調和振動子基底展開による
時間依存Schrödinger方程式の
数値解法

2013年2月

福井大学 工学部 物理工学科
平成21年度入学
09380337　　台　亮嗣

目 次

第1章　序論

19 世紀まで，ニュートンにより始められた力学と，マクスウェルが確立した電磁気学をはじめとする，古典物理学により宇宙の森羅万象はすべて説明できると考えられていた．これを根本から覆したのが，19 世紀の最後の年に登場した量子力学である．量子力学は，非相対論的な範囲に話を限れば，理論体系は完成されており，電子工学や化学その他の様々な分野に応用され，今日ではもはや物理学の新理論ではなくなっている．

我々の日常生活からは，想像し難い状態ベクトルと呼ばれる抽象的な量を用いなければ，物質を構成する原子などの科学の根幹ともいえる極微な世界は記述できないというところに，ある種の哲学的なおもしろさも，量子力学にはあるように思う．

量子力学はその解釈の難しさはあるが，ニュートン力学で登場するニュートンの運動方程式と同様に，基本的には微分方程式を扱うことで，系の状態を理解することができる．1 階の微分方程式である Schrödinger 方程式を数値的に解き，系の状態の時間発展を求めていく過程で，量子力学の基礎を学び直すことを本研究のひとつの目的としている．

本研究では，一次元ポテンシャル内の一粒子の量子力学的状態ベクトルの時間発展を，時間に依存する Schrödinger 方程式を解くことで求める．その手法として，まず状態ベクトルを調和振動子基底で展開する．基底で状態を展開することで時間に関わる部分は展開係数のみになるため，この展開系数の時間発展を Runge-Kutta 法で数値的に解けば状態ベクトルの時間発展も求まる．一次元量子系は実空間表示や平面波展開などの手法で扱われることが多いのだが，本研究では巨大な次元の調和振動子基底で一次元量子系を扱うことができるかどうかを論ずる．

第2章　量子力学の基礎

この章では量子力学の基本方程式である Schrödinger 方程式について述べたあと，その解を記述するためのいくつかの概念を説明する．

2.1　Schrödinger 方程式

2.1.1　時間に依存する Schrödinger 方程式

ポテンシャル $V(x,t)$ で表される力を受けて運動する質量 m の粒子の量子力学的状態を表す状態ベクトル $|\,\psi(x,t)\,\rangle$ は，時間に依存する Schrödinger 方程式

$$i\hbar\frac{\partial}{\partial t}|\,\psi(x,t)\,\rangle = \hat{H}|\,\psi(x,t)\,\rangle \tag{2.1}$$

の解として与えられる．ここで $\hbar$ はプランク定数の 2π 分の 1 倍であり，また $\hat{H}$ は系のエネルギーを表す自己共役演算子で，ハミルトニアンと呼ばれ，

$$\hat{H} = -\frac{\hbar^2}{2m}\frac{\partial^2}{\partial x^2} + V(x,t) \tag{2.2}$$

と表される．

Schrödinger 方程式は，時間に関して 1 階の微分方程式だから，初期状態（時刻 $t=0$ における状態ベクトル $|\,\psi(x,0)\,\rangle$）を与えれば，解（$t\neq 0$ における状態ベクトル $|\,\psi(x,t)\,\rangle$）が一意的に定まる．なお古典論でも量子論と同様に，系が閉じている限り（外界から予測不能な影響を受けることがなければ），時刻 $t=0$ における状態を与えれば，$t\neq 0$ における状態が一意的に定まるが，古典力学の運動方程式は時間に関して 2 階の微分方程式なので，状態は力学変数の時間微分をも指定しないと定まらない．

2.1.2　時間に依存しない Schrödinger 方程式

ハミルトニアン $\hat{H}$ が時間 t を含まない，即ちポテンシャルエネルギーが $V(x,t)=V(x)$ と表される場合には，定常状態と呼ばれる特別な状態が存在する．定常状態とは状態ベクトルが

$$|\,\psi(x,t)\,\rangle = e^{-i\omega t}|\,\varphi(x)\,\rangle \tag{2.3}$$

のように表記できる状態であり，時間には位相因子 $e^{-i\omega t}$ を通じてのみ依存する．状態ベクトルの空間部分である $|\,\varphi(x)\,\rangle$ は次式のような時間に依存しない Schrödinger 方程式に従う．

$$\left(-\frac{\hbar^2}{2m}\frac{\partial^2}{\partial x^2}+V(x)\right)|\,\varphi(x)\,\rangle=\varepsilon|\,\varphi(x)\,\rangle. \tag{2.4}$$

ここで ε はハミルトニアン $\hat{H}$ の固有値でエネルギー固有値と呼ばれる．

2.2 基底による展開

状態ベクトルを基底で展開するとはどういうことであるかを，以下では順を追って説明する．

2.2.1 正規直交基底

複素ベクトル空間（$\mathbf{C}$ 上の線形空間）に内積が定義されている場合は，元々の基底の適当な線形結合をとることで，互いに直交するように基底を選び直すことがシュミットの直交化法などによりいつでも可能である．基底を構成する状態ベクトルを自然数 $n=1,2,3\cdots$ で番号付けして $|\,n\,\rangle$ と表し，$|\,n\,\rangle$ と $|\,n'\,\rangle$ の内積を $\langle\,n\,|\,n'\,\rangle$ と表すとき，基底を次の条件を満足するように選ぶと便利である．

$$\langle\,n\,|\,n'\,\rangle=\left\{\begin{array}{ll}1 & (n=n')\\ 0 & (n\neq n')\end{array}\right\}=\delta_{n,n'}. \tag{2.5}$$

この条件を満たす状態ベクトルの集合 $\{|\,n\,\rangle\}$ を正規直交基底（正規直交系）と呼ぶ．

2.2.2 完全性

複素数体 $\mathbf{C}$ 上の完備な内積空間 $\mathcal{H}$ を複素ヒルベルト空間と呼ぶ．内積空間 $\mathcal{H}$ が完備であるとは，$\mathcal{H}$ 内のどんなベクトル列 $|\,\psi_1\,\rangle,|\,\psi_2\,\rangle,\cdots$ でも，もしそれが

$$\lim_{n,m\to\infty}\|\,|\,\psi_n\,\rangle-|\,\psi_m\,\rangle\|=0 \tag{2.6}$$

を満たすベクトル列であれば，必ずその収束先 $|\,\psi\,\rangle$，つまり

$$\lim_{n\to\infty}\|\,|\,\psi_n\,\rangle-|\,\psi\,\rangle\|=0 \tag{2.7}$$

を満たすベクトル $|\,\psi\,\rangle$ が $\mathcal{H}$ 内に存在することである．ここで $\|\cdots\|$ は状態ベクトルのノルムを表す．

どんなベクトル $|\,\psi\,\rangle\in\mathcal{H}$ も，同じ $\mathcal{H}$ 上のベクトルの集合 $\{|\,n\,\rangle\}$ の線形結合として表せるとき，$\{|\,n\,\rangle\}$ は $\mathcal{H}$ の完全系を成すという．ひとつの自己共役演算子の固有ベクトルをもれなく集めてくれば，それは完全系を成すことが知られている．従って，自己

共役演算子 $\hat{A}$ の固有ベクトルを $|\,n\,\rangle$ とすると，任意のベクトル $|\,\psi\,\rangle \in \mathcal{H}$ は，n が離散的な場合は

$$|\,\psi\,\rangle = \sum_{n=1}^{\infty} \psi_n |\,n\,\rangle \tag{2.8}$$

のように，適当な係数 $\psi_n \in \mathbf{C}$ を用いて展開できる．

なお,「完備」と「完全」は共に”complete”の訳語であり,「完全系」でなく「完備系」と訳す流儀もあることを申し添えておく．

2.3 ユニタリー変換

ベクトル空間からベクトル空間への 1 体 1 対応の線形写像で，ノルムを変えないものをユニタリー変換と呼ぶ．

時間に依存する Schrödinger 方程式 (2.1) 式の解は，次式のように時刻 $t = 0$ における状態ベクトル $|\,\psi(0)\,\rangle$ を時間発展演算子と呼ばれるユニタリー演算子でユニタリー変換した形をとる．

$$|\,\psi(t)\,\rangle = \hat{U}(t)|\,\psi(0)\,\rangle. \tag{2.9}$$

状態ベクトルは, 時間とともに連続的にユニタリー変換される（観測過程を除く）．(2.2) 式のハミルトニアンで $V(x,t) = V(x)$(ポテンシャルが時間に依存しない場合) と書けるときは,

$$\hat{U}(t) = \exp\left(\frac{\hat{H}}{i\hbar}\right) \tag{2.10}$$

と形式的に書き表すことができる．

2.4 虚時間発展法

実時間 t に虚数単位 i をかけたものを虚時間と呼ぶ．

(2.1) 式の Schrödinger 方程式を考える．

$$i\hbar\frac{\partial}{\partial t}|\,\psi(x,t)\,\rangle = \hat{H}|\,\psi(x,t)\,\rangle. \tag{2.11}$$

(2.8) 式より，任意の状態ベクトルは固有ベクトルで展開できるため，初期状態を

$$|\,\psi(x,0)\,\rangle = \sum_{n=1}^{\infty} \psi_n |\,n\,\rangle, \tag{2.12}$$

$$\hat{H}|\,n\,\rangle = \varepsilon_n |\,n\,\rangle \quad (n = 1, 2\cdots) \tag{2.13}$$

$$\varepsilon_1 \leq \varepsilon_2 \leq \varepsilon_3 \leq \cdots,$$

と展開すると，(2.11) 式の解は

$$|\,\psi(x,t)\,\rangle = \sum_{n=1}^{\infty} \psi_n \exp\left(-i\frac{\varepsilon_n}{\hbar}t\right)|\,n\,\rangle \tag{2.14}$$

と記述される．ここで，時間 t の虚数単位倍を τ で表し（即ち $\tau = it,\ \ t,\tau$ は実数），(2.14) 式を τ（虚時間と呼ばれる）で書き換えると

$$|\,\psi(x,\tau)\,\rangle = \sum_{n=1}^{\infty} \psi_n \exp\left(-\frac{\varepsilon_n}{\hbar}\tau\right)|\,n\,\rangle \tag{2.15}$$

を得る．したがって充分大きな虚時間 τ に対しては，状態ベクトルの基底 $|\,1\,\rangle$ の展開係数に対して基底 $|\,n\,\rangle$ $(n>0)$ の展開係数は

$$\frac{\psi_n \exp\left(-\frac{\varepsilon_n}{\hbar}\tau\right)}{\psi_1 \exp\left(-\frac{\varepsilon_1}{\hbar}\tau\right)} = \frac{\psi_n}{\psi_1}\exp\left\{-\frac{\tau}{\hbar}(\varepsilon_n-\varepsilon_1)\right\} \longrightarrow 0 \qquad (\tau\to\infty) \tag{2.16}$$

のように高位の無限小となるので $\tau\to\infty$ の極限で

$$|\,\psi(x,\tau)\,\rangle \propto |\,n\,\rangle \tag{2.17}$$

となる．このことは虚時間発展とともに状態ベクトルが最低エネルギーの状態に近づいていくことを意味する．本研究では，これを利用して初期状態の波動関数を準備する．

2.5 Runge-Kutta 法

本研究では，Runge-Kutta 法で時間発展演算子 $\hat{U}(t)$ の作用を数値的に扱う．ここでは Runge-Kutta 法について説明する．

関数 $y=f(x)$ の一階微分 y' が $y'=F(x,y)$ の形の常微分方程式で与えられ，$y_0 = f(x_0)$ の初期条件のもとに $y=f(x)$ を求める初期値問題を考える．

最も簡単な方法は，$y_0=f(x_0)$ から始めて，近似式

$$f(x+h) \approx f(x) + f'(x)h \tag{2.18}$$

を漸化式として $f(x_0+h), f(x_0+2h), \cdots$ を順に求めていく Euler 法（誤差が h の 1 乗に比例する.）である．

計算をより高精度なものにするには，(2.18) 式のような簡単な近似式の代わりに

$$f(x+h) \approx f(x) + f'(x)h + \frac{1}{2!}f''(x)h^2 + \frac{1}{3!}f'''(x)h^3 + \cdots \tag{2.19}$$

のような Taylor 展開を使えばよいが，Taylor 展開を使わずに，いくつかの場所で評価した $F(x,y)$ を混ぜ合わせて同じ結果を得る方法が Runge-Kutta 法である．本研究で

は次式のような「4 次の $\frac{1}{6}$ 公式」を使用した.

$$\vec{y} = \vec{y_0} + \frac{1}{6}(\vec{k_1} + 2\vec{k_2} + 2\vec{k_3} + \vec{k_4}) + \mathcal{O}(h^5) \tag{2.20}$$
$$\vec{k_1} = \vec{f}(x_0, \vec{y_0})$$
$$\vec{k_2} = f(x_0 + \frac{h}{2}, \vec{y_0} + \frac{h}{2}\vec{k_1})$$
$$\vec{k_3} = \vec{f}(x_0 + \frac{h}{2}, \vec{y_0} + \frac{h}{2}\vec{k_2})$$
$$\vec{k_4} = \vec{f}(x_0 + h, \vec{y_0} + h\vec{k_3}).$$

これは結果の誤差が h の 4 乗に比例する．(2.20) 式を

$$|\,\psi(x,t)\,\rangle = \sum_{n=1}^{\infty} \psi_n(t)|\,n\,\rangle \tag{2.21}$$

を求めるのに利用するには,

$$x = t, \quad x_0 = t_0 \tag{2.22}$$
$$\vec{y} = (\psi_1(t), \psi_2(t), \psi_3(t), \cdots) \tag{2.23}$$

のように対応させる.

第3章　調和振動子基底

調和振動子では粒子は外場（ポテンシャルエネルギー）によって空間のある領域に局在した束縛状態にある．ここでは，一次元調和振動子の振る舞いについて要点を整理して示し，重要な演算子を具体的に行列表示で表す.

3.1　一次元調和振動子

x 軸上で原点 O からの距離に比例する引力

$$F(x) = -kx = -m\omega^2 x \tag{3.1}$$

は，ポテンシャル

$$V(x) = \frac{1}{2}m\omega^2 x^2 \tag{3.2}$$

から導かれる．このような力を受けている粒子のハミルトニアンは

$$\hat{H} = -\frac{\hbar^2}{2m}\frac{\partial^2}{\partial x^2} + \frac{1}{2}m\omega^2 x^2 \tag{3.3}$$

と記述される．したがって，この粒子に対する時間に依存しない Schrödinger 方程式は

$$\left\{-\frac{\hbar^2}{2m}\frac{\partial^2}{\partial x^2} + \frac{1}{2}m\omega^2 x^2\right\} |\,\varphi(x)\,\rangle = \varepsilon |\,\varphi(x)\,\rangle \tag{3.4}$$

と表記される．ここで，式を簡略化し，計算を容易にするために無次元の変数

$$\xi = \sqrt{\frac{m\omega}{\hbar}} = \frac{x}{b}, \tag{3.5}$$

$$\lambda = \frac{2\varepsilon}{\hbar\omega} \tag{3.6}$$

を導入すると（$b = \sqrt{\frac{\hbar}{m\omega}}$ は調和振動子長と呼ばれる），(3.4) 式は，次のように書き換えられる.

$$\left(-\frac{\partial^2}{\partial \xi^2} + \xi^2\right) |\,\varphi(\xi)\,\rangle = \lambda |\,\varphi(\xi)\,\rangle \tag{3.7}$$

ある基底 $|\,n\,\rangle$ における状態ベクトル $|\,\varphi(\xi)\,\rangle$ の波動関数 $\varphi_n(\xi)$ は，

$$\langle\, n\,|\,\varphi(\xi)\,\rangle \tag{3.8}$$

で得られる．この波動関数は，エルミート多項式と $\exp\left(-\frac{\xi^2}{2}\right)$ の積に比例する形をとる．

$$\varphi_n(\xi) \propto H_n(\xi)\exp\left(-\frac{\xi^2}{2}\right) \tag{3.9}$$
$$\lambda_n = 2n+1, \quad n = 0, 1, 2, \cdots.$$

エルミート多項式 $H_n(\xi)$ は

$$H_n(x) = (-1)^n \exp(x^2)\frac{d^n}{dx^n}\exp(-x^2) \tag{3.10}$$

によって定義され，

$$H_{n+1}(\xi) = 2\xi H_n(\xi) - 2nH_{n-1}, \tag{3.11}$$
$$H_0 = 1, \tag{3.12}$$
$$H_1(\xi) = 2\xi, \tag{3.13}$$

$$\left(\frac{\partial^2}{\partial\xi^2} - 2\xi\frac{\partial}{\partial\xi} + 2n\right)H_n(\xi) = 0, \tag{3.14}$$

$$\int_{-\infty}^{\infty} H_n(\xi)H_m(\xi)\exp(-\xi^2)d\xi = \begin{cases} 0 & n \neq m \\ 2^n!\sqrt{\pi} & n = m \end{cases} \tag{3.15}$$

などの関係を満たす関数の一群である．

変数を x と λ にもどし，(3.15) によって規格化した関数をつくると，

$$\psi_n(x) = \left(\frac{\sqrt{2m\omega/h}}{2^n n!}\right)^{1/2} H_n\left(\sqrt{\frac{m\omega}{\hbar}}x\right)\exp\left(-\frac{m\omega}{2\hbar}x^2\right), \tag{3.16}$$

$$\varepsilon_n = (n+\frac{1}{2})\hbar\omega, \quad n = 0, 1, 2, \cdots \tag{3.17}$$

が得られる．

ポテンシャルエネルギーが，(3.2) で表されるとき，粒子は $x = 0$ のまわりに局在する（図 3.1 参照）．波動関数 $\varphi_n(\xi)$ は状態 n で n 個のノードを持ち，エネルギー固有値は $\hbar\omega$ のとびとびの値を持つ．

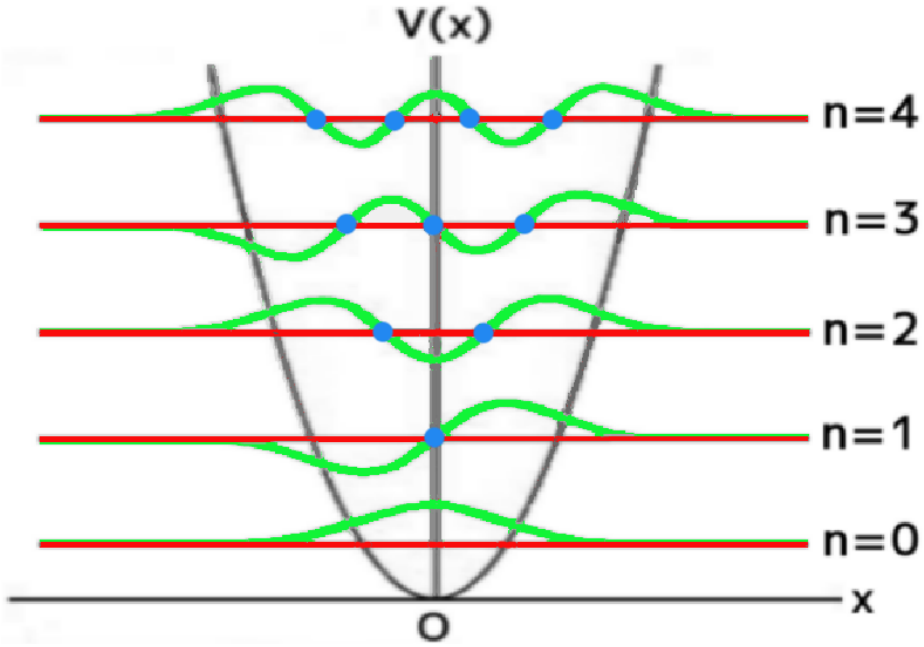

図 3.1: 一次元調和振動子の定常状態での波動関数

3.2 調和振動子の行列表示

一次元調和振動子のハミルトニアン

$$\hat{H} = -\frac{\hbar^2}{2m}\frac{\partial^2}{\partial x^2} + \frac{1}{2}m\omega^2 x^2 = \frac{\hbar\omega}{2}\left(-\frac{\partial^2}{\partial \xi^2} + \xi^2\right) \tag{3.18}$$

は，

$$\hat{H}|\,n\,\rangle = \hbar\omega(n + \frac{1}{2})|\,n\,\rangle \tag{3.19}$$

に従う $\hat{H}$ の固有ベクトル $|\,n\,\rangle$ を基底とする表現で，次のような対角行列で表わされる．

$$\hat{H} = \sqrt{\frac{\hbar}{2m\omega}}\begin{pmatrix} \frac{1}{2}\hbar\omega & 0 & 0 & 0 & \cdots \\ 0 & \frac{3}{2}\hbar\omega & 0 & 0 & \cdots \\ 0 & 0 & \frac{5}{2}\hbar\omega & 0 & \cdots \\ 0 & 0 & 0 & \frac{7}{2}\hbar\omega & \cdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \end{pmatrix}. \tag{3.20}$$

上昇演算子（生成演算子）$a^\dagger$，下降演算子（消滅演算子）a を

$$a^\dagger = \frac{1}{\sqrt{2}}\left(\xi - \frac{\partial}{\partial \xi}\right) = \sqrt{\frac{m\omega}{2\hbar}}\left(x - \frac{\hbar}{m\omega}\frac{\partial}{\partial x}\right), \tag{3.21}$$

$$a = \frac{1}{\sqrt{2}}\left(\xi + \frac{\partial}{\partial \xi}\right) = \sqrt{\frac{m\omega}{2\hbar}}\left(x + \frac{\hbar}{m\omega}\frac{\partial}{\partial x}\right) \tag{3.22}$$

と書くと，基底 $|\,n\,\rangle$ は

$$a^\dagger|\,n\,\rangle = \sqrt{n+1}|\,n+1\,\rangle, \tag{3.23}$$

$$a|\,n\,\rangle = \sqrt{n}|\,n-1\,\rangle \tag{3.24}$$

により構成される．これ等の等式の両辺と $|\,m\,\rangle$ との内積をとると，

$$\langle\,m\,|a^\dagger|\,n\,\rangle = \sqrt{n+1}\delta_{m,n+1}, \tag{3.25}$$

$$\langle\,m\,|a|\,n\,\rangle = \sqrt{n}\delta_{m,n-1} \tag{3.26}$$

となり $a^\dagger|\,n\,\rangle$ と $a|\,n\,\rangle$ の行列要素が得られる．したがって $a^\dagger$ と a の行列表記は次のようになる．

$$a^\dagger = \begin{pmatrix} 0 & 0 & 0 & 0 & \cdots \\ \sqrt{1} & 0 & 0 & 0 & \cdots \\ 0 & \sqrt{2} & 0 & 0 & \cdots \\ 0 & 0 & \sqrt{3} & 0 & \cdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \end{pmatrix}, \tag{3.27}$$

$$a = \begin{pmatrix} 0 & \sqrt{1} & 0 & 0 & \cdots \\ 0 & 0 & \sqrt{2} & 0 & \cdots \\ 0 & 0 & 0 & \sqrt{3} & \cdots \\ 0 & 0 & 0 & 0 & \cdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \end{pmatrix}. \tag{3.28}$$

(3.21)，(3.22) より ξ と $\frac{\partial}{\partial\xi}$ は

$$\xi = \frac{1}{\sqrt{2}}\left(a + a^\dagger\right), \quad \frac{\partial}{\partial\xi} = \frac{1}{\sqrt{2}}\left(a - a^\dagger\right) \tag{3.29}$$

と表され，(3.5) より x と $\frac{\partial}{\partial x}$ も $a^\dagger$ と a で次のように表される．

$$x = \sqrt{\frac{\hbar}{2m\omega}}\left(a + a^\dagger\right), \tag{3.30}$$

$$\frac{\partial}{\partial x} = \sqrt{\frac{m\omega}{2\hbar}}\left(a - a^\dagger\right). \tag{3.31}$$

このようにして，下記の x と $\frac{\partial}{\partial x}$ の行列表示を得る．

$$x = \sqrt{\frac{\hbar}{2m\omega}} \begin{pmatrix} 0 & \sqrt{1} & 0 & 0 & \cdots \\ \sqrt{1} & 0 & \sqrt{2} & 0 & \cdots \\ 0 & \sqrt{2} & 0 & \sqrt{3} & \cdots \\ 0 & 0 & \sqrt{3} & 0 & \cdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \end{pmatrix}, \tag{3.32}$$

$$\frac{\partial}{\partial x} = \sqrt{\frac{m\omega}{2\hbar}} \begin{pmatrix} 0 & \sqrt{1} & 0 & 0 & \cdots \\ -\sqrt{1} & 0 & \sqrt{2} & 0 & \cdots \\ 0 & -\sqrt{2} & 0 & \sqrt{3} & \cdots \\ 0 & 0 & -\sqrt{3} & 0 & \cdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \end{pmatrix}. \tag{3.33}$$

x^n の行列要素 $\langle\, i \,|x^n|\, j \,\rangle$ は，完全性 $\sum_{k_n} |\, k_n \,\rangle\langle\, k_n \,| = 1$ を利用して，以下のように x^1 の行列表示の積として求めることができ，結果として x^n を簡単な疎行列として表すことができる．

$$\langle\, i \,|x^n|\, j \,\rangle = \sum_{k_1,k_2\cdots k_{n-1}} \langle\, i \,|x|\, k_1 \,\rangle\langle\, k_1 \,|x|\, k_2 \,\rangle\langle\, k_2 \,|x|\cdots|\, k_{n-1} \,\rangle\langle\, k_{n-1} \,|x|\, j \,\rangle. \tag{3.34}$$

本研究で使用する，$\frac{\partial^2}{\partial x^2}$，$x^0$，$x^1$，$x^2$，$x^3$，$x^4$，$x^5$，$x^6$ の行列要素を以下に示す．

$$\langle\, i \,|\frac{\partial^2}{\partial x^2}|\, j \,\rangle = \frac{m\omega}{\hbar}\left\{ \frac{\sqrt{n(n-1)}}{2}\delta_{i,j-2} - \left(n+\frac{1}{2}\right)\delta_{i,j} + \frac{\sqrt{(n+1)(n+2)}}{2}\delta_{i,j+2} \right\}, \tag{3.35}$$

$$\langle\, i \,|1|\, j \,\rangle = \delta_{i,j}, \tag{3.36}$$

$$\langle\, i \,|x|\, j \,\rangle = \left(\frac{\hbar}{m\omega}\right)^{1/2}\left\{ \sqrt{\frac{n}{2}}\delta_{i,j-1} + \sqrt{\frac{n+1}{2}}\delta_{i,j+1} \right\}, \tag{3.37}$$

$$\langle\, i \,|x^2|\, j \,\rangle = \left(\frac{\hbar}{m\omega}\right)^{2/2}\left\{ \frac{\sqrt{n(n-1)}}{2}\delta_{i,j-2} + \left(n+\frac{1}{2}\right)\delta_{i,j} + \frac{\sqrt{(n+1)(n+2)}}{2}\delta_{i,j+2} \right\}, \tag{3.38}$$

$$\begin{aligned}\langle\, i \,|x^3|\, j \,\rangle = \left(\frac{\hbar}{m\omega}\right)^{3/2}\Bigg\{ & \frac{1}{2}\sqrt{\frac{n(n-1)(n-2)}{2}}\delta_{i,j-3} + \frac{3n}{2}\sqrt{\frac{n}{2}}\delta_{i,j-1} \\ & + \frac{3(n+1)}{2}\sqrt{\frac{n+1}{2}}\delta_{i,j+1} + \frac{1}{2}\sqrt{\frac{(n+1)(n+2)(n+3)}{2}}\delta_{i,j+3} \Bigg\},\end{aligned} \tag{3.39}$$

$$\begin{aligned}\langle\, i \,|x^4|\, j \,\rangle = \left(\frac{\hbar}{m\omega}\right)^{4/2}\Bigg\{ & \sqrt{\frac{n(n-1)(n-2)(n-3)}{4}}\delta_{i,j-4} + \left(n-\frac{1}{2}\right)\sqrt{n(n-1)}\delta_{i,j-2} \\ & + \frac{3}{4}\left(n^2+(n+1)^2\right)\delta_{i,j} + \left(n-\frac{3}{2}\right)\sqrt{(n+1)(n+2)}\delta_{i,j+2} \\ & + \frac{\sqrt{(n+1)(n+2)(n+3)(n+4)}}{4}\delta_{i,j+4} \Bigg\},\end{aligned} \tag{3.40}$$

$$\langle\, i\,|x^5|\,j\,\rangle = \left(\frac{\hbar}{m\omega}\right)^{5/2}\left\{\frac{1}{4}\sqrt{\frac{n(n-1)(n-2)(n-3)(n-4)}{2}}\delta_{i,j-5}\right.$$
$$+\frac{5(n-1)}{4}\sqrt{\frac{n(n-1)(n-2)}{2}}\delta_{i,j-3}+\frac{5(2n^2+1)}{4}\sqrt{\frac{n}{2}}\delta_{i,j-1}$$
$$+\frac{5(2n^2+4n+1)}{4}\sqrt{\frac{n+1}{2}}\delta_{i,j+1}+\frac{5(n+2)}{4}\sqrt{\frac{(n+1)(n+2)(n+3)}{2}}\delta_{i,j+3}$$
$$\left.+\frac{1}{4}\sqrt{\frac{(n+1)(n+2)(n+3)(n+4)(n+5)}{2}}\delta_{i,j+5}\right\}, \tag{3.41}$$

$$\langle\, i\,|x^6|\,j\,\rangle = \left(\frac{\hbar}{m\omega}\right)^{6/2}\left\{\frac{1}{8}\sqrt{n(n-1)(n-2)(n-3)(n-4)(n-5)}\delta_{i,j-6}\right.$$
$$+\frac{6n-9}{8}\sqrt{n(n-1)(n-2)(n-3)}\delta_{i,j-4}+\frac{15(n^2-n+1)}{8}\sqrt{n(n-1)}\delta_{i,j-2}$$
$$+\frac{5(4n^3+6n^2+8n+3)}{8}\delta_{i,j}+\frac{15(n^2+3n+3)}{8}\sqrt{(n+1)(n+2)}\delta_{i,j+2}$$
$$+\frac{6n+15}{8}\sqrt{(n+1)(n+2)(n+3)(n+4)}\delta_{i,j+4}$$
$$\left.+\frac{1}{8}\sqrt{(n+1)(n+2)(n+3)(n+4)(n+5)(n+6)}\delta_{i,j+6}\right\}. \tag{3.42}$$

(3.35)〜(3.42) 式が間違いないことは，(3.34) 式と数値的に一致する結果を与えることにより確認済である．

第4章　数値計算の精度

数値計算に用いたパラメータと精度の関係について論じる．第4〜5章では，$\hbar = m = 1$ となる単位系を用いる．

4.1　基底の切断（truncation）

本来ならば，基底の状態番号 n は $n = 0, 1, 2 \cdots \infty$ と無限大まで取り得る．しかし数値計算で無限大まで考えることはできないため，本研究では n の最大値 $n_{\max}$ が充分大きいが有限であるとして計算を行う．

テスト用のポテンシャルエネルギーとして

$$V(x) = \left(\frac{x}{31.6} - 1.0\right)^6 \tag{4.1}$$

をとり，初期状態を $t = 0$ で $x = 0$ 付近に局在した波束とする（次章で初期状態の作り方を記述してある）．これを実時間発展させ，ある時刻 t(ここでは $t = 10.0$) でのエネルギー期待値 E の値が変化しなくなるまで（差が充分小さくなるまで）$n_{\max}$ の値を増やしていき，計算精度上，充分大きいといえる．$n_{\max}$ の値を決める．図 4.1 に $t = 10.0$ での $n_{\max} = 5000$ のエネルギー期待値との差を $n_{\max}$ に対してプロットした．図 4.1 より $n_{\max}$ は 1000 で充分であると判断し，本研究では $n_{\max} = 1000$ とした．調和振動子のパラメータとして，調和振動子長（$b = \sqrt{\frac{\hbar}{m\omega}}$）がある．$n$ の最大値を $n_{\max}$ としたときの，ポテンシャルの x 座標の広がり $x_{\max}$ は，調和振動子長 b を用いて

$$x_{\max} = b\sqrt{2n_{\max} + 1} \tag{4.2}$$

と表され，確保したい x 座標の広がりを得るのに，充分大きな $n_{\max}$ と bが必要になる．これについては，(4.1) 式のポテンシャルに対し，$n_{\max} = 1000$ を用いる場合には，値として $b = 1.0$ が計算精度上充分大きいといえることを確認した．

4.2　誤差の微小時間に対する依存性

Euler 法では $n_{\max}$ が大きいと，dt を小さくとらなければならないことが知られている．4 次の Runge-Kutta 法においても同様の傾向があることが予想される．
dt の値が充分小さければ，計算による誤差は無視できるほど小さい（表 4.1 参照）．本研究では，微小時間 $dt = 1.0 \times 10^{-6}$ で，充分な精度を与えるとみなしている．

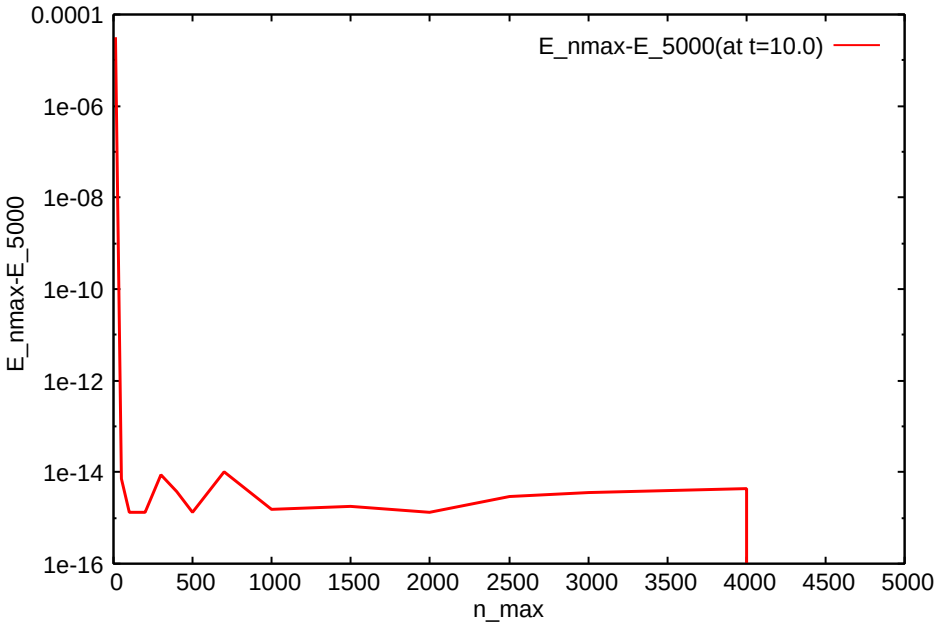

図 4.1: $t = 10.0$ における $n_{\max}$ のエネルギー期待値と $n_{\max} = 5000$ のエネルギー期待値との差

dt	$t = 10.0$ でのエネルギー期待値 E
1.0×10^{-3}	1.1374690988184457
1.0×10^{-4}	1.1374690988184430
1.0×10^{-5}	1.1374690988184470
1.0×10^{-6}	1.1374690988184375

表 4.1: 微小時間 dt と $t = 10.0$ でのエネルギー期待値 E

4.3　エネルギー期待値の保存

ある物理量 A を表す演算子 $\hat{A}$ とハミルトニアン $\hat{H}$ が交換するときには，(ただし $\hat{A}$ が時間に依存する場合を除く）物理量 A は保存される．

$\hat{H}$ と $\hat{H}$ の交換関係は

$$[\hat{H}, \hat{H}] = 0 \tag{4.3}$$

となり，$\hat{H}$ が時間に依存しなければエネルギーは保存される．したがって，実時間発展においてエネルギーの期待値は一定である．

図 4.2 は，ポテンシャルエネルギーを (4.1) 式のものにとり，$n_{\max} = 1000$ での実時間発展によるエネルギー期待値の変動を時間に対してプロットしたものであり，一定に保たれていることが分かる．しかし，あくまで数値計算であるため実は完全に一定というわけではない．$t = 0$ での値（$E(0) = 1.1374690988184504$）からのずれを時間の関数としてプロットしたものを図 4.3 に示す．この結果から $t = 0$ での値からのずれは無視できるほど小さく，数値的実時間発展におけるエネルギー期待値は高い精度で時間に依存せず一定とみなせることがわかる．

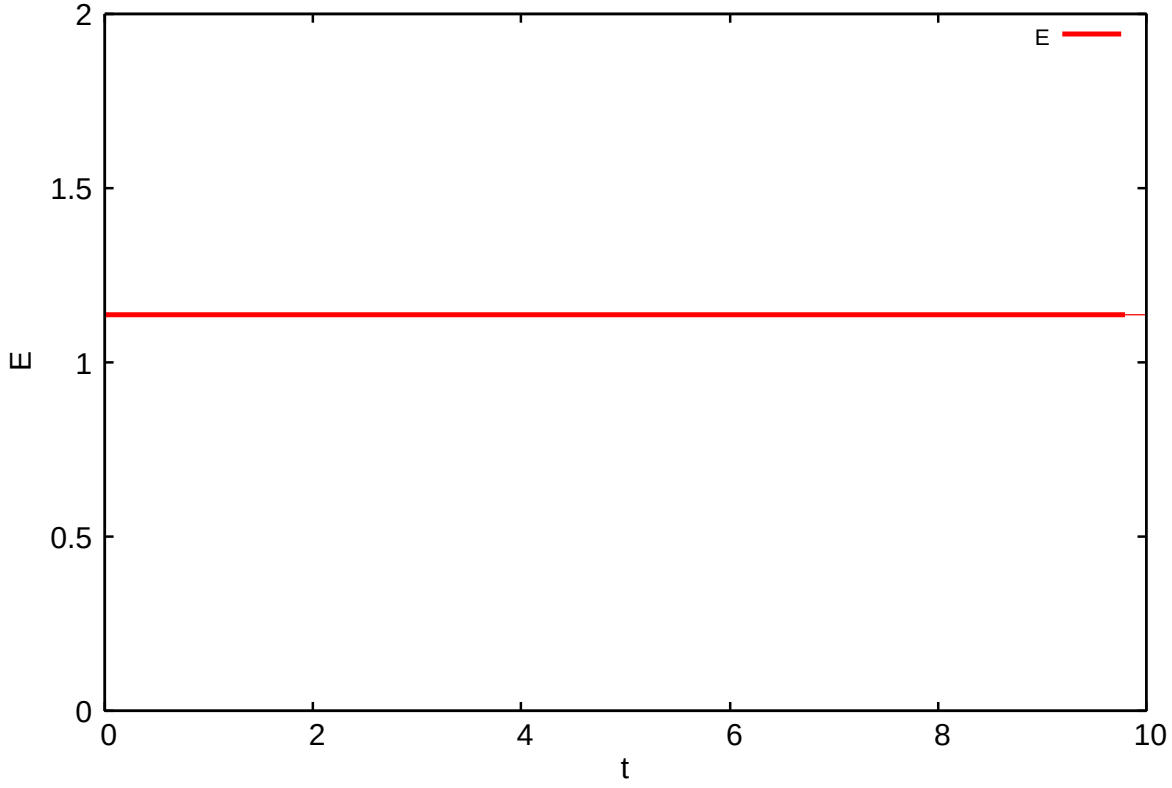

図 4.2: 実時間発展におけるエネルギー期待値の時間変化

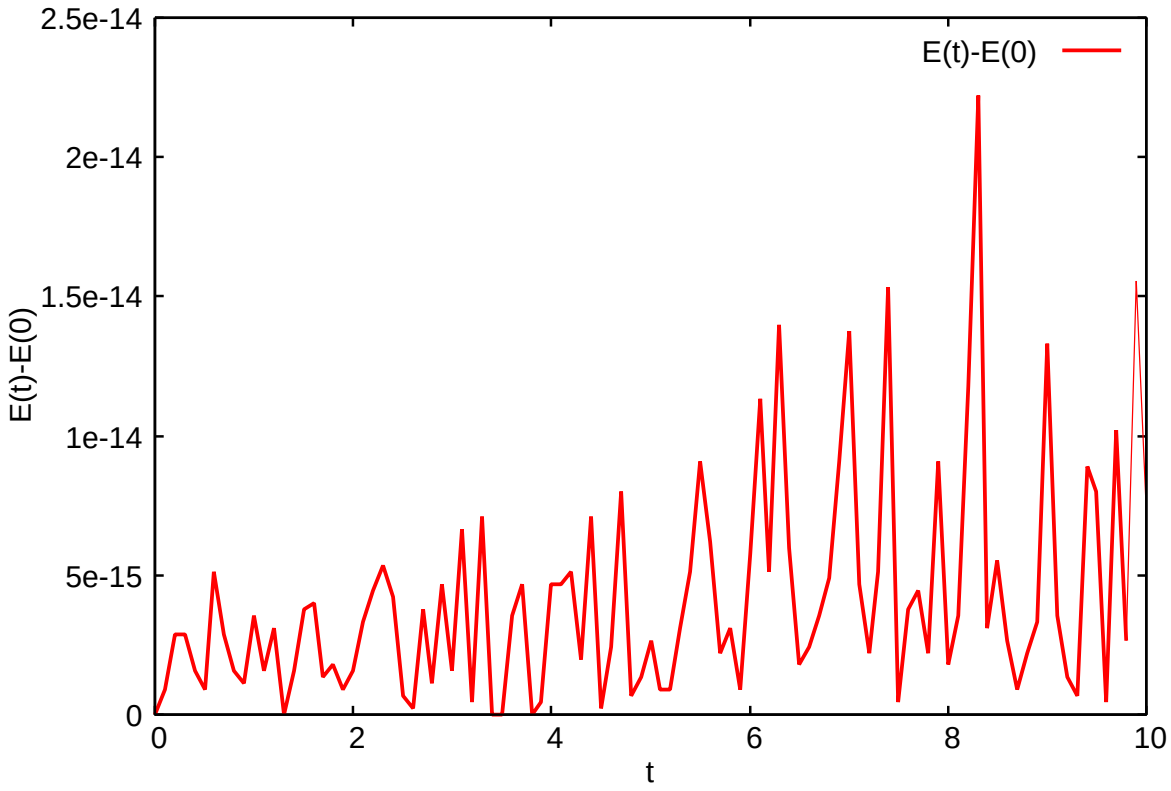

図 4.3: ある時刻 t でのエネルギー期待値の $t = 0$ でのエネルギー期待値からのずれ

4.4 ノルムの保存

実時間発展ではノルムは保存されるはずである．

図 4.4 は，時刻 t での規格化前の状態ベクトルのノルムを，時刻 $t = 0$ での状態ベクトルのノルムで割ったものをプロットしたものである．数値計算の誤差によりノルムは完全に保存されているわけではないが，その誤差は非常に小さく，ノルムは保存されていると考えてよい．

図 4.4 のそれぞれの値の 1 からのずれを図 4.5 に示す．数値計算により規格化された値の 1 からのずれは非常に小さく，充分規格化できていると考えてよい．なお，図 4.4, 4.5 に示した以外の計算では，Runge-Kutta 法を 1 回行うごとに，状態ベクトルの再規格化をして，この微小なノルムのずれを修正している．

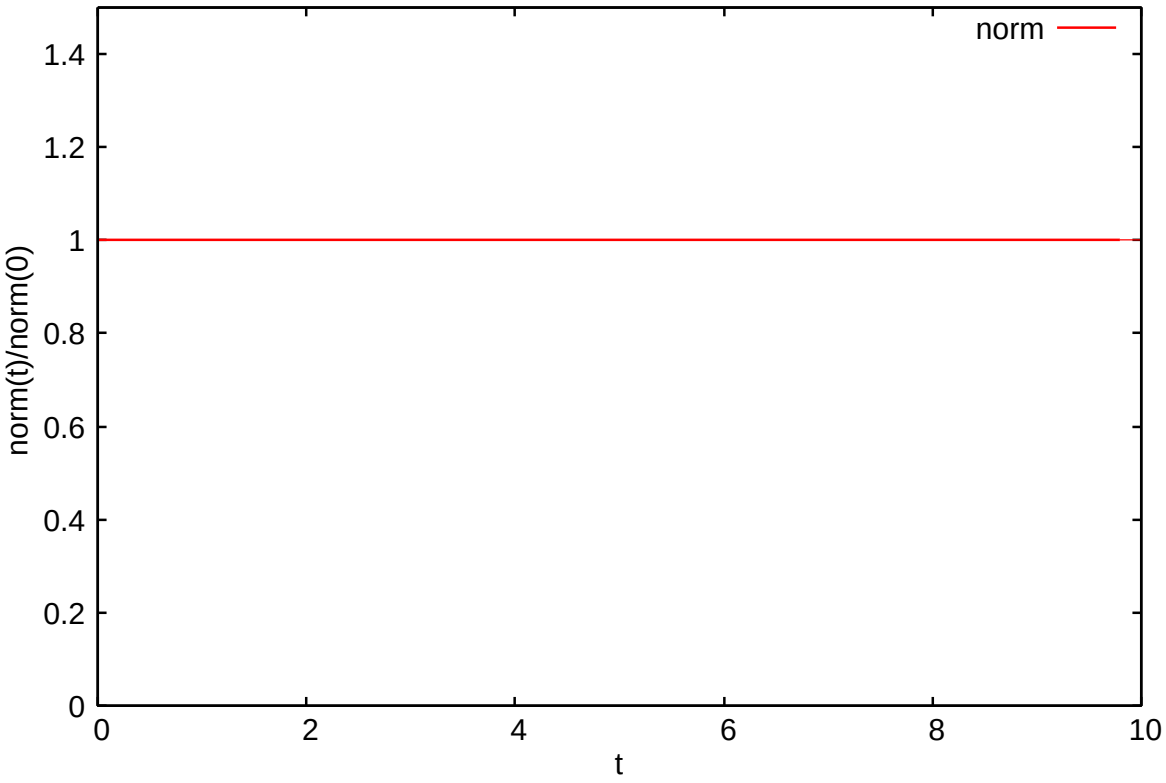

図 4.4: 時刻 t での状態ベクトルのノルムを時刻 $t = 0$ での状態ベクトルのノルムで割ったもの

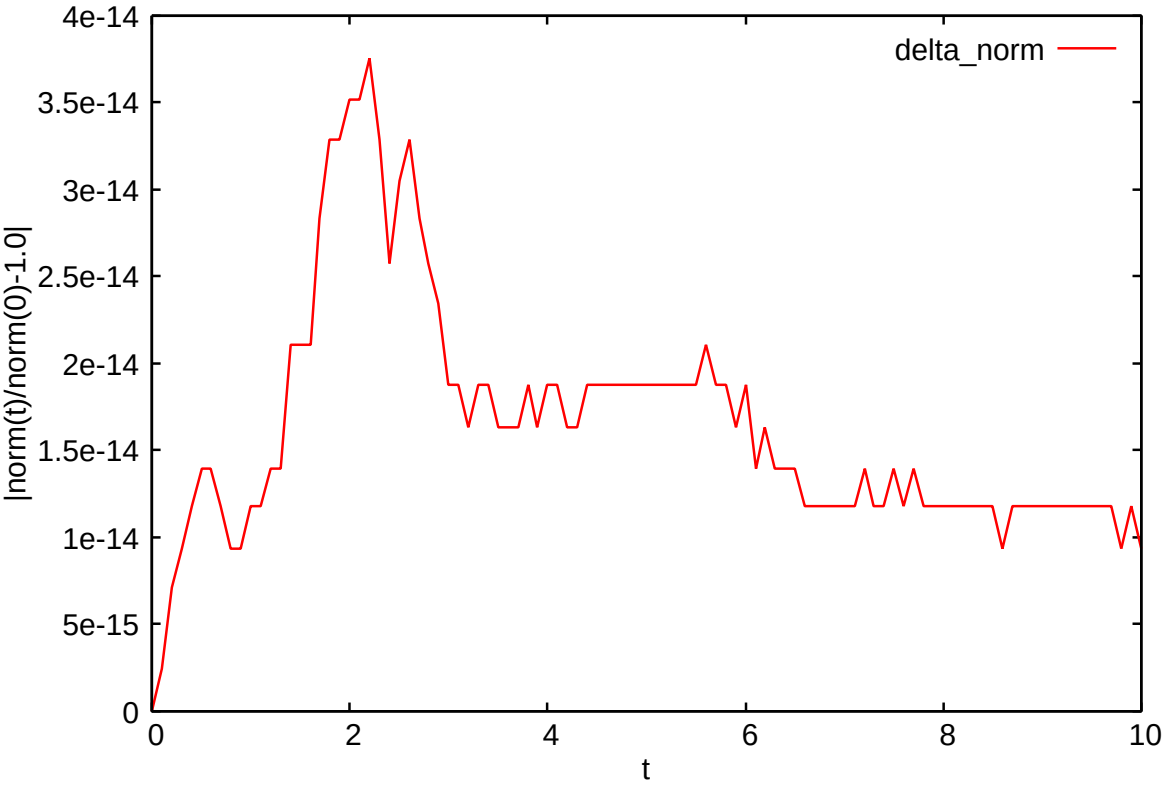

図 4.5: 図 4.4 のそれぞれの値の 1 からのずれ

4.5　冷却（虚時間発展）により得た固有状態の精度

虚時間発展ではエネルギーの期待値は時間とともに減少し，基底状態のエネルギーに収束する．これにより虚時間発展で任意の初期状態を固有状態へと冷却することができる．

図 4.6，4.7 は，ポテンシャルエネルギーを

$$V(x) = \frac{1}{2}x^2 \tag{4.4}$$

として $n_{max} = 1000$ での虚時間発展におけるエネルギー期待値を時間に対してプロットしたものである．図 4.8 は，時刻 t でのエネルギー期待値と $t = 100.0$ でのエネルギー期待値との差を対数プロットしたものである．$t = 40.0$ 付近から完全に一定値（$E = 0.249987\cdots \simeq \frac{\hbar\omega}{2} = 0.25$）となり，$e^{-t}$ に比例して基底状態に収束する．本研究では $t = 10.0$ で充分収束できているとみなす．

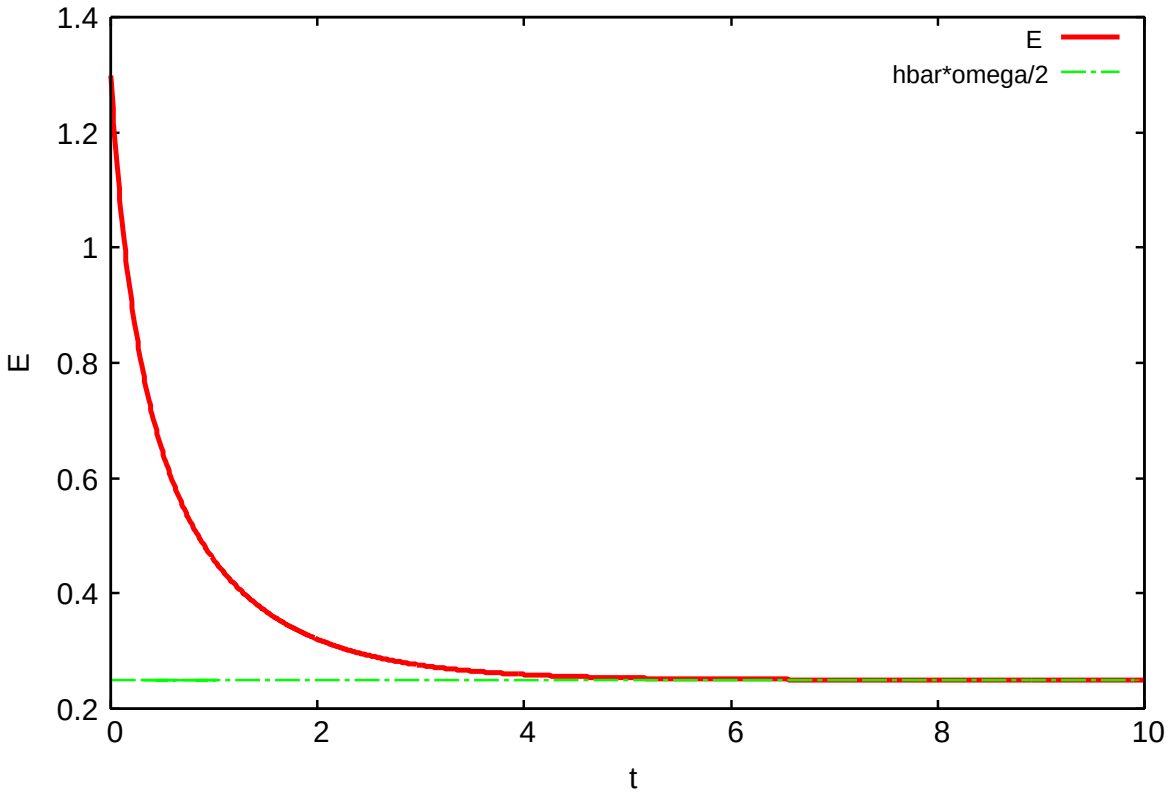

図 4.6: 虚時間発展におけるエネルギー期待値の時間変化 ($t = 0$〜10)

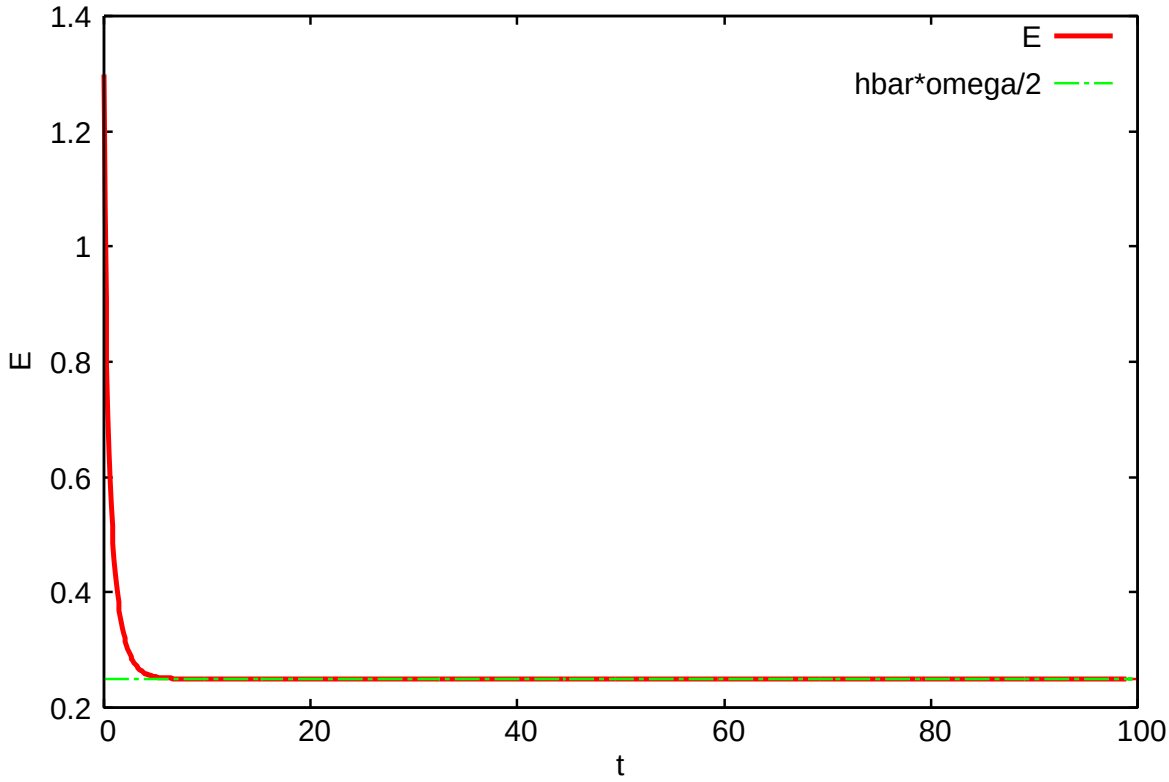

図 4.7: 虚時間発展におけるエネルギー期待値の時間変化 ($t = 0$〜100)

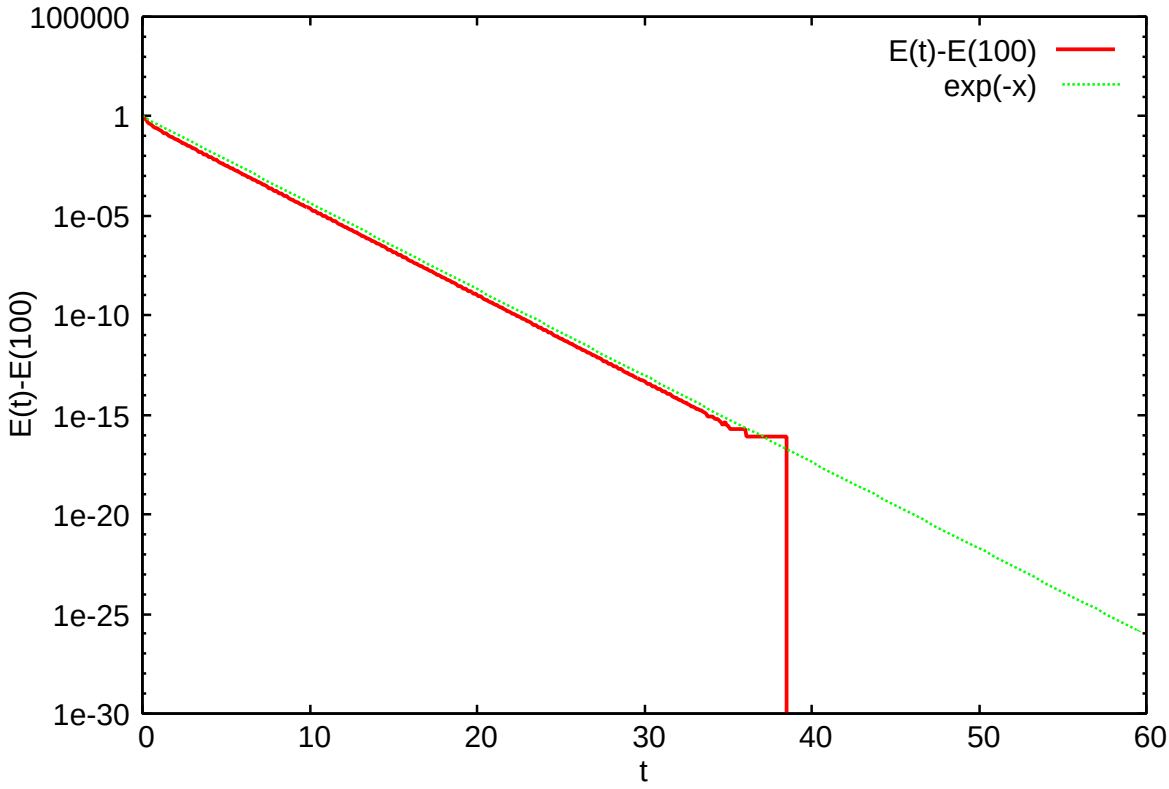

図 4.8: $t = 100$ でのエネルギー期待値からのずれの時間変化

第5章　波束の時間発展

この章では，2つのポテンシャル $V(x)$ を例にとり，波動関数の時間発展を求める．

まず，初期状態を準備しなければならない．本研究では，本来のポテンシャルとは異なるポテンシャル $V'(x)$ 中で状態を虚時間発展させて $V'(x)$ の基底状態へと冷却させることにより初期状態を作る．次に，このようにして準備した初期状態の波束を本来のポテンシャル $V(x)$ 中で実時間発展させ，その振る舞いを調べる．

5.1　非調和振動子

まず，2次のポテンシャル $V'(x)$ 中で虚時間発展させ，$x = 0$ の付近に初期状態の波束を準備する．

$$V'(x) = \left(\frac{x}{\sqrt{8}}\right)^2 \tag{5.1}$$

次に，箱のような6次のポテンシャル $V(x)$ を用意し，準備した初期状態の波束を実時間発展させる．

$$V(x) = \left(\frac{x}{31.6}\right)^6 \tag{5.2}$$

図5.1は，ポテンシャル (5.2)(5.1) およびエネルギーの期待値 E である．図5.2は，時刻 $t = 0, 10, 20, 40, 60, 80, 99.9$ での波動関数の実部と虚部である．

古典力学では，ポテンシャルの底 $(x = 0)$ に静止したままだが，量子力学では波束が拡散してゆくことがわかる．これは位置と運動量の不確定性関係によるものとして理解できる．

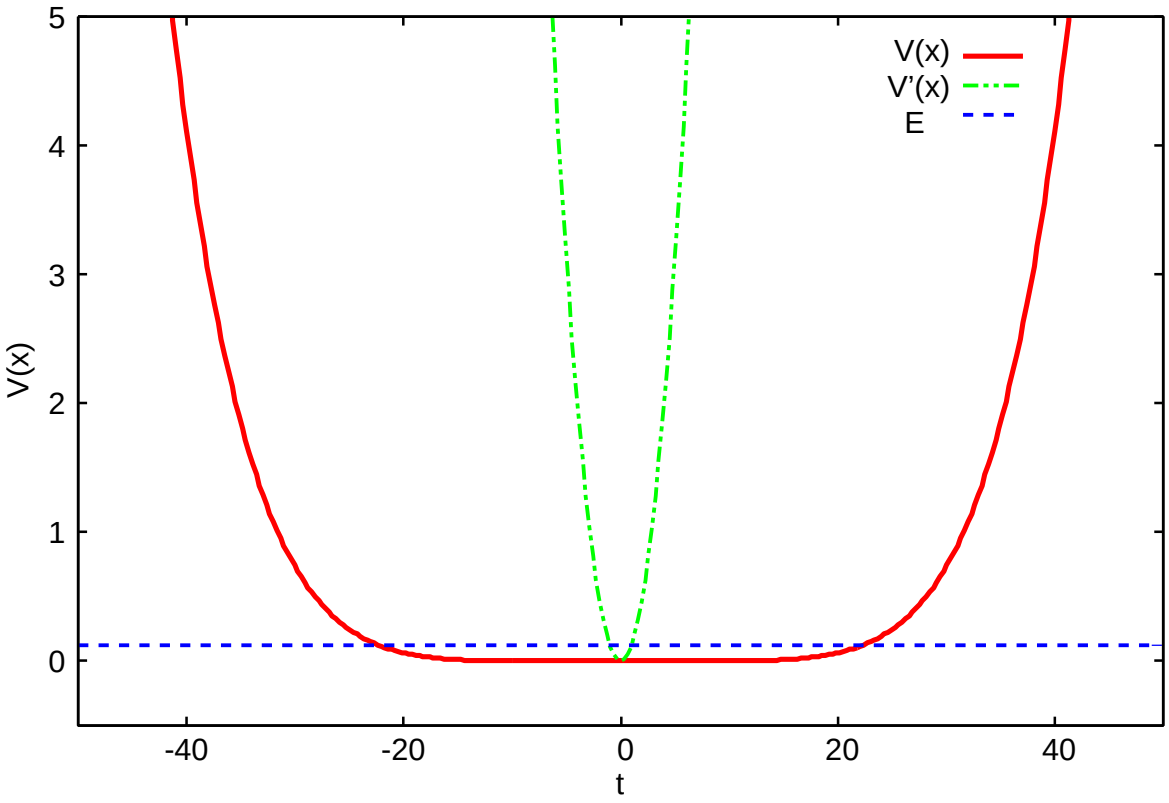

図 5.1: 実時間および虚時間のポテンシャル

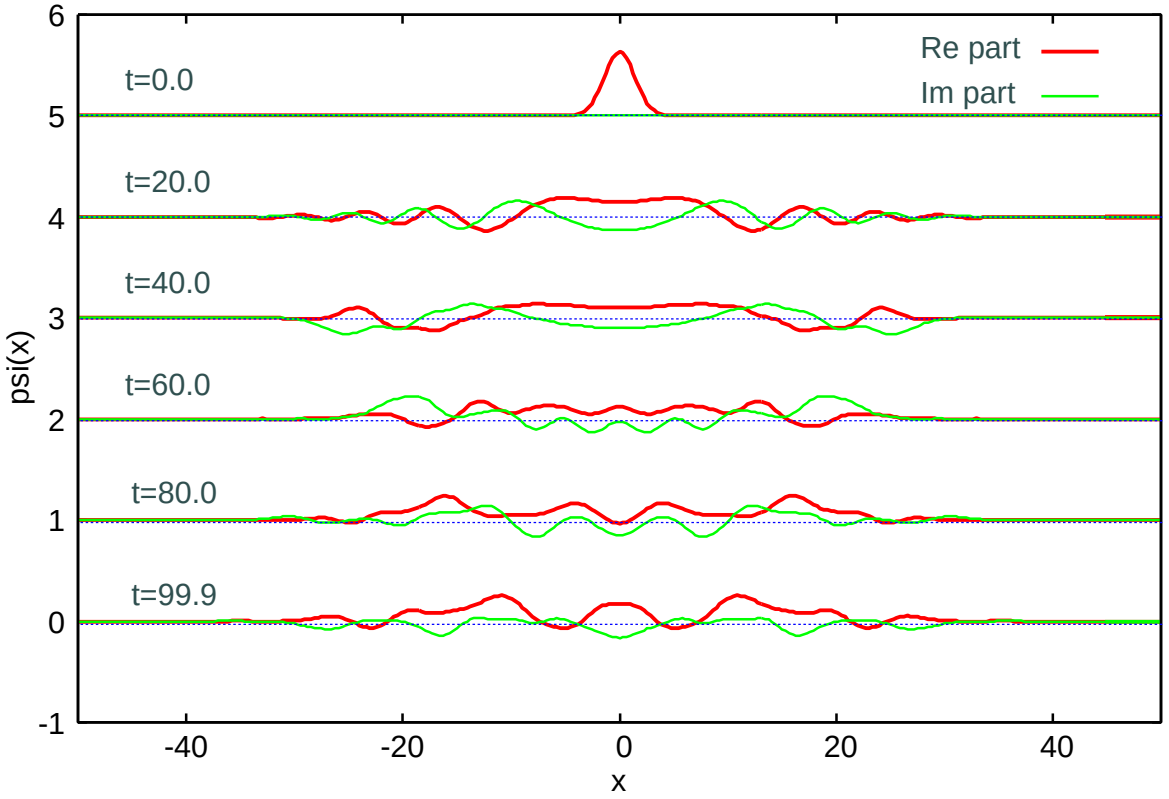

図 5.2: 波束の実時間発展

今度は，初期状態を $x = 0$ からずらしたところにつくり，同様に箱型の 6 次のポテンシャルで実時間発展させる．

実時間でのポテンシャル $V(x)$，初期状態の波束を準備するための虚時間でのポテンシャル $V'(x)$ はそれぞれ以下のとおりである．

$$V(x) = \left(\frac{x}{31.6}\right)^6 \tag{5.3}$$

$$V'(x) = \left(\frac{x-30}{\sqrt{8}}\right)^2 \tag{5.4}$$

図 5.3 は，ポテンシャル (5.3)(5.4) およびエネルギーの期待値 E である．図 5.4 は，時刻 $t = 0, 10, 20, 40, 60, 80, 99.9$ での波動関数の実部と虚部である．

ポテンシャルエネルギーの高い地点におかれた波束は，拡散するだけでなく，古典的粒子と同様にポテンシャルの斜面をすべり落ち，ポテンシャルの平らな底をよこぎり，左側のポテンシャル壁に衝突してはね返されることが見てとれる．これは古典力学が量子力学の近似として実現される例と考えることができる．

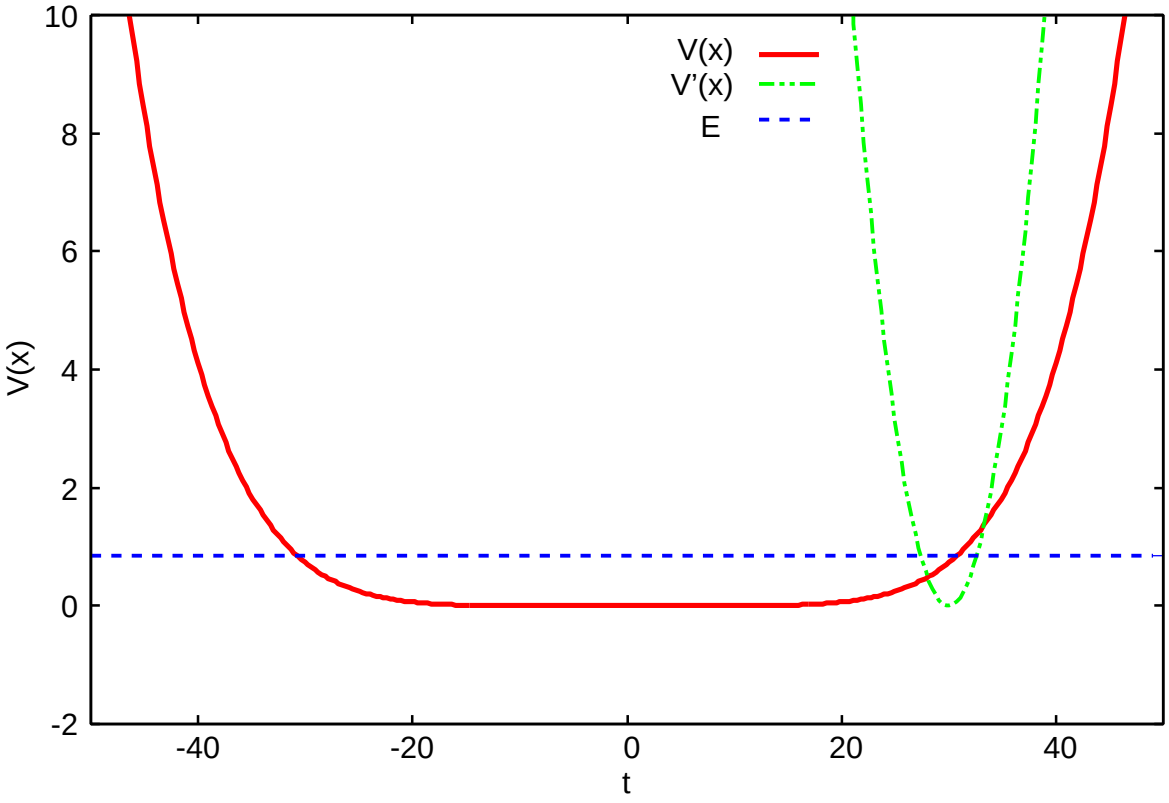

図 5.3: 実時間および虚時間のポテンシャル

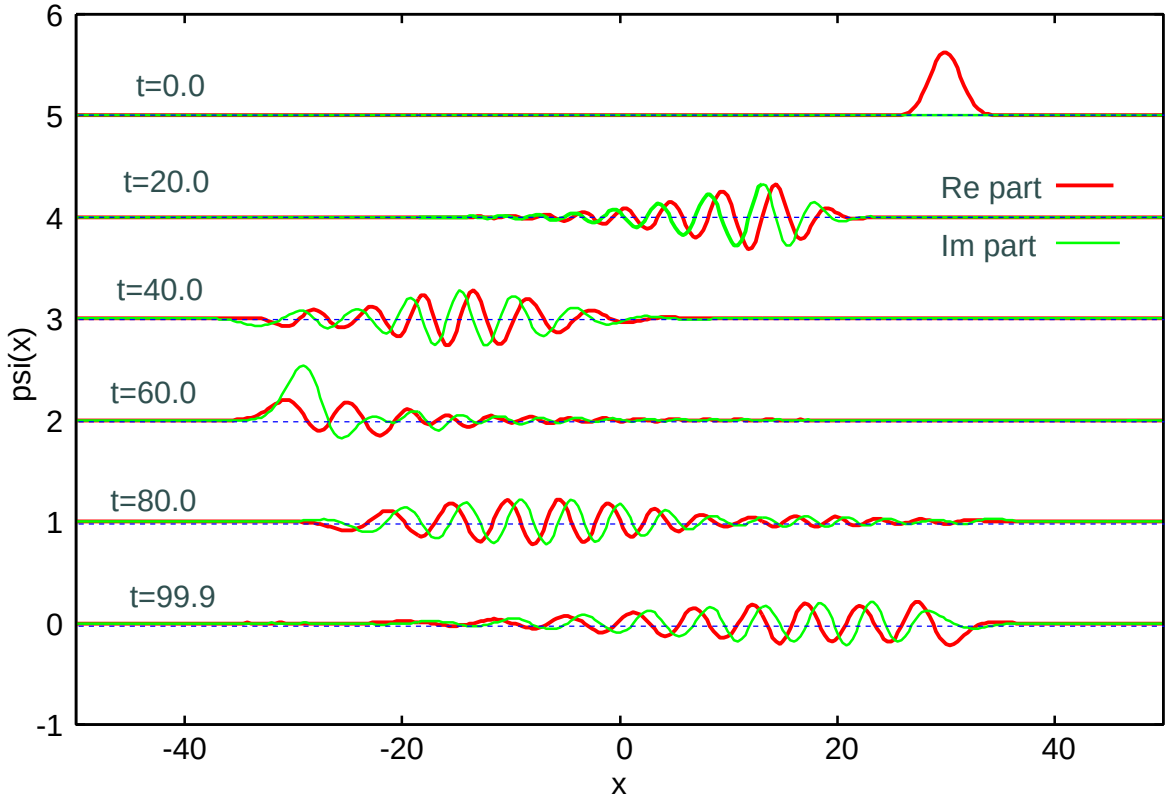

図 5.4: 波束の実時間発展

5.2 トンネル効果

次に，実時間において二重の井戸型のポテンシャルを考える．初期状態をつくるための虚時間のポテンシャル $V'(x)$ は，$x > 0$ の井戸に局在した初期状態となるように，$x < 0$ のポテンシャルを埋め立てるような，実時間のポテンシャルに 6 次の項を加えた形にしてある．

実時間でのポテンシャル $V(x)$，初期状態の波束を準備するための虚時間でのポテンシャル $V'(x)$ はそれぞれ以下のとおりである．

$$V(x) = -\frac{1}{10}x^2 + \frac{1}{200}x^4 \tag{5.5}$$

$$V'(x) = V(x) + \left(\frac{x-10}{24.4}\right)^6 \tag{5.6}$$

図 5.5 は，ポテンシャル (5.5)(5.6) およびエネルギーの期待値 E である．図 5.6 は，時刻 $t = 0, 10, 20, 40, 60, 80, 99.9$ での波動関数の実部と虚部である．

エネルギーの期待値がポテンシャル障壁の最高点より低いので，古典的には粒子は左側のポテンシャル井戸へしみ出すことはできないはずだが，量子力学ではトンネリングによりしみ出してゆくことがわかる．

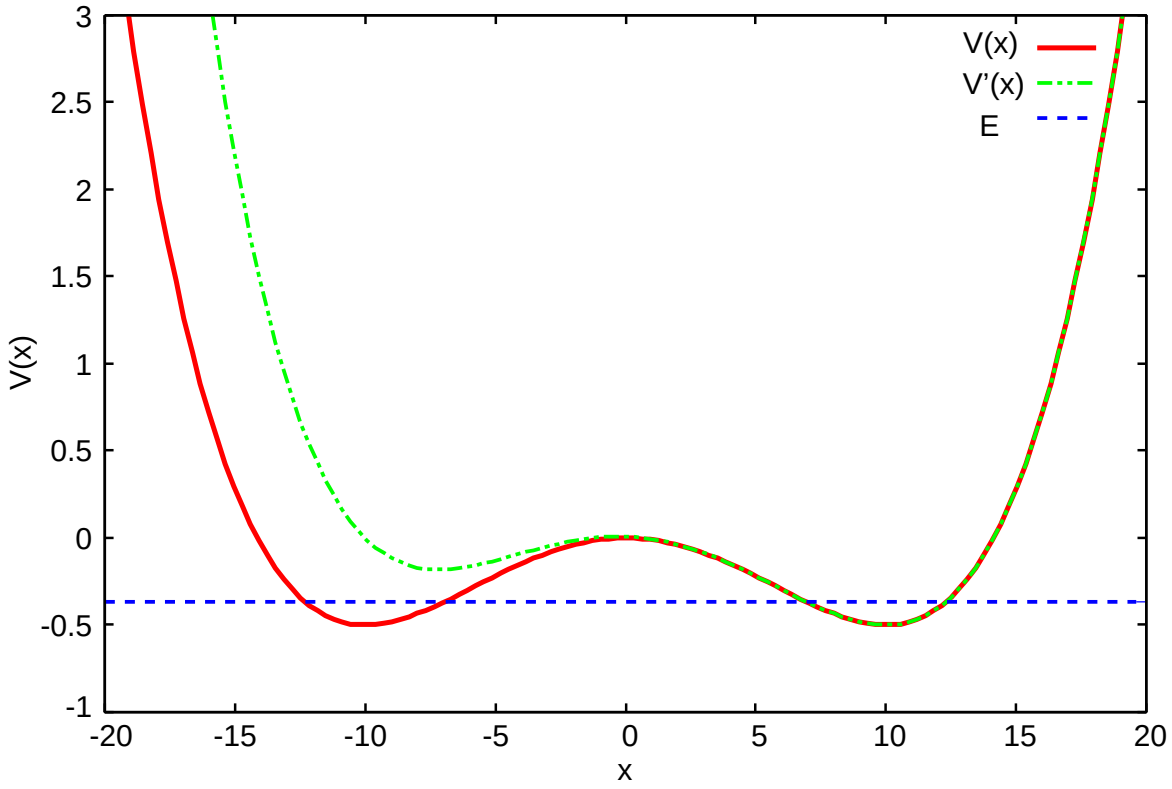

図 5.5: 実時間および虚時間のポテンシャル

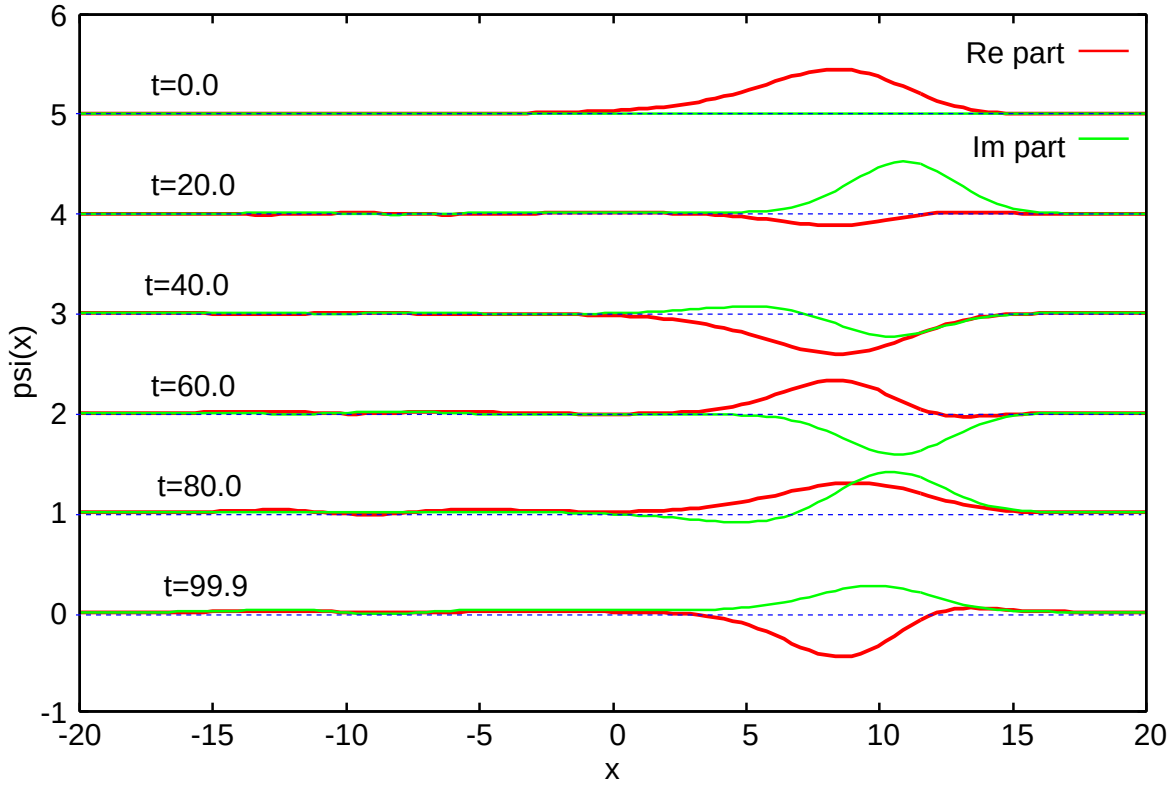

図 5.6: 波束の時間発展

今度は，二重井戸の障壁を高くし，同様に $x > 0$ の井戸に初期状態をつくり実時間発展させる．実時間でのポテンシャル $V(x)$，初期状態の波束を準備するための虚時間でのポテンシャル $V'(x)$ はそれぞれ以下のとおりである．

$$V(x) = -\frac{2}{5}x^2 + \frac{1}{50}x^4 \tag{5.7}$$

$$V'(x) = V(x) + \left(\frac{x-10}{17.8}\right)^6 \tag{5.8}$$

図 5.7 は，ポテンシャル (5.7)(5.8) およびエネルギーの期待値 E である．図 5.8 は，時刻 $t = 0, 10, 20, 40, 60, 80, 99.9$ での波動関数の実部と虚部である．

今度は，トンネリングが非常に遅くなり，プロットした時間範囲では，左側のポテンシャル井戸へのしみ出しは見られない．また，右側の井戸中の波束の運動は，全体が一斉に位相 $e^{-i\omega t}$ で変化する定常状態によく似ている．このような右側の井戸中の波束は準定常状態と呼ぶ．

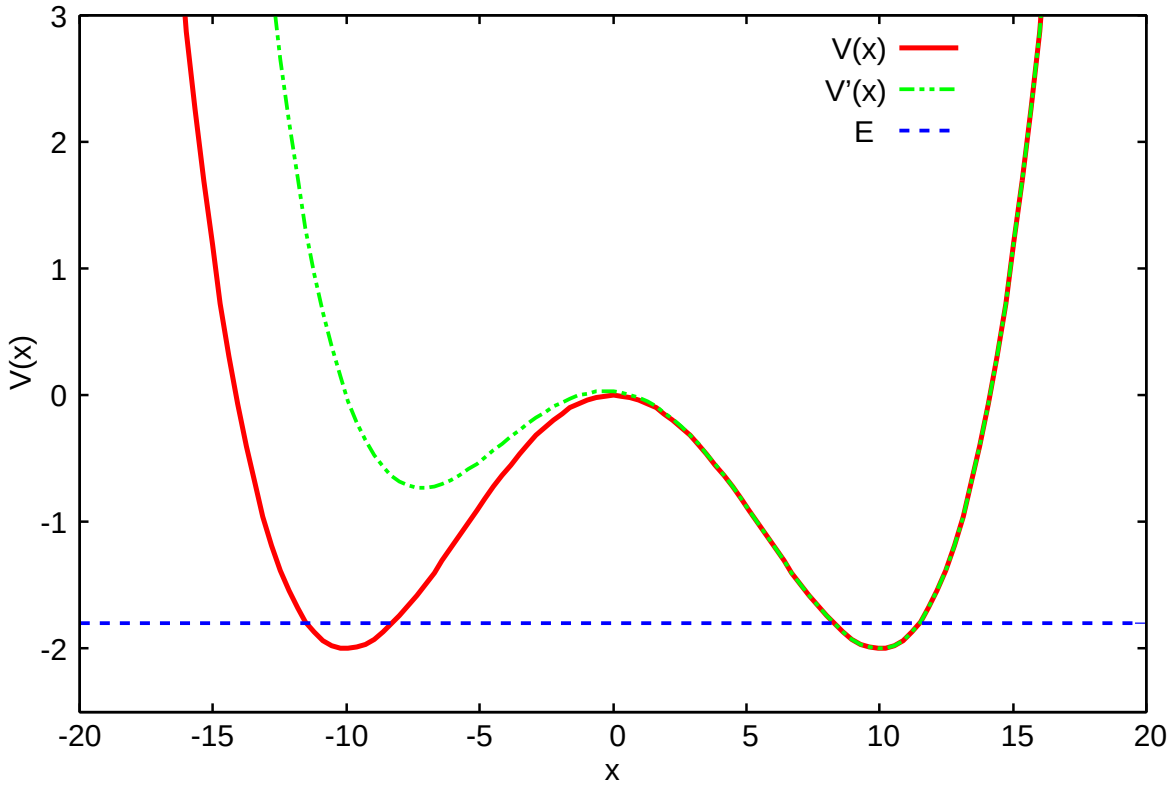

図 5.7: 実時間および虚時間のポテンシャル

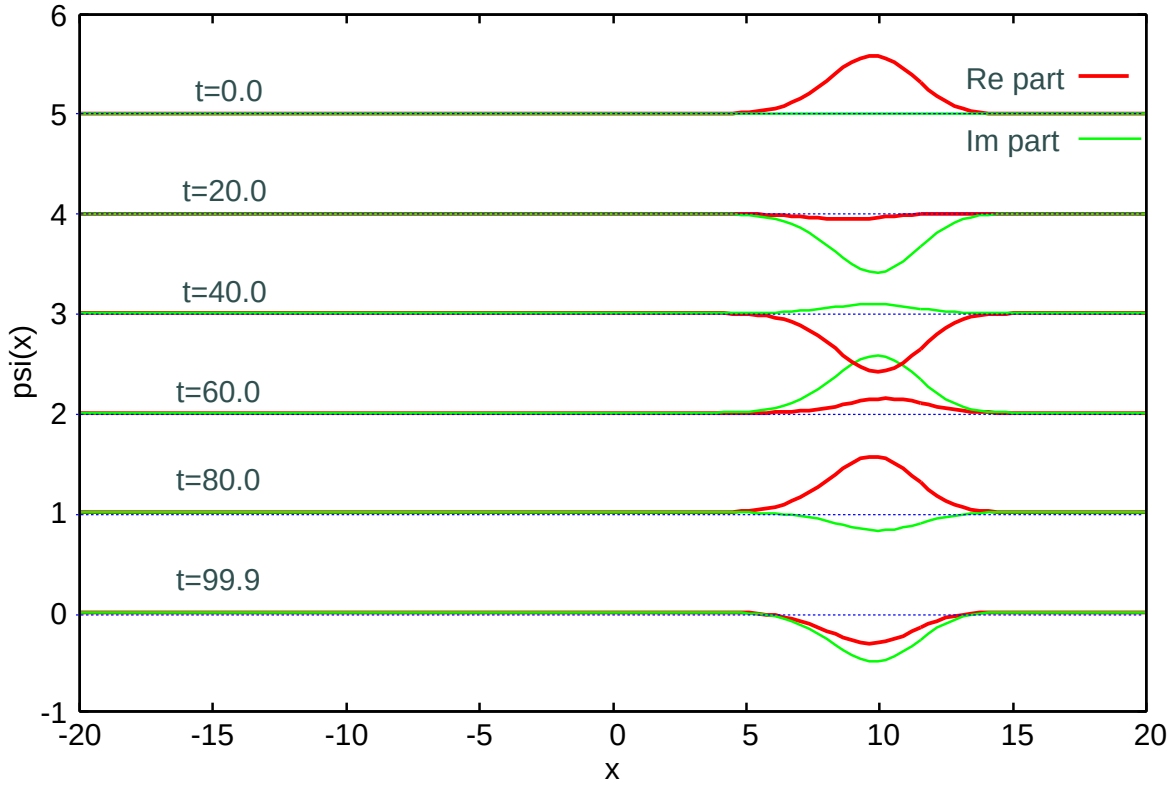

図 5.8: 波束の実時間発展

第6章　結論

時間に依存する Schrödinger 方程式の調和振動子基底展開による表現を求め，それを数値的に解く C 言語プログラムを作製した (プログラムリストを付録に収録した)．また，本研究を進めるにあたって，量子力学の基礎を系統的に学び直した．

応用として，波束の実時間発展の量子力学的な側面と，古典力学と類似性のある側面を示した．これは 4 本の動画として公開しており，参考文献にその URL を記載してある．

まず，箱型のポテンシャルでの実時間発展について，$x = 0$ 付近に初期状態の波束を準備する例では，実時間発展により波束が拡散していく様子を確認した．これは明らかに量子力学的な振る舞いを示している．

$x = 0$ からずらしたところ（$x = 30$）に初期状態の波束を準備する例では，実時間発展により波束が拡散しながらも，ポテンシャルにそって振動を繰り返した．量子力学的な振る舞いと古典力学と類似性のある振る舞いが混在した様子を確認した．

実時間発展によるトンネル効果について，二重井戸のポテンシャルの障壁高が低い場合には，波動関数のすそが $x < 0$ までのびているため，実時間発展ですぐに $x < 0$ の井戸の成分が混ざった．ここでは量子力学特有の現象であるトンネル効果の様子が確認された．

障壁高をある程度高くすると，波動関数のすそが $x < 0$ までのびることができず，トンネル効果は見られない．波束は $x > 0$ の井戸に局在し，実時間発展で固有状態のように全体が同位相 $e^{-i\omega t}$ で振動しているように見える．ただし，$x < 0$ の井戸へのトンネリングが起こりにくいため，固有状態ではなく準定常状態といえる．

本研究では，巨大な次元の調和振動子基底で一次元量子系を数値的に扱うことができた．今後は，量子複雑系の例として，高次元空間の振動運動に拡張することが発展課題として考えられる．

参考文献

[1] 清水明：量子論の基礎 - その本質のやさしい理解のために (サイエンス社, 2003)

[2] 小出昭一郎：量子力学 (I) (裳華房, 2010)

[3] Leonard I. Schiff：新版 量子力学 上 (吉岡書店, 2008)

[4] 砂川重信：量子力学の考え方 (岩波書店, 1993)

[5] 奥村晴彦：C 言語による最新アルゴリズム事典 (技術評論社, 1992)

[6] 筑波大学 課題探求実習レポート 拡散モンテカルロ法による軽い原子基底状態の数値計算
http://www.tac.tsukuba.ac.jp/ ozawa/Sato.pdf

[7] 東京学芸大学 資料「昇降演算子法による調和振動子」 新田英雄
http://www.u-gakugei.ac.jp/ nitta/operator.pdf

[8] 波束の時間発展
動画１ http://apphy.u-fukui.ac.jp/~nucleus/daiThesis12_movie1.m1v
動画２ http://apphy.u-fukui.ac.jp/~nucleus/daiThesis12_movie2.m1v
動画３ http://apphy.u-fukui.ac.jp/~nucleus/daiThesis12_movie3.m1v
動画４ http://apphy.u-fukui.ac.jp/~nucleus/daiThesis12_movie4.m1v

謝辞

本論文を作成するにあたり、田嶋直樹先生には終始丁寧なご指導をしていただいたことに感謝し、お礼申し上げます。また林明久先生にも本研究及び日常的なことにおいても、実に丁寧な指導、お世話をしていただきました。

本研究に対してご意見をいただいた、多くの物理工学科の先生方をはじめ、諸先輩方や同卒研生にもお礼申し上げ、謝辞の言葉とさせていただきます。

付録　プログラムソースコード

以下のプログラムでは概ね以下のような処理が行われる.

- 状態ベクトルの初期値を設定
- Runge-Kutta 法で虚時間発展（冷却）させ，何らかのハミルトニアンの基底状態として，初期状態の波束を準備
- Runge-Kutta 法で，準備した初期状態の波束を実時間発展させる
- エネルギー期待値 E，エネルギーのゆらぎ ΔE，x の期待値，x のゆらぎ Δx を出力
- 状態ベクトルおよび波動関数を出力

```
# include <stdio.h>
# include <stdlib.h>
# include <complex.h>
# include <math.h>

# define nmax 1000     // 量子数 n の最大値
# define N (nmax+1)   // 基底に含める（量子数 n=0..nmax をもつ）基底の個数
# define maxpower 6   // x のべきの最大値

const double hbar = 1.0;
const int ki = 2*maxpower;
double ol_b; // b (oscillator length of the basis)

int runge(double tmax,int itmax,int dit_obs, int dit_vec,
                            int dit_wav,int cooling,double complex *v);
int set_hamiltonian_matrix(int option);
double deriv_xi_2 (int i, int j);
double xi_0 (int i, int j);
double xi_1 (int i, int j);
double xi_2 (int i, int j);
double xi_3 (int i, int j);
double xi_4 (int i, int j);
double xi_5 (int i, int j);
double xi_6 (int i, int j);
double complex timederivative( int i, double t, double complex *x);
double norm_square(double complex *x);
int normalize(double complex *x);
double harmonic_oscillator_wavefunction(int n, double x);
int write_vector(double time, double complex *v);
```

```
int write_wavefunction(double time, double complex *v);
int write_potential(double q0, double q1, double q2, double q3,
                                        double q4, double q5, double q6);
double deriv_xi_2 (int i, int j);
double xi_2 (int i, int j);
double energy_expectation_value( double complex *x);
double energy_2_expectation_value( double complex *x);
double x_expectation_value( double complex *x);
double x_2_expectation_value( double complex *x);

int main(){
   // main1();
   main2();
   // main3();
   return 0;
}

int main1(){
   double complex v[N]; // 状態ベクトル
   double tmax; // 時間発展を行う時間の長さ
   int itmax = 10000; // 時間の刻みの総数（0 番..itmax 番の刻みがある）
   int dit_obs = 10; // observables を出力する時間ステップ番号の間隔
   int dit_vec = 10; // 状態ベクトルを出力する時間ステップ番号の間隔
   int dit_wav = 100; // 波動関数をファイル出力する時間ステップ番号の間隔
    // 注：dit_vec,dit_wav ともにゼロ以下の値を設定すると出力が完全に抑止される。
   double omega = 0.5; // 注意：同一用途の同名変数の値を 2 箇所で定義している。
   double beta = 0.1; // 初期状態設定用パラメータ
                      // （温度の逆数、単位＝ 1/基底準位間隔）
   int cooling = 0; // =0 で実時間発展、=1 で虚時間発展
   int i; // 作業用変数 (カウンター)

   set_hamiltonian_matrix(1);

   tmax = (4.0*M_PI/omega)*1;

   /* 初期状態の設定 */

   for ( i=0; i<N; i++){
     // v[i] = 1.0;
      v[i] = exp(-beta*i);
   }
   normalize(v);   // 状態の規格化

   /* 虚時間発展 : 何らかのハミルトニアンの基底状態として初期状態を作る */

   // runge(tmax, itmax, 0, 0, 0, 1, v);
   // runge(tmax, itmax, dit_obs, dit_vec, dit_wav, 1, v);

   /* 実時間発展 */

   runge(tmax, itmax, dit_obs, dit_vec, dit_wav, cooling, v);

   return 0;
}

int main2(){
   double complex v[N]; // 状態ベクトル
```

```
   double tmax; // 時間発展を行う時間の長さ
   int itmax;  // 時間の刻みの総数（0 番..itmax 番の刻みがある）
   int dit_obs=0; // observables を出力する時間ステップ番号の間隔
   int dit_vec=0; // 状態ベクトルを出力する時間ステップ番号の間隔
   int dit_wav=0; // 波動関数をファイル出力する時間ステップ番号の間隔
    // 注：dit_vec,dit_wav ともにゼロ以下の値を設定すると出力が完全に抑止される。
   double omega = 0.5; // 注意：同一用途の同名変数の値を 2 箇所で定義している。
   double beta = 0.1; // 初期状態設定用パラメータ
                      // （温度の逆数、単位＝ 1/基底準位間隔）
   int cooling = 1; // =0 で実時間発展、=1 で虚時間発展
   int i; // 作業用変数 (カウンター)

   /* 初期状態の設定 */

   for ( i=0; i<N; i++){
     // v[i] = 1.0;
      v[i] = exp(-beta*i);
   }
   normalize(v);   // 状態の規格化

   /* 虚時間発展 : 何らかのハミルトニアンの基底状態として初期状態を作る */

   set_hamiltonian_matrix(1);
   tmax = 10.0;
   itmax=10000000;
   dit_obs = 0;
   dit_vec = 0;
   dit_wav = 0;
   cooling = 1;
   runge(tmax, itmax, dit_obs, dit_vec, dit_wav, cooling, v);

   /* 実時間発展 */

   set_hamiltonian_matrix(2) ;
   tmax = 100.0;
   itmax= 100000000;
   dit_obs = 100000;
   dit_vec = 100000;
   dit_wav = 100000;
   cooling = 0;
   runge(tmax, itmax, dit_obs, dit_vec, dit_wav, cooling, v);

   return 0;
}

/* ルンゲ・クッタ法による時間発展 */

int runge(double tmax, int itmax, int dit_obs, int dit_vec,
                        int dit_wav, int cooling, double complex *v){
   int nprint=6; // 出力する状態ベクトルの成分の個数（n=0..nprint-1 を出力）

   double complex vtmp[N], k0[N], k1[N], k2[N], k3[N];
   double t,dt; // 時刻, 時刻の刻み幅
   double complex tfct; // 実時間発展では dt, 虚時間発展では-I*dt
   double norm2; // 状態ベクトルのノルムの 2 乗
   double fct; // 作業変数（因子）
   double E; // エネルギー期待値 E
```

```
double E_2; // E の 2 乗の期待値
double X; // x の期待値
double X_2; // x の 2 乗の期待値
double delta_E; // Δ E
double delta_x; // Δ x
int itime; // 時刻の刻みの番号=0 番..itmax 番
int i; // 作業変数（カウンター）

dt = tmax/itmax;
if(nprint>N)nprint=N;

/* t=0 */
t = 0.0;
if(dit_obs > 0){  // t=0 での observables の出力
  E = energy_expectation_value(v);
  E_2 = energy_2_expectation_value(v);
  X = x_expectation_value(v);
  X_2 = x_2_expectation_value(v);
  delta_E = sqrt(fabs(E_2 - E*E));
  delta_x = sqrt(fabs(X_2 - X*X));
  printf("# t, E, delta_E, X, delta_x\n");
  printf("%f %.16f %.16f %.16f %.16f :obs\n", t, E, delta_E, X, delta_x);
}

if(dit_vec > 0) write_vector(t,v); // t=0 での vector の出力

if(dit_wav > 0) write_wavefunction(t,v);

/* t>0 */
if(cooling==0){ tfct=dt; } else { tfct=-dt*I; }
for ( itime=1; itime<=itmax; itime++){

   t = dt*itime;

   for ( i=0; i<N; i++) k0[i] = timederivative( i, t, v)*tfct;

   for ( i=0; i<N; i++) vtmp[i]=v[i] + k0[i]/2.0;
   for ( i=0; i<N; i++) k1[i] = timederivative( i, t + dt/2.0, vtmp)*tfct;

   for ( i=0; i<N; i++) vtmp[i]=v[i] + k1[i]/2.0;
   for ( i=0; i<N; i++) k2[i] = timederivative( i, t + dt/2.0, vtmp)*tfct;

   for ( i=0; i<N; i++) vtmp[i]=v[i] + k2[i];
   for ( i=0; i<N; i++) k3[i] = timederivative( i, t + dt, vtmp)*tfct;

   for ( i=0; i<N; i++) v[i] += ( k0[i] + 2.0*k1[i] + 2.0*k2[i] + k3[i])/6.0;


   if(dit_obs > 0 && itime % dit_obs == 0){
     E = energy_expectation_value(v);
     E_2 = energy_2_expectation_value(v);
     X = x_expectation_value(v);
     X_2 = x_2_expectation_value(v);
     delta_E = sqrt(fabs(E_2 - E*E));
     delta_x = sqrt(fabs(X_2 - X*X));

     printf("%f %.16f %.16f %.16f %.16f :obs\n", t, E, delta_E, X, delta_x);
```

```
      }

      normalize(v); // 数値計算誤差によるノルムの１からのずれを修正する

      if(dit_vec > 0 && itime % dit_vec == 0) write_vector(t,v);

      if(dit_wav > 0 && itime % dit_wav == 0 ) write_wavefunction(t,v);

   }
   if(dit_wav < 0 ) write_wavefunction(t,v);
   return 0;
}

/* hamiltonian の計算 */

// double ham1[N][N];
double *ham1 = NULL;

int set_hamiltonian_matrix(int option){

   double mass_b = 1.0;
   double omega_b = 1.0;
   double mass = 1.0;
   double omega = 0.5;
   double x0 = 30.0;
   double x1 = 30.0;
   double hight1 = 1000.0;
   double hight2 = 2.0;
   double p, q1, q2, q3, q4, q5, q6, q0;
   int i, j;
   double x0_pot=10.0; // location of minimum
   double h_pot=0.5; // barrier height
   double d_pot=0.0; // 0.25; // splitting of the depth of two minima

   if(ham1 == NULL){
     ham1=malloc(N*(2*maxpower + 1)*sizeof(double));
     if(ham1 == NULL) {fprintf(stderr,"malloc:%d\n",N);exit(1);}
   }

   ol_b = sqrt(hbar/(mass_b*omega_b));

//  虚時間発展のハミルトニアンの各項の係数 (cooling により波束をつくる)
   if( option == 1 ){
      p = -hbar*hbar/(2.0*mass_b*ol_b*ol_b); // ２階微分の係数
      q1 = 0.0; // １次の項の係数
      q2 = mass*omega*omega*ol_b*ol_b/2.0; // ２次の項の係数
      q3 = 0.0; // ３次の項の係数
      q4 = 0.0; // ４次の項の係数
      q5 = 0.0; // ５次の項の係数
      q6 = 0.0; // mass*omega_b*omega_b*ol_b*ol_b*ol_b*ol_b*ol_b*ol_b*ol_b/2.0; // 6
次の項の係数
      q0 = 0.0; // mass*omega_b*omega_b*x0*x0*x0*x0*x0*x0*x0/2.0; // 定数項
   }
   else if( abs(option) == 2 ){
//  実時間発展のハミルトニアン (準備した波束を振動させる)
     double bxpow1=ol_b/x0_pot,bxpow=bxpow1;
      p = -hbar*hbar/(2.0*mass*ol_b*ol_b);
```

```
      q0 = 0.0;
      q1 = 0.0;
      q2 = 0.0;
      q3 = 0.0;
      q4 = 0.0;
      q5 = 0.0;
      q6 = 1.0/995686217.814; // (1/31.6)^6;
      if(option < 0){
//  虚数時間発展のハミルトニアン(波束を準備する)
        double vspp=h_pot/pow(2*x0_pot,6.0);
        q0 += x0_pot*x0_pot*x0_pot*x0_pot*x0_pot*x0_pot*vspp;
        q1 -= 6.0*x0_pot*x0_pot*x0_pot*x0_pot*x0_pot*vspp;
        q2 += 15.0*x0_pot*x0_pot*x0_pot*x0_pot*vspp;
        q3 -= 20.0*x0_pot*x0_pot*x0_pot*vspp;
        q4 += 15.0*x0_pot*x0_pot*vspp;
        q5 = -6.0*x0_pot*vspp;
        q6 = vspp;
      }
   }

   for( i=0; i<N; i++){

      int js = i-maxpower;  // jstart
      if (js < 0) js = 0;

      int je = i+maxpower;  // jend
      if (je > nmax) je = nmax;

      for ( j=js; j<=je; j++){
         int k = ki*i + j;
         ham1[k] = p*deriv_xi_2(i, j)+q0*xi_0(i, j)+q1*xi_1 (i, j)+q2*xi_2(i, j)
                     +q3*xi_3(i, j)+q4*xi_4(i, j)+q5*xi_5(i, j)+q6*xi_6(i, j);
      }
   }

   printf("# %f =mass\n",mass);
   printf("# %f =ol_b\n",ol_b);
   write_potential(q0, q1, q2, q3, q4, q5, q6);
   return 0;
}


// <i|(d^2/dxi^2)|j> 部分の計算
double deriv_xi_2 (int i, int j){

   if( i==(j-2) ){
      return sqrt(j*j-j)*0.5;
   }
   else if( i==j ){
      return -j-0.5;
   }
   else if( i==(j+2)){
      return sqrt((j+1.0)*(j+2.0))*0.5;
   }
   else {
      return 0;
   }
```

```
}

// <i|j> 部分の計算
double xi_0 (int i, int j){

   if( i==j ){
      return 1;
   }
   else {
      return 0;
   }
}

// <i|(xi^1)|j> 部分の計算
double xi_1 (int i, int j){

   if( i==(j-1) ){
      return sqrt(j*0.5);
   }
   else if( i==(j+1)){
      return sqrt((j+1.0)*0.5);
   }
   else {
      return 0;
   }
}

// <i|(xi^2)|j> 部分の計算
double xi_2 (int i, int j){

   if( i==(j-2) ){
      return sqrt(j*j-j)*0.5;
   }
   else if( i==j ){
      return j+0.5;
   }
   else if( i==(j+2)){
      return sqrt((j+1.0)*(j+2.0))*0.5;
   }
   else {
      return 0;
   }
}

// <i|(xi^3)|j> 部分の計算
double xi_3 (int i, int j){

   if( i==(j-3) ){
      return sqrt(j*(j-1.0)*(j-2.0)*0.5)*0.5;
   }
   else if( i==(j-1) ){
      return 3.0*j*0.5*sqrt(j*0.5);
   }
   else if( i==(j+1) ){
      return 3.0*(j+1.0)*0.5*sqrt((j+1.0)*0.5);
   }
   else if( i==(j+3) ){
```

```
      return 0.5*sqrt((j+1.0)*(j+2.0)*(j+3.0)*0.5);
   }
   else {
      return 0;
   }
}

// <i|(xi^4)|j> 部分の計算
double xi_4 (int i, int j){

   if( i==(j-4) ){
      return sqrt(j*(j-1.0)*(j-2.0)*(j-3.0))*0.25;
   }
   else if( i==(j-2) ){
      return (2.0*j-1.0)*0.5*sqrt(j*(j-1.0));
   }
   else if( i==j ){
      return 3.0*0.25*(j*j+(j+1.0)*(j+1.0));
   }
   else if( i==(j+2) ){
      return (2.0*j+3.0)*0.5*sqrt((j+1.0)*(j+2.0));
   }
   else if( i==(j+4) ){
      return sqrt((j+1.0)*(j+2.0)*(j+3.0)*(j+4.0))*0.25;
   }
   else {
      return 0;
   }
}

// <i|(xi^5)|j> 部分の計算
double xi_5 (int i, int j){

   if( i==(j-5) ){
      return sqrt(j*(j-1.0)*(j-2.0)*(j-3.0)*(j-4.0)/2.0)*0.25;
   }
   else if( i==(j-3) ){
      return (5.0*j-5.0)*sqrt(j*(j-1.0)*(j-2.0)/2.0)*0.25;
   }
   else if( i==(j-1) ){
      return (10.0*j*j+5.0)*sqrt(j/2.0)*0.25;
   }
   else if( i==(j+1) ){
      return (10.0*j*j+20.0*j+15.0)*sqrt((j+1.0)/2.0)*0.25;
   }
   else if( i==(j+3) ){
      return (5.0*j+10.0)*sqrt((j+1.0)*(j+2.0)*(j+3.0)/2.0)*0.25;
   }
   else if( i==(j+5) ){
      return sqrt((j+1.0)*(j+2.0)*(j+3.0)*(j+4.0)*(j+5.0)/2.0)*0.25;
   }
   else {
      return 0;
   }
}

// <i|(xi^6)|j> 部分の計算
```

```
double xi_6 (int i, int j){

   if( i==(j-6) ){
      return sqrt(j*(j-1.0)*(j-2.0)*(j-3.0)*(j-4.0)*(j-5.0))/8.0;
   }
   else if( i==(j-4) ){
      return (6.0*j-9.0)*sqrt(j*(j-1.0)*(j-2.0)*(j-3.0))/8.0;
   }
   else if( i==(j-2) ){
      return (15.0*j*j-15.0*j+15.0)*sqrt(j*(j-1.0))/8.0;
   }
   else if( i==j ){
      return (20.0*j*j*j+30.0*j*j+40.0*j+15.0)/8.0;
   }
   else if( i==(j+2) ){
      return (15.0*j*j+45.0*j+45.0)*sqrt((j+1.0)*(j+2.0))/8.0;
   }
   else if( i==(j+4) ){
      return (6.0*j+15.0)*sqrt((j+1.0)*(j+2.0)*(j+3.0)*(j+4.0))/8.0;
   }
   else if( i==(j+6) ){
      return sqrt((j+1.0)*(j+2.0)*(j+3.0)*(j+4.0)*(j+5.0)*(j+6.0))/8.0;
   }
   else {
      return 0;
   }
}

/* 右辺 = sigma(hmn/i*hbar)*vn の計算 */

double complex timederivative( int i, double t, double complex *x){

   double complex s;
   int j, js, je;

   js = i-maxpower;
   if (js < 0) js = 0;
   je = i+maxpower;
   if (je > nmax) je = nmax;
   s = 0;
   for ( j=js; j<=je; j++){

      s += ham1[ki*i+j]*x[j];

   }
   return -I/hbar*s;
}

double norm_square(double complex *x){ // 状態ベクトルのノルムの２乗を計算
   int i;
   double s;
   s = 0.0;
   for (i=0; i<N; i++){
     s += creal(x[i])*creal(x[i]) + cimag(x[i])*cimag(x[i]);
   }
   return s;
}
```

```
int normalize(double complex *x){ // 状態ベクトルを規格化
   int i;
   double fct;
   fct = 1.0/sqrt(norm_square(x));
   for ( i=0; i<N; i++){
      x[i] *= fct;
   }
   return 0;
}

double harmonic_oscillator_wavefunction(int n, double x){
  // y/m/d=13/1/17
  //    exp(-x^2/2) H_n(x)/sqrt(sqrt(pi) n^2 n!)
  // using H_n(x) = 2*x*H_{n-1}(x) -2*(n-1)*H_{n-2}(x)
  double a,b,c;
  double fnorm; // normalization factor
  double y; // retuned value psi_n(x)
            // = value of the normalized n-node eigenwavefunction at x
  int k, k1=0, k2=0;
  double x2; // x*x
  const double fct=1.0e50,fctreci=1.0/fct;

  x2=x*x;
  if(n>=2){
    b=1.0;
    c=2.0*x;
    for(k=1;k<n;k++){
      a=b;
      b=c;
      c=2*(x*b-k*a);
      if(fabs(c) > fct){
        k1++; a*=fctreci; b*=fctreci; c*=fctreci;
      }
    }
    fnorm=1;
    for(k=1;k<=n;k++){
      fnorm*=2*k;
      if(fabs(fnorm) > fct){
        k2++; fnorm*=fctreci;
      }
    }
    y=exp(-x2*0.5+(k1-k2*0.5)*log(fct))*c/sqrt(sqrt(M_PI)*fnorm);
  }
  else if(n==1) y=exp(-x2*0.5)*2*x/sqrt(sqrt(M_PI)*2.0);
  else if(n==0) y=exp(-x2*0.5)/sqrt(sqrt(M_PI));
  else { // if(n<0)
    printf("harmonic_oscillator_wavefunction:argument error %d %f\n",n,x);
    return 0.0;
  }
  return y;
}

int write_vector(double time, double complex *v){
// 状態ベクトル (v) を名前に番号が付いたファイルに出力する。
//   time: 時刻. 参考情報として出力ファイルに書き加える。
//   v   : 状態ベクトル（調和振動子基底での展開係数)
```

```
  int i;
  static int filenumber = 0;
  double s;
  FILE *fo;
  char filename[256];

  if(filenumber > 999) return -1; // 出力ファイル数過大。ジョブ設定ミスの疑い。
  snprintf(filename,256,"vec%03d.dat",filenumber++);
  fo=fopen(filename,"w");
  fprintf(fo,"# time=%.5f\n",time);
  s=0.0;
  for(i=0;i<=nmax;i++){
    fprintf(fo,"%3d %.9f %.9f\n",i,creal(v[i]),cimag(v[i]));
    s+=creal(v[i])*creal(v[i])+cimag(v[i])*cimag(v[i]);
  }
  fprintf(fo,"# squared norm=%.10f\n",s);
  fclose(fo);
  return 0;
}

int write_wavefunction(double time, double complex *v){
// 波動関数 (x) を名前に番号が付いたファイルに出力する。
//   time: 時刻. 参考情報として出力ファイルに書き加える。
//   v   : 状態ベクトル（調和振動子基底での展開係数)
  int npsi=500; // 波動関数の値を出力する x 軸上の点の個数-1 (0 番..npsi 番の点)
  double xmin=-80.0; // 波動関数の値を出力する左端の点の座標
  double xmax= 80.0; // 波動関数の値を出力する右端の点の座標
  int i,n;
  static int filenumber = 0;
  double complex s;
  double norm2; // 状態のノルムの 2 乗（x 表示で求めたもの)
  double x,dx;
  FILE *fo;
  char filename[256];

  if(filenumber > 999) return -1; // 出力ファイル数過大。ジョブ設定ミスの疑い。
  snprintf(filename,256,"wf%03d.dat",filenumber++);
  fo=fopen(filename,"w");
  fprintf(fo,"# time=%.5f\n",time);
  dx=(xmax-xmin)/npsi;
  norm2=0.0;
  for(i=0;i<=npsi;i++) {
    x=xmin+dx*i;
    s=0.0;
    for(n=0;n<=nmax;n++){
      s+=v[n]*harmonic_oscillator_wavefunction(n,x/ol_b);
    }
    s/=sqrt(ol_b);
    fprintf(fo,"%.3f %.6f %.6f\n",x,creal(s),cimag(s));
    norm2+=creal(s)*creal(s)+cimag(s)*cimag(s);
  }
  fprintf(fo,"# squared norm=%.10f\n",norm2*dx);
  fclose(fo);
  return 0;
}

int write_potential(double q0, double q1, double q2, double q3,
```

```
                                                double q4, double q5, double q6){
// potential energy を名前に番号が付いたファイルに出力する。
  int npsi=500; // potential の値を出力する x 軸上の点の個数-1 (0 番..npsi 番の点)
  double xmin=-80.0; // potential の値を出力する左端の点の座標
  double xmax= 80.0; // potential の値を出力する右端の点の座標
  int i;
  static int filenumber = 0;
  double x,dx;
  FILE *fo;
  char filename[256];

  if(filenumber > 999) return -1; // 出力ファイル数過大。ジョブ設定ミスの疑い。
  snprintf(filename,256,"pot%03d.dat",filenumber++);
  fo=fopen(filename,"w");

  printf("# %f =q0\n",q0);
  printf("# %f =q1\n",q1);
  printf("# %f =q2\n",q2);
  printf("# %f =q3\n",q3);
  printf("# %f =q4\n",q4);
  printf("# %f =q5\n",q5);
  printf("# %f =q6\n",q6);

  dx=(xmax-xmin)/npsi;
  for(i=0;i<=npsi;i++) {
    x=xmin+dx*i;
    fprintf(fo,"%.3f %.6f\n",x,q0+x*(q1+x*(q2+x*(q3+x*(q4+x*(q5+x*q6))))));
  }
  fclose(fo);
  return 0;
}

// エネルギー期待値の計算
double energy_expectation_value( double complex *x){

   double complex s;
   int i, j, k, js, je;

   s = 0;
   for ( i=0; i<N; i++){
      js = i-maxpower;
      if (js < 0) js = 0;
      je = i+maxpower;
      if (je > nmax) je = nmax;
      k = ki*i+j;
      for ( j=js; j<=je; j++){
          s += conj(x[i])*ham1[ki*i+j]*x[j];
      }
   }
   // s の虚部が 0 であることの確認
   if (fabs(cimag(s)) > 1.0e-20){
      printf("%f %f\n", creal(s), cimag(s));
   }

   return creal(s);
}
```

```
// エネルギーの 2 乗の期待値の計算
double energy_2_expectation_value( double complex *x){

   double complex s, s1;
   int i, j, k, js, je, ks, ke;

   s = 0.0;
   for ( i=0; i<N; i++){
       js = i-maxpower*2;
        if (js < 0) js = 0;
       je = i+maxpower*2;
        if (je > nmax) je = nmax;
      for ( j=js; j<=je; j++){
          if (i > j) ks = i-maxpower;
           else ks = j-maxpower;
          if (ks < 0) ks = 0;

          if (i > j) ke = j+maxpower;
           else ke = i+maxpower;
          if (ke > nmax) ke = nmax;

          s1 = 0.0;
         for (k=ks; k<=ke; k++){
            s1 += ham1[ki*i+k]*ham1[ki*k+j];
          }
         s += s1*conj(x[i])*x[j];
      }
   }
   // s の虚部が 0 であることの確認
   if (fabs(cimag(s)) > 1.0e-20){
      printf("%f %f\n", creal(s), cimag(s));
   }

   return creal(s);
}

// x の期待値の計算
double x_expectation_value( double complex *x){

   double complex s;
   int i, j;

   s = 0;
   for ( i=0; i<N; i++){
      for ( j=0; j<N; j++){
          s += conj(x[i])*x[j]*ol_b*xi_1 (i, j);
      }
   }
   // s の虚部が 0 であることの確認
   if (fabs(cimag(s)) > 1.0e-20){
      printf("%f %f\n", creal(s), cimag(s));
   }

   return creal(s);
}

// x の 2 乗の期待値の計算
```

```
double x_2_expectation_value( double complex *x){

   double complex s;
   int i, j;

   s = 0;
   for ( i=0; i<N; i++){
      for ( j=0; j<N; j++){
          s += conj(x[i])*x[j]*ol_b*ol_b*xi_2 (i, j);
      }
   }
   // s の虚部が 0 であることの確認
   if (fabs(cimag(s)) > 1.0e-20){
      printf("%f %f\n", creal(s), cimag(s));
   }

   return creal(s);
}
```