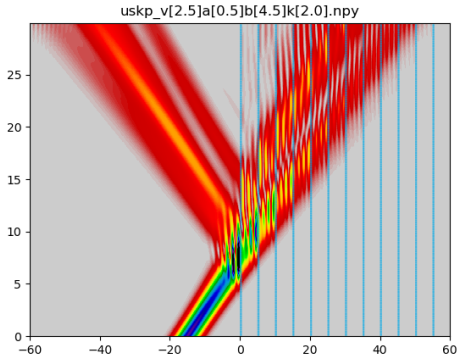

図 34　分裂状態の例:2 つの進行波が最も綺麗に分かれているもの V_0=2.5 , a=0.5, b=4.5

ポテンシャルが 2 前後でよくみられる。ポテンシャルが高すぎるとみられなくなる。壁と井戸の関係に簡単な関係は見出されないが, 壁と井戸が同程度の場合に多く見られる。また, 壁が厚いと停滞の影響を受けて波束が分かれて見えないことが多い。

ポテンシャルの内部に侵入した波束が二つ以上に分裂する。同時刻で切り取ると, 2 つ以上の波束が固まって伝搬しているように見える。

図 33 が最も分かりやすいが, 入射波と反射波が重なり合っているため, どの時刻で反射が起こるのかといった判別はできない。しかし, 全体として複数の進行波が伝搬しているように見える。

井戸の幅が大きくなると進行波同士の時間的間隔が大きくなるが, 2 次進行波に含まれる波が少なくなる。逆に井戸の間隔が狭いと, 1 次進行波がすぐに減衰する。そのため, 2 次進行波と 1 次進行波が綺麗に分かれるには, 井戸が適度に広く, 1 次進行波と 2 次進行波がどちらも減衰しないようなバランスが必要になる。

また, 図 31 のように 2 次以上の高次の進行波が発生することも珍しくない。進行波は次第に減衰し, さらに高次の進行波が支配的になる。この様子は高次進行波の章で記述した内容と一致している。入射, 反射が絡みあった状態ながら, 全体として同様の傾向が見えることは非常に興味深い。

さらに, 物質波波束において類似の現象が見られる [9] が, これは物質波波束の非線形効果の影響と言われている。非線形効果を付加していない本研究での計算でも類似の現象が見られることも非常に興味深い。

7.7 屈折状態

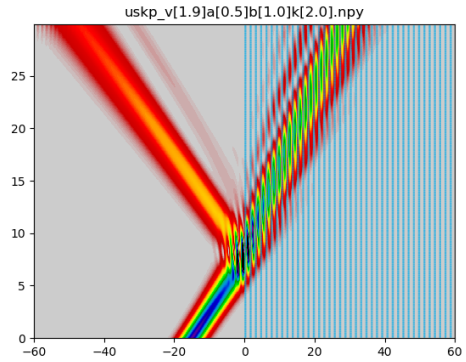

図 35 屈折状態の例:周期が 1.5 のもの v_0=1.9, a=0.5, b=1.0

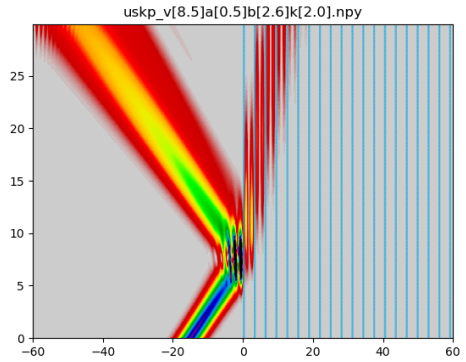

図 36 屈折状態の例:周期が 3.1 のもの v_0=8.5, a=0.5, b=2.6

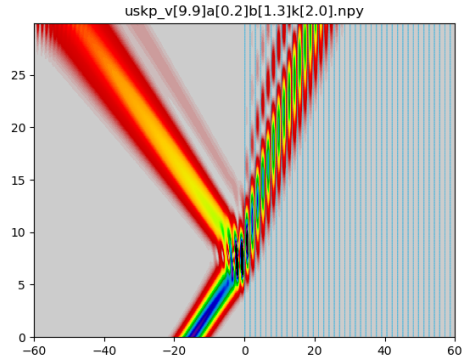

図 37 屈折状態の例:ポテンシャルが高いもの v_0=9.9, a=0.2, b=1.3

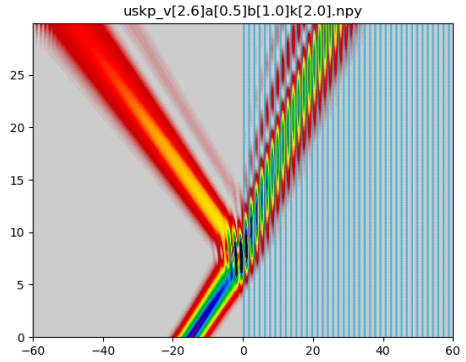

図 38 屈折状態の例:進行波が最も一体として伝搬しているもの v_0=1.9, a=0.5, b=1.0

ポテンシャルの周期が 1.5, 3.1 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=9.9 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。特に V_0 が 5 を超えるような設定では, ほとんどの場合で全反射が起こるが, 屈折状態だけは例外的に反射率が低い。

ポテンシャルに入射した波束の傾きが, 真空中に比べて大きくなっていることから, 同じ位置に長時間滞在し続け, ゆっくり伝搬していく様子を見ることができる。

屈折状態が見られた周期はおおよそ π, $\pi/2$ であり, de Broglie 波長の平均に一致する。よって, この現象は何らかの形でポテンシャルと波束との共鳴的な現象が起こっていると考えられる。並進対称性がないに

も関わらず Bloch の定理のように逆格子定数のようなものが関わってくるならば, 非常に興味深い現象と言える。

7.8　トラップ状態

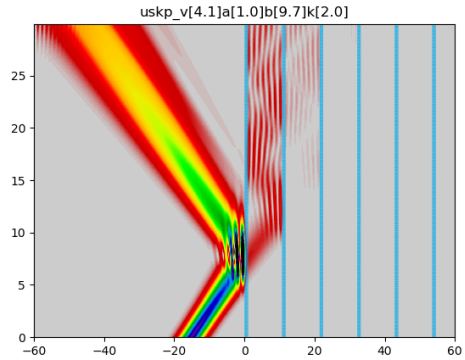

図 39　トラップ状態の例:1 層目と 2 層目の間で反射し続けている V_0=4.1, a=1.0, b=9.7

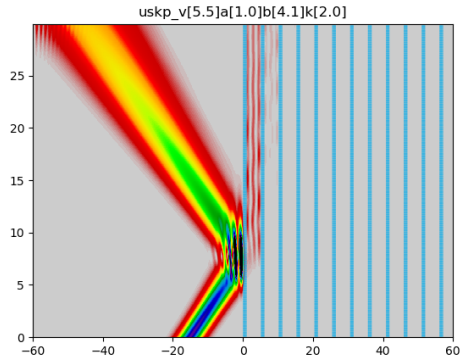

図 40　トラップ状態の例:井戸内で 3 つの腹が確認できる V_0=5.5, a=1.0, b=4.1

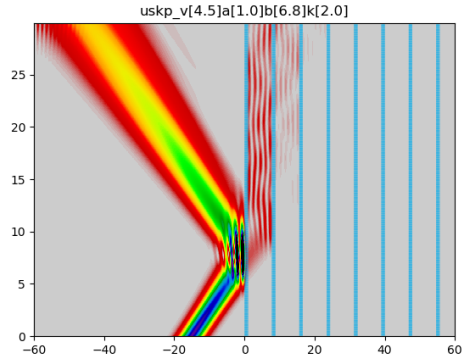

図 41　トラップ状態の例:井戸内で 4 つの腹が確認できる V_0=4.5, a=1.0, b=6.8

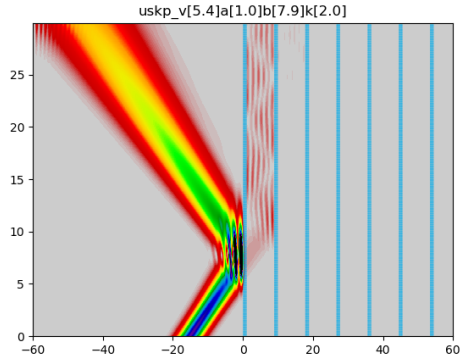

図 42　トラップ状態の例:井戸内で 5 つの腹が確認できる V_0=5.4, a=1.0, b=7.9

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。

3 程度のポテンシャルの設定でトラップ状態のようなものが見られないわけではないが, ポテンシャルが 5 を超えるような高い設定の場合に $x-t$ パターン上で縦縞模様が現れるような典型的なトラップ状態が見られる。5 程度のポテンシャルの設定では, 壁が薄い場合にトラップ状態や分裂状態が見られ, 壁が厚くなるごとに全反射へと近づいていく。ポテンシャルが 8 程度になると全反射と屈折状態が支配的になり, トラップ状態のようなものは稀にしか見られない。

また, 図 39 が分かりやすいが, トラップ状態といっても 1 層と 2 層の間に完全にとどまり続けることはなく, 次第に 2 層目と 3 層目の間に伝搬していく。

図 40 では赤い領域が縦縞の様に 3 つ周期的に並んでおり, 共鳴のような構造が見られる。共鳴にならっ

て確率密度が高くなっているところを腹と呼ぶことにする。図 40, 図 41, 図 42 と井戸の領域が広がるごとに, ひとつの井戸内に含まれる腹の数は増加しており,「共鳴の波長」が井戸の幅によらないことが見受けられる。ガウス型波束の透過波が, あたかも切り取られた平面波のように振舞っているようにみえる様子は非常に興味深い。

しかし, $x-t$ パターンでは確率密度しか見取ることができないため, 位相の情報を得ることができない。確率密度が周期的に波打っていることは, トラップ状態の波動関数が平面波と類似していることを直接示すものではない。

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。

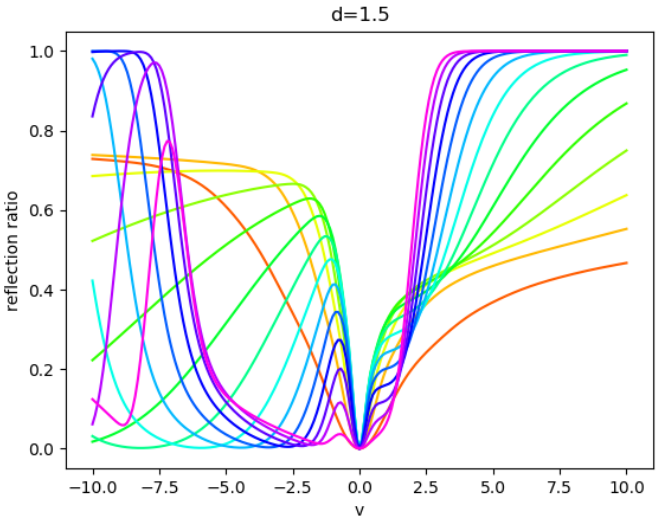

図 43　周期 1.5 のポテンシャルの反射率

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, シミュレーション結果はそのようにはならない。

比較のために箱型ポテンシャルの定常状態について, 壁の幅を対応させた形で同様のグラフを作成したところ, 図 44 を得た。

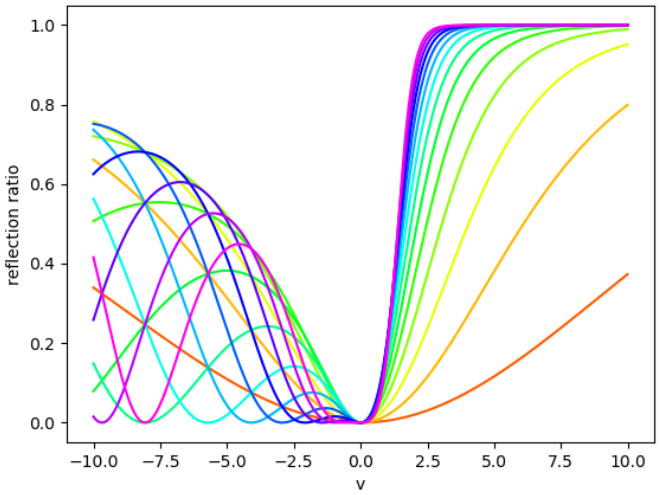

図 44　箱型ポテンシャルの定常状態における反射率

図 44 では図 43 のような逆転現象は見られない。これは反射率の定義の違いによるものと考えられる。図 43 では 30 単位時間後という有限の時間が経過した時点の真空領域に粒子が存在する確率を反射率の定義にしているが, 図 44 では定常状態という, 無限の時間が経過した状態における真空領域での進行波と後退波の振幅の比によって定義している。

有限の時間では, セクション 7.3 で述べたように壁内部に波が停滞し, 長い時間スケールに渡ってゆっくりと波が放出される現象の影響を受ける。図 43 において, 壁が非常に厚くなった場合に反射率が低くなる現象が見られる理由は, 停滞状態が関係していると考えられる。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, 有限時間での反射率は低く見積もられる。

8 結論

波束の時間発展に関して, 最適化されたコードを実行するためのフレームワーク QantumSketchBook の作成を行った。これによって, Schrödinger 方程式の初期値問題の求積と可視化を 7 行で記述できるようなフレームワークを作成することが出来た。

QSB の妥当性を検証するために Zhang の計算を再計算した。その結果, 二つの計算方法でほぼ同様の結果を得られたことが確認できたため, QSB の計算について, ある程度の妥当性を保証することができた。

QSB を用いて Kronig-Penny ポテンシャルに入射する Gauss 型波束の時間発展を 5000 ケース以上シミュレーションした。このような大量のケースを計算することが可能となったのは, フレームワークによって効率化されたプログラムを簡単に再利用することが可能になったからである。フレームワークを活用することの恩恵を十分に受けることができたといえる。

$x-t$ パターンを作成したところ, 特徴的な 7 パターンを抽出することが出来た。

停滞状態では直感に反してポテンシャル領域に長時間滞在する粒子が存在することを見取ることができた。これは時間発展する粒子の興味深い性質の一つである。

また, 2 個以上の波束が真空領域へ反射する高次反射波の現象を見取ることができた。特に 1 次反射波よりも 2 次反射波が卓越することと停滞状態との関係を見いだすことが出来たことは, 今回見取ることができた量子波束の興味深い性質の一つといえる。

更に, 高次進行波の時間的間隔が単純な井戸の距離だけでは表されず, 反射そのものにかかる時間も影響することを匂わせる現象が見られたことも興味深い結果と言える。

分裂状態では入射波と反射波の影響が絡み合っている状況でも波束が全体として進行波のように伝搬していく様子を確認できた。また, 非線形項の影響が入っていないにも関わらず, 波束が 2 つ以上に分裂していく現象は直感的に予想することが難しい現象の一つと言える。

屈折状態では, 一定の周期を持つポテンシャルで, ポテンシャルの高さに関係なく類似の現象が見られたことは, 何らかのポテンシャル構造に関する普遍性を示すものとして興味深い。また, 高いポテンシャルの設定でも透過率の高い状況を選択的に作り出すことができる可能性を与えるこの結果は, 実用的な応用を微かに期待させる。

トラップ状態では様々な波数の波が重なり合っているはずの波束において, ひとつの周波数を持っているような振る舞いが見られた。波束の振る舞いについてのひとつの知見を得ることが出来たと考えられる。

最後に, 壁の厚さと反射率の関係では, 壁が厚くなるほど反射率が下がる場合があるという, 直感とは反する結果が得られた。有限の時間内に起こる反射と定常状態における反射の違いが問題になる場合の例を一つ提供することができたと考えられる。

本研究では, フレームワークの有効性を示したことと波束の時間発展に関する興味深い現象の知見を得たことという, 二つの成果が得られたと考えられる。

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。

$$[T_d, H] = 0 \tag{付録 A.1}$$

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。

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

ϕの周期をndとした。ここで以下の関数uを考える。

$$u(x) = \mathrm{e}^{iKx}\phi(x) \tag{付録 A.3}$$

$$\begin{aligned} T_d u(x) &= \mathrm{e}^{iK(x-d)}\mathrm{e}^{iKd}\phi(x) \\ T_d u(x) &= \mathrm{e}^{iKx}\phi(x) = u(x) \end{aligned} \tag{付録 A.4}$$

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。

$$\phi(x) = \mathrm{e}^{-iKx}u(x) \tag{付録 A.5}$$

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。

$$\begin{aligned} \phi_{well}(x) &= W^+\mathrm{e}^{+ikx} + W^-\mathrm{e}^{-ikx} \\ \phi_{barrier}(x) &= B^+\mathrm{e}^{+iqx} + B^-\mathrm{e}^{-iqx} \\ k &= \sqrt{2m\varepsilon} \\ q &= \sqrt{2m(\varepsilon - V_0)} \end{aligned} \tag{付録 B.1}$$

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。

$$\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} W^+ \\ W^- \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ \rho & -\rho \end{pmatrix} \begin{pmatrix} B^+ \\ B^- \end{pmatrix} \tag{付録 B.2}$$

$$\begin{pmatrix} \mathrm{e}^{ika} & \mathrm{e}^{-ika} \\ \mathrm{e}^{ika} & -\mathrm{e}^{-ika} \end{pmatrix} \begin{pmatrix} W^+ \\ W^- \end{pmatrix} = \begin{pmatrix} \mathrm{e}^{iqa} & \mathrm{e}^{-iqa} \\ \rho\mathrm{e}^{iqa} & -\rho\mathrm{e}^{-iqa} \end{pmatrix} \begin{pmatrix} B^+ \\ B^- \end{pmatrix} \tag{付録 B.3}$$

ここで $\rho = q/k$。
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できる。

$$\mathrm{e}^{ika} \to \mathrm{e}^{ikb}\mathrm{e}^{-iKd} \tag{付録 B.4}$$
$$\mathrm{e}^{-ika} \to \mathrm{e}^{-ikb}\mathrm{e}^{-iKd} \tag{付録 B.5}$$

簡略のために以下の置換えをする。

$$\begin{aligned} H &= \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \\ \alpha &= \begin{pmatrix} 1 & 0 \\ 0 & \rho \end{pmatrix} \\ K_x &= \begin{pmatrix} \mathrm{e}^{ikx} & 0 \\ 0 & \mathrm{e}^{-ikx} \end{pmatrix} \\ Q_x &= \begin{pmatrix} \mathrm{e}^{iqx} & 0 \\ 0 & \mathrm{e}^{-iqx} \end{pmatrix} \\ \mathbf{W} &= \begin{pmatrix} W^+ \\ W^- \end{pmatrix} \\ \mathbf{B} &= \begin{pmatrix} B^+ \\ B^- \end{pmatrix} \end{aligned} \tag{付録 B.6}$$

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。

$$\begin{aligned} H\mathbf{W} - H\alpha\mathbf{B} &= 0 \\ \mathrm{e}^{-iKd}K_bH\mathbf{W} - Q_aH\alpha\mathbf{B} &= 0 \end{aligned} \tag{付録 B.7}$$

整理すると以下のような方程式が得られる。

$$\begin{pmatrix} E & \alpha \\ E & e^{iKd}H^{-1}K_b^{-1}Q_aH\alpha \end{pmatrix} \begin{pmatrix} \mathbf{W} \\ \mathbf{B} \end{pmatrix} = 0 \tag{付録 B.8}$$

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できる。

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

(付録 C.1)

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。

$$\begin{aligned} K_{dn} H \mathbf{W_n} &= Q_{dn} H \alpha \mathbf{B_n} \\ K_a K_{dn} H \mathbf{W}_{n+1} &= Q_a Q_{dn} H \alpha \mathbf{B}_n \end{aligned}$$

(付録 C.2)

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になる。

$$\mathbf{W}_{n+1} = H^{-1} K_{dn}^{-1} K_a^{-1} Q_a K_{dn} H \mathbf{W}_n$$

(付録 C.3)

ちなみに, 一般項は以下のようになる。

$$\mathbf{W}_n = H^{-1} \left(K_b Q_a \right)^n H \mathbf{W}_0$$

(付録 C.4)

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。

$$\begin{aligned} \mathbf{W}_n &= c_1 A_1 + c_2 A_2 \\ W_0 = A^{-n} W_n &= c_1 a_1^{-n} A_1 + c_2 a_2^{-n} A_2 \end{aligned}$$

(付録 C.5)

ここで, $n \to \infty$ の場合を考えると, $|a_1|^{-n} >> a_2^{-n}$ 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できる。

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 [19]。

Listing 7　___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
```

```
from QuantumSketchBook.schroedinger import Schroedinger


def plot(plotted, show=True, save=False, title="no_title", *args, **kwargs):
    return plotted.__plot__(show, save, title, *args, **kwargs)
```

Listing 8 mesh.py

```
from scipy import arange
from numbers import Real
import QuantumSketchBook.meta as meta


class Mesh(meta.Mesh):

    def __new__(cls, x_min: Real, x_max: Real, dx: Real,
                t_min: Real, t_max: Real, dt: Real)-> "Mesh":
        if not all(isinstance(x, Real) for x in (x_min, x_max, dx, t_min, t_max, dt)):
            raise TypeError("input␣should␣be␣a␣number")
        if not (x_min < x_max and t_min < t_max):
            raise ValueError("The␣min␣should␣be␣smaller␣than␣The␣max.␣")
        if dx <= 0 or dt <= 0:
            raise ValueError("The␣dx␣and␣dt␣should␣be␣positive.␣")
        if not (x_max - x_min > dx and t_max - t_min > dt):
            raise ValueError("too␣big␣dx␣or␣dt␣to␣make␣mesh.␣")
        return super().__new__(cls, x_min, x_max, dx, t_min, t_max, dt)

    @property
    def param(self):
        return self.x_min, self.x_max, self.dx, self.t_min, self.t_max, self.dt

    @property
    def x_vector(self):
        return arange(self.x_min, self.x_max, self.dx)

    @property
    def t_vector(self):
        return arange(self.t_min, self.t_max, self.dt)

    @property
    def x_num(self):
        return len(self.x_vector)

    @property
    def t_num(self):
        return len(self.t_vector)

    def __str__(self):
        x_string = "{}<x<{}␣␣(dx={})".format(self.x_min, self.x_max, self.dt)
        t_string = "{}<t<{}␣␣(dt={})".format(self.t_min, self.t_max, self.dx)
```

```
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 9　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␣over␣written␣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__,
                       Mesh.__class__))
27         return cls._context
28
29     @classmethod
30     def get_mesh(cls):
31         if cls._mesh is not None:
32             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)
```

```
        except MeshContextError:
            return True
        return False

    assert error_check()
    assert error_check(3)
    assert not error_check(mesh2)
    MeshContext().clean()
    with Context(-10, 0, 1, -10, 0, 1) as mm:
        assert mm == Mesh(-10, 0, 1, -10, 0, 1)

    try:
        MeshContext.get_mesh()
    except MeshContextError as e:
        assert e.args[0] == "not␣found␣any␣mesh␣in␣Context"
```

Listing 10 quantized.py

```
from QuantumSketchBook.context import MeshContext
from typing import Optional, TYPE_CHECKING
if TYPE_CHECKING:
    from QuantumSketchBook import Mesh


class Quantized:
    def __init__(self, mesh: Optional["Mesh"]=None):
        if mesh is not None:
            self.mesh: Mesh = mesh
        else:
            self.mesh: Mesh = MeshContext.get_mesh()
            MeshContext.add_observer(self)

    def update_mesh(self):
        self.mesh = MeshContext.get_mesh()


if __name__ == "__main__":
    mod = __import__("QuantumSketchBook.mesh")
    mesh1 = mod.my_mesh()
    mesh2 = mod.my_mesh(x_min=0)
    MeshContext(mesh1)
    a = Quantized()
    assert a.mesh == mesh1

    MeshContext.update_mesh(mesh2)
    assert Quantized().mesh == mesh2
    assert not a.mesh == mesh1
    assert a.mesh == mesh2
```

Listing 11 field.py

```
from QuantumSketchBook.quantized import Quantized
from scipy import array
from typing import TYPE_CHECKING, Optional
from matplotlib.pyplot import figure, axes
if TYPE_CHECKING:
    from QuantumSketchBook import Mesh


class Field(Quantized):

    def __init__(self, arg, mesh: Optional["Mesh"]=None):
        super().__init__(mesh=mesh)
        if hasattr(arg, "__iter__"):
            vec = array(tuple(arg))
            if not vec.size == self.mesh.x_num:
                raise ValueError("Length␣of␣input␣iterable␣must␣be␣equal␣to␣mesh.num")
            self.vector = vec
        elif callable(arg):
            self.vector = arg(self.mesh.x_vector)
        else:
            raise TypeError("the␣first␣argument␣should␣be␣iterable␣or␣callable")

    def __plot__(self, show, save, title, ylabel, values, *args, **kwargs):
        fig = figure()
        ax = axes()
        ax.plot(self.mesh.x_vector, values)
        ax.set_title(title)
        ax.set_xlabel("x")
        ax.set_ylabel(ylabel)
        fig.add_axes(ax)

        if save:
            fig.savefig(title + ".png")

        if show:
            fig.show()
        return fig


if __name__ == "__main__":
    import QuantumSketchBook as QSB
    from scipy import ndarray
    with QSB.Context(0, 10, 1, 0, 10, 1):
        assert Field(range(10)).vector.__class__ == ndarray
        assert Field(lambda x: x).vector.__class__ == ndarray
```

Listing 12 potential.py

```
from QuantumSketchBook.field import Field
from QuantumSketchBook.context import MeshContext
from scipy.sparse import dia_matrix
from scipy import array
from numbers import Real
from math import ceil, floor


class Potential(Field):

    def matrix(self):
        offset = [0]
        n = self.mesh.x_num
        mat = dia_matrix((self.vector, offset), shape=(n, n), dtype=complex)
        return mat.tocsr()

    def __add__(self, other):
        if not self.mesh == other.mesh:
            raise ValueError("meshes␣on␣{}␣and␣{}␣are␣not␣the␣same.␣")
        return Potential(self.vector + other.vector, mesh=self.mesh)

    def __mul__(self, other):
        if not isinstance(other, Real):
            raise TypeError("unsupported␣operand␣type(s)␣for␣*:'Potential'␣and␣'{}'".format(type(
                other)))
        return Potential(other * self.vector, mesh=self.mesh)

    def __sub__(self, other):
        return self.__add__(other.__mul__(-1))

    def __rmul__(self, other):
        return self.__mul__(other)

    def __eq__(self, other):
        return self.mesh == other.mesh and self.vector == other.vector

    def __plot__(self, show, save, title, *args, **kwargs):
        return super().__plot__(show, save, title, "V", self.vector)


def potential(arg):
    return Potential(arg)


def free():
    return potential(lambda x: 0)

```

```
def step(height: Real, distance: Real):
    if not all(isinstance(x, Real) for x in (height, distance)):
        raise TypeError
    v = array([height if distance <= x else 0 for x in MeshContext.get_mesh().x_vector])
    return potential(v)


def box(height: Real, distance: Real, barrier: Real):
    if 0 >= barrier:
        raise ValueError("barrier␣should␣be␣positive.␣")
    near = step(height, distance)
    far = step(height, distance + barrier)
    return near - far


def vacuum_kp(height, well, barrier):
    x_max = MeshContext.get_mesh().x_max
    period = well + barrier
    cycle = ceil(x_max / period) if x_max > 0 else 1
    gen = (box(height, i * period, barrier) for i in range(cycle))
    return sum(gen, free())


def kp_vacuum(height, well, barrier):
    x_min = MeshContext.get_mesh().x_min
    period = well + barrier
    cycle = -floor(x_min / period) if x_min < 0 else 1
    gen = (box(height, -(i + 1) * period, barrier) for i in range(cycle))
    return sum(gen, free())


def kp(height, well, barrier):
    v1 = vacuum_kp(height, well, barrier)
    v2 = kp_vacuum(height, well, barrier)
    return v1 + v2


if __name__ == "__main__":
    import QuantumSketchBook as QSB
    with QSB.Context(0, 10, 1, 0, 10, 1) as mesh:
        test1 = step(3, 1).vector
        test2 = step(2, 2).vector
        test3 = vacuum_kp(1, 1, 1).vector
        assert all(a == 0 for i, a in enumerate(test1) if i < 1)
        assert all(a == 3 for i, a in enumerate(test1) if i >= 1)
        assert all(b == 0 for i, b in enumerate(test2) if i < 2)
        assert all(b == 2 for i, b in enumerate(test2) if i >= 2)
        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 13　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 14　laplasian.py

```
1  from scipy import arange, array, zeros
```

```
from scipy.sparse import dia_matrix, csr_matrix, coo_matrix
from QuantumSketchBook.mesh import Mesh


class Laplasian:

    def __init__(self, mesh: Mesh):
        self.dx = mesh.dx
        self.x_num = mesh.x_num

    def _core_matrix(self) -> csr_matrix:
        det = 1 / (360 * self.dx ** 2)
        coeff = array([[4], [-54], [540], [-980], [540], [-54], [4]])
        std = det * coeff
        data = std.repeat(self.x_num + 1, axis=1)
        pad = arange(-3, 3 + 1)
        dia = dia_matrix((data, pad), shape=(self.x_num, self.x_num), dtype=complex)
        csr = dia.tocsr()
        return csr

    def matrix(self, boundary="free") -> csr_matrix:
        det = 1 / (360 * self.dx ** 2)
        csr = self._core_matrix()

        if boundary == "free":
            data = csr.data
            bound = array([-980, 1080, -108, 8, 540, -1034, 544, -54, 4, -54, 544]) * det
            data[:11] = bound
            data[-11:] = bound[::-1]
            matrix = csr_matrix((data, csr.indices, csr.indptr), dtype=complex)
            return matrix

        elif boundary == "fix":
            data = csr.data
            bound = array([-980, 0, 0, 0, 540, -926, 536, -54, 4, -54, 536]) * det
            data[:11] = bound
            data[-11:] = bound[::-1]
            matrix = csr_matrix((data, csr.indices, csr.indptr), dtype=complex)
            return matrix

        elif boundary == "period":
            n = self.x_num
            data = array([4, -54, 540, 4, -54, 4, 4, -54, 4, 540, -54, 4]) * det
            row = array([0, 0, 0, 1, 1, 2, n-3, n-2, n-2, n-1, n-1, n-1])
            col = array([n-3, n-2, n-1, n-2, n-1, n-1, 0, 0, 1, 0, 1, 2])
            coo = coo_matrix((data, (row, col)), shape=(n, n), dtype=complex)
            bound = coo.tocsr()
            matrix = csr + bound
            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　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 16　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
                    "\n", flush=True)
31              yield self.ode.integrate(self.mesh.t_vector[index])
32              index += 1
33
34      def solution(self):
35          if self._sol is None:
36              self._sol = zeros([self.mesh.t_num, self.mesh.x_num], dtype=complex)
37              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")
```

```
        above.set_ylabel("time(a.u.)")
        under.tick_params(labelleft="off")
        under.set_xlabel("space(a.u.)")

        above.pcolormesh(x_grid, t_grid, phi2, cmap="nipy_spectral_r")
        under.plot(x, potential_)

        if limit is not ():
            above.set_xlim(*limit)
            under.set_xlim(*limit)

        if save:
            fig.savefig(title + ".png")

        if show:
            fig.show()
        return fig
```

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] Nobelprize.org, (2018/1/20 閲覧),The Nobel Prize in Physics 1977, https://www.nobelprize.org/nobel_prizes/physics/laureates/1977/

[8] Nobelprize.org, (2018/1/20 閲覧),The Nobel Prize in Physics 1973, https://www.nobelprize.org/nobel_prizes/physics/laureates/1973/

[9] Anker, T., Albiez, M., Eiermann, B., Taglieber, M., & Oberthaler, M. K. (2004). Linear and nonlinear dynamics of matter wave packets in periodic potentials. Optics express, 12(1), 11-18.

[10] 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.

[11] Zhang, Z., Tong, P., Gong, J., & Li, B. (2012). Quantum hyperdiffusion in one-dimensional tight-binding lattices. Physical review letters, 108(7), 070603.

[12] 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

[13] Hufnagel, L., Ketzmerick, R., Kottos, T., & Geisel, T. (2001). Superballistic spreading of wave packets. Physical Review E, 64(1), 012301.
[14] 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.
[15] 前田雅人 (2014). 量子波束の跳ね返りシミュレーション. 卒業論文
[16] 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
[17] 後藤敬佑 (2015). 確率力学に基づいた量子跳ね返り時間のシミュレーション. 卒業論文
[18] 太箸良介 (2015). 相対論的粒子の量子跳ね返り時間. 卒業論文
[19] https://github.com/tmaeda11235/schroedinger_solver　で開発版のソースコードを公開している
[20] 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.
[21] python.org, (2018/1/19 閲覧) Python 3.6.3 ドキュメント, https://docs.python.jp/3/index.html
[22] scipy.org, (2018/1/19 閲覧), Scipy Lecture Notes, http://www.scipy-lectures.org/
[23] pyton.org, (2018/1/19 閲覧), PEP 8 – Style Guide for Python Code, https://www.python.org/dev/peps/pep-0008/
[24] pyton.org, (2018/1/19 閲覧), PEP 484 – Type Hints, https://www.python.org/dev/peps/pep-0484/
[25] pyton.org, (2018/1/19 閲覧), PEP 20 – The Zen of Python, https://www.python.org/dev/peps/pep-0020/
[26] 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
[27] Siegle, P., Goychuk, I., & Hänggi, P. (2010). Origin of hyperdiffusion in generalized brownian motion. Physical review letters, 105(10), 100602.