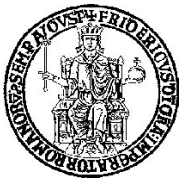
UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

FACOLTÀ DI SCIENZE MATEMATICHE, FISICHE E NATURALI

Corso di Laurea in Fisica



Simulazione di sistemi quantistici assistita dal calcolatore

Tesi di Laurea in Fisica

Relatori:
Prof. Fedele Lizzi
Dott.ssa Ofelia Pisanti

Candidato:
Guido De Rosa
matr. 60/914

Anno Accademico 2007/2008

Simulazione di sistemi quantistici assistita dal calcolatore

Guido De Rosa

a mio nonno Guido
e a Fabio

Indice

Elenco delle figure

Introduzione

La Meccanica Quantistica ha avuto una gran mole di successi sperimentali ed è, con i suoi sviluppi, l'unica teoria in grado di descrivere i fenomeni legati alla natura intima della materia. È però fondata su un formalismo matematico sofisticato e in genere piuttosto lontano dall'intuizione. Esistono diversi modi, più o meno generali, di formulare i concetti fondamentali della teoria; il più noto è certamente basato sulla descrizione di Schrödinger e sulla rappresentazione nello spazio delle posizioni, ovvero sull'intuizione di L. V. De Broglie circa la duplice natura, corpuscolare e ondulatoria, delle particelle materiali.

La maggior parte delle simulazioni al calcolatore, a scopo didattico o di ricerca, si basa sulla *Meccanica Ondulatoria*, proprio perché è la via più efficace per fornire una visione immediata di come la teoria "funzioni" e sia in grado di prevedere i fenomeni naturali. Il significato della *funzione d'onda* come probabilità di trovare una particella in una regione dello spazio ci motiva a investigare, per via simulata oltre che sperimentale, le eventuali analogie o le profonde differenze fra un *pacchetto* d'onda localizzato in una regione finita ed una particella classica — pensiamo agli stati coerenti di un oscillatore armonico piuttosto che all'effetto tunnel. Per non dire delle domande che ci possiamo porre quando confrontiamo le *onde di materia* con altri oggetti di natura classicamente ondulatoria come la radiazione elettromagnetica — si pensi alla diffrazione di un fascio di elettroni da parte di un reticolo cristallino piuttosto che alle relazioni di dispersione non lineari col conseguente *slargamento* del pacchetto associato ad una particella libera.

In questo lavoro di tesi è descritto lo studio effettuato per simulare l'evoluzione temporale di due particolari sistemi: una particella in una dimensione in presenza di una barriera di potenziale ed una buca bidimensionale a pareti infinite. Considerare sistemi ad una o due dimensioni è utile non solo perché il calcolo è più semplice e ci si può concentrare sui concetti fondamentali, ma anche perché, di fatto, molti sistemi naturali o riproducibili in laboratorio hanno caratteristiche tali da poter essere studiati senza il reale bisogno di considerare tutte le dimensioni dello spazio fisico. Sistemi siffatti rendono più agevole la rappresentazione grafica delle funzioni che li descrivono e dunque le simulazioni che realizzeremo.

È tipico rappresentare una funzione d'onda in una dimensione mediante il grafico del quadrato del modulo, ma in questo modo si perde una parte importante dell'informazione sul sistema. Se è vero che una funzione d'onda normalizzata è significativa a meno di un fattore di fase *costante*, è anche vero che, per esempio, $\psi(x)$ ed $e^{i\alpha x}\psi(x)$ descrivono particelle aventi la stessa distribuzione di probabilità per la posizione ma una diversa distribuzione del momento lineare! (In termini matematici, basti applicare la proprietà di traslazione della trasformata di Fourier). Per questi motivi si è scelto di rappresentare l'evoluzione nel tempo della funzione *complessa*. Per un sistema unidimensionale si possono utilizzare grafici a tre dimensioni, uno dei cui assi sarà associato alla variabile indipendente mentre gli altri due corrisponderanno alla parte reale e alla parte immaginaria della funzione. In questo modo il grafico di e^{ikx} avrebbe la forma di un'elica, ma si possono fare numerosi altri esempî.

Nella prima simulazione è stato considerato un pacchetto d'onda con una energia (o meglio con una distribuzione statistica di valori dell'energia) strettamente inferiore al livello della barriera: in un analogo sistema classico l'attraversamento di quest'ultima è impossibile. La soluzione delle equazioni quantistiche mostra invece che esiste una probabilità di trovare la particella *oltre* la barriera. Fra i successi della meccanica quantistica possiamo annoverare l'interpretazione

del decadimento α proprio in termini di questo *effetto tunnel*. Nella nostra schematizzazione i parametri del sistema sono stati scelti in modo tale da rendere un simile effetto piuttosto evidente.

La seconda simulazione riguarda un sistema bidimensionale. Una volta usati gli assi x e y per le variabili indipendenti, sull'asse z può essere rappresentato, nel modo usuale, $|\psi(x,y)|^2$, mentre l'argomento complesso può essere reso da una scala di colori. Si è combinato linearmente un numero finito e piccolo di autostati dell'energia, ma per il resto sono state scelte combinazioni del tutto arbitrarie. Solo in alcuni casi si è ottenuto un comportamento che potremmo definire semiclassico, e in tali casi si è potuto vedere che, cambiando solo l'argomento complesso di uno dei coefficienti (lasciando inalterata, dunque, la distribuzione statistica dell'energia), si hanno "traiettorie" del tutto diverse.

Prima di illustrare, nel Cap. 2, le simulazioni e la loro implementazione, dedicheremo il Cap. 1 a un breve richiamo dei postulati fondamentali della meccanica quantistica, facendo riferimento ad alcuni concetti di teoria spettrale. Seguiranno considerazioni relative alla rappresentazione nello spazio delle configurazioni che ci consentiranno di esprimere, nel linguaggio della meccanica ondulatoria, l'equazione agli autovalori dell'hamiltoniana e l'evoluzione temporale delle autofunzioni, ovvero gli strumenti indispensabili per impostare il seguito di questo lavoro.

Tutte le animazioni, i grafici, i file di dati e il codice sorgente sono disponibili alla pagina web `http://people.na.infn.it/~pq-qp/pages/simulations.html`**, parte del *portale* didattico "$PQ - QP$"[3].**

Capitolo 1

I sistemi quantistici

1.1 Il teorema di risoluzione spettrale e la formulazione generale della Meccanica Quantistica

1.1.1 Teorema di risoluzione spettrale

Ogni operatore autoaggiunto è esprimibile nella forma[7, 1]

$$\hat{A} = \int_{-\infty}^{\infty} a \, d\hat{E}(a), \tag{1.1}$$

dove $\hat{E}$ indica una misura a valori di proiezione definita sui boreliani B dell'asse reale, che soddisfa le condizioni

$$\hat{E}(\mathbb{R}) = \mathbf{1},$$
$$\hat{E}(B_1 \cap B_2) = \hat{E}(B_1)\hat{E}(B_2)\,.$$

1.1.2 I postulati della Meccanica Quantistica

Nella *descrizione*[1] *di Schrödinger* le proprietà quantistiche di una particella possono ricavarsi dai seguenti postulati[2]:

i. Ad ogni sistema fisico è associato un opportuno spazio di Hilbert, $\mathcal{H}$. Ogni stato del sistema può essere rappresentato mediante un elemento ψ

[1] È opportuno distinguere *descrizione* da *rappresentazione*: il formalismo da cui prende le mosse il presente lavoro prescinde, come vedremo, dalla particolare *rappresentazione*.

di $\mathcal{H}$ con norma uguale a 1, detto *vettore di stato*, che contiene tutte le possibili informazioni sul sistema.

L'evoluzione temporale di detto vettore è regolata da una equazione della forma

$$i\hbar\frac{d\psi}{dt} = \hat{H}\psi \tag{1.2}$$

dove $\hat{H}$ è un opportuno operatore autoaggiunto detto *operatore di Hamilton* o *hamiltoniana* del sistema e l'equazione prende il nome di *equazione di Schrödinger*.

ii. Ad ogni grandezza osservabile A corrisponde un operatore autoaggiunto $\hat{A}$ nello spazio $\mathcal{H}$, il cui spettro discreto σ_d e continuo σ_c costituiscono l'insieme dei possibili risultati di una misura di A.

La probabilità che una misura fornisca un risultato nell'insieme $\mathcal{A}$ è data da

$$P_{\mathcal{A}} = \int_{\mathcal{A}} \left|\left| d\hat{E}(a)\psi \right|\right|^2, \tag{1.3}$$

dove $\hat{E}$ è la misura a valori di proiezione associata ad $\hat{A}$ mediante la (1.1).

iii. Se da una misura risulta che il valore della grandezza A appartiene all'insieme $\mathcal{A}$, e con i simboli ψ_i e ψ_f si indicano rispettivamente il vettore di stato prima e dopo l'osservazione, si ha, a meno di normalizzazioni,

$$\psi_f = \hat{E}(\mathcal{A})\psi_i.$$

In altre parole, *la misura agisce sullo spazio degli stati come un* proiettore *sull'autospazio relativo ai risultati ottenuti.*

iv. Ad una particella nello spazio fisico tridimensionale sono associati gli operatori

$$\hat{\mathbf{r}} = (\hat{x}, \hat{y}, \hat{z}), \tag{1.4}$$

$$\hat{\mathbf{p}} = (\hat{p}_x, \hat{p}_y, \hat{p}_z), \tag{1.5}$$

che formano in $\mathcal{H}$ un insieme irriducibile[2], soddisfano le regole di commutazione

$$[\hat{r}_i, \hat{r}_j] = [\hat{p}_i, \hat{p}_j] = 0, \tag{1.6}$$

$$[\hat{r}_i, \hat{p}_j] = i\hbar\,\delta_{ij}, \tag{1.7}$$

e corrispondono alle coordinate cartesiane e ai loro momenti coniugati.

v. Gli operatori corrispondenti alle altre osservabili[3] sono *funzioni* degli operatori fondamentali (1.4) e (1.5).

vi. L'operatore $\hat{H}$ che compare nella (1.2) è della forma[4]

$$\hat{H} = \frac{\hat{\mathbf{p}}^2}{2m} + U(\hat{\mathbf{r}}). \tag{1.8}$$

I postulati suddetti si riferiscono a sistemi ad una particella ma sono facilmente generalizzabili al caso di N particelle. In particolare la (1.8) diventa $\hat{H} = \frac{1}{2m}\sum_{k=1}^{N} \hat{\mathbf{p}}_\mathbf{k}^2 + U(\hat{\mathbf{r}}_\mathbf{1}, \ldots, \hat{\mathbf{r}}_\mathbf{N})$, mentre le relazioni di commutazione (1.6) e (1.7) diventano $[\hat{r}_i^{(k)}, \hat{r}_j^{(l)}] = [\hat{p}_i^{(k)}, \hat{p}_j^{(l)}] = 0$ e $[\hat{r}_i^{(k)}, \hat{p}_j^{(l)}] = i\hbar\,\delta_{ij}\delta_{kl}$. Nel seguito, tuttavia, faremo sempre riferimento per semplicità a sistemi costituiti da una singola particella.

1.2 La rappresentazione nello spazio delle posizioni

Come spazio degli stati possiamo scegliere $\mathcal{L}^2(\mathbb{R}^3)$ e come operatori fondamentali gli operatori di moltiplicazione (x, y, z) e gli operatori differenziali $\frac{\hbar}{i}(\partial_x, \partial_y, \partial_z)$

[2] Cioè non esiste alcun sottospazio proprio di $\mathcal{H}$ invariante sia per $\hat{\mathbf{r}}$ che per $\hat{\mathbf{p}}$.

[3] Stiamo considerando grandezze che hanno un analogo classico. Lo *spin*, ad esempio, è un'altra osservabile fondamentale, considerare la quale richiederebbe qualche ulteriore precisazione.

[4] In presenza di campo magnetico, per una particella di carica e (senza spin) la (1.8) diventa

$$\hat{H} = \frac{(\hat{\mathbf{p}} - e\hat{\mathbf{A}})^2}{2m} + eV(\hat{\mathbf{r}}) + U(\hat{\mathbf{r}})$$

dove $\hat{\mathbf{A}}$ corrisponde al *potenziale vettore* del campo elettromagnetico, V al potenziale scalare, e si è posto $c = 1$.

a rappresentare rispettivamente posizione e momento lineare. È immediato verificare che le relazioni di commutazione (1.6) e (1.7) sono soddisfatte.

In questa rappresentazione, la (1.2) si esprime:

$$i\hbar\frac{\partial\Psi(x,y,z;t)}{\partial t} = \left[-\frac{\hbar^2}{2m}\nabla^2 + U(x,y,z)\right]\Psi(x,y,z;t), \tag{1.9}$$

mentre la (1.3), riferita agli operatori di posizione, si scrive:

$$P_X(t) = \int_X |\Psi(x,y,z;t)|^2\, dxdydz$$

mostrando che $|\Psi(x,y,z;t)|^2$ ha il notevole significato di *densità di probabilità* di trovare la particella in una data posizione al tempo t.

1.3 Evoluzione temporale

Se Ψ è un'autofunzione dell'*hamiltoniana*

$$H = -\frac{\hbar^2}{2m}\nabla^2 + U$$

relativa all'autovalore ϵ, la (1.9) diventa

$$i\hbar\frac{\partial\Psi}{\partial t} = \epsilon\Psi$$

e la sua soluzione generale (noto lo stato iniziale $\Psi(t_0)$) sarà del tipo

$$\Psi(t) = \Psi(t_0)e^{-\frac{i\epsilon(t-t_0)}{\hbar}}.$$

Si può dimostrare che, come conseguenza della (1.1), una generica funzione di $\mathcal{L}^2(\mathbb{R})$ è esprimibile come sovrapposizione lineare di autofunzioni dell'hamiltoniana:

$$\Psi = \sum_n c_n\varphi_n + \int_{\sigma_c} d\epsilon\, f(\epsilon)\varphi_\epsilon$$

a patto di considerare anche funzioni φ_ϵ non a quadrato sommabile ma che, integrate mediante una opportuna funzione f, diano come risultato una funzione di $\mathcal{L}^2$ (*autofunzioni improprie*).

Per la linearità della (1.9), l'evoluzione temporale della Ψ sarà

$$\Psi(t) = \sum_n c_n \Psi_n(t_0) e^{-\frac{i\epsilon_n(t-t_0)}{\hbar}} + \int_{\sigma_c} d\epsilon\, f(\epsilon)\Psi_\epsilon(t_0) e^{-\frac{i\epsilon(t-t_0)}{\hbar}}, \tag{1.10}$$

che, definendo opportunamente l'esponenziale di un operatore hermitiano, si può esprimere come

$$\Psi(t) = U(t,t_0)\Psi(t_0) = e^{-\frac{i\hat{H}(t-t_0)}{\hbar}}\Psi(t_0),$$

dove $U(t,t_0)$ è detto *operatore di evoluzione temporale*, e l'ultima equazione vale a prescindere dalla rappresentazione.

In conclusione, per conoscere l'evoluzione temporale di un sistema quantistico descritto dalla hamiltoniana H una strada da seguire consiste nel risolvere l'equazione agli autovalori $H\psi = \epsilon\psi$ (*equazione di Schrödinger indipendente dal tempo*), per poi esprimere la funzione d'onda all'istante iniziale come combinazione lineare di autofunzioni, la cui evoluzione è data dalla (1.10).

1.4 Barriera di potenziale unidimensionale ed effetto tunnel

Nella prima simulazione consideriamo una particella in una dimensione, di massa m, soggetta al potenziale:

$$U(x) = \begin{cases} V, & \text{per } x \in (x_1, x_2) \\ 0, & \text{altrove} \end{cases}$$

Lo spettro dell'energia è puramente continuo e comprende tutti i valori maggiori di 0. Per un analogo sistema classico l'attraversamento della barriera è possibile solo per $E > V$, mentre in Meccanica Quantistica esiste una probabilità non nulla che ciò accada anche per $0 < E < V$. Tale fenomeno va sotto il nome di *effetto tunnel*. Per darne una dimostrazione dobbiamo innanzi tutto individuare una schiera di autofunzioni (improprie) di energia $E \in]0, V[$.

Possiamo risolvere l'equazione di Schrödinger indipendente dal tempo,

$$\left[-\frac{\hbar^2}{2m}\frac{d^2}{dx^2} + U(x)\right]\psi(x) = E\psi(x),$$

separatamente nelle tre regioni $x < x_1$, $x \in (x_1, x_2)$ e $x > x_2$ e ottenere

$$\psi_E(x) = \begin{cases} A_1 e^{ik_E x} + A_2 e^{-ik_E x} & \text{per } x < x_1 \\ B_1 e^{\chi_E x} + B_2 e^{-\chi_E x} & \text{per } x \in (x_1, x_2)\ , \\ C_1 e^{ik_E x} + C_2 e^{-ik_E x} & \text{per } x > x_2 \end{cases} \tag{1.11}$$

dove si è posto

$$k_E = \frac{\sqrt{2mE}}{\hbar}, \qquad \chi_E = \frac{\sqrt{2m(V-E)}}{\hbar}.$$

Resta da imporre che ψ appartenga al dominio di autoaggiuntezza dell'operatore hamiltoniano $-\frac{\hbar^2}{2m}\frac{d^2}{dx^2} + U$. Si può dimostrare che ciò equivale a richiedere la continuità della funzione con la sua derivata, ovunque e, in modo particolare, nei punti x_1 e x_2, dove è verificata solo per alcuni valori dei coefficienti A_1, A_2, B_1, B_2, C_1 e C_2.

Sappiamo che una funzione d'onda moltiplicata per un arbitrario fattore complesso rappresenta lo stesso stato. Dunque possiamo scegliere liberamente il valore di uno dei coefficienti: nel nostro caso imporremo $A_1 = 1$.

Vogliamo costruire un pacchetto d'onda che abbia un "analogo classico". Scegliamo, nella nostra simulazione, di rappresentare una particella che proviene dalla sinistra della barriera. Questo ci porta ad imporre $C_2 = 0$, in quanto una sovrapposizione di funzioni del tipo e^{-ikx} nella regione $x > x_2$ corrisponderebbe ad una particella *proveniente da destra* (sono tutte autofunzioni relative ad autovalori *negativi* del momento lineare).

Abbiamo così ridotto a quattro il numero delle incognite. La richiesta di continuità di ψ e ψ' in x_1 e x_2 equivale ad altrettante equazioni lineari:

$$\begin{aligned} e^{ik_E x_1} + A_2 e^{-ik_E x_1} &= B_1 e^{\chi_E x_1} + B_2 e^{-\chi_E x_1} \\ B_1 e^{\chi_E x_2} + B_2 e^{-\chi_E x_2} &= C_1 e^{ik_E x_2} \\ ik_E e^{ik_E x_1} - ik_E A_2 e^{-ik_E x_1} &= B_1 \chi_E e^{\chi_E x_1} - B_2 \chi_E e^{-\chi_E x_1} \\ B_1 \chi_E e^{\chi_E x_2} - B_2 \chi_E e^{-\chi_E x_2} &= iC_1 k_E e^{ik_E x_2}. \end{aligned} \tag{1.12}$$

1.5 Buca di potenziale bidimensionale a pareti infinite

Per *buca quadrata* s'intende in generale il sistema costituito da una particella in due dimensioni soggetta al potenziale

$$U(x,y) = \begin{cases} 0, & \text{per } (x,y) \in (-a,a) \times (-a,a) \\ V, & \text{altrove} \end{cases} .$$

La funzione d'onda deve appartenere al dominio di autoaggiuntezza dell'hamiltoniana. Ciò equivale alla richiesta di continuità della stessa e delle sue derivate spaziali. Si può dimostrare che, al limite per $V \to \infty$, le condizioni al contorno si riducono a:

$$\psi(x,y) = 0, \text{ per } (x,y) \in \mathbb{R}^2 - (-a,a) \times (-a,a) .$$

In termini intuitivi, questa condizione descrive una particella costretta nel quadrato $(-a,a)\times(-a,a)$ da un potenziale che valga zero all'interno e sia "infinito" al di fuori di tale porzione di piano.

Per semplicità di calcolo scegliamo unità di misura tali da soddisfare la relazione $\frac{\hbar^2}{2m} = 1$; si pone inoltre $a = \frac{\pi}{2}$. In questo modo l'operatore hamiltoniano ha la seguente espressione:

$$H = H_1 + H_2, \quad \text{con} \quad H_1 = -\frac{\partial^2}{\partial x^2} \quad \text{e} \quad H_2 = -\frac{\partial^2}{\partial y^2}$$

e le sue autofunzioni sono tutte del tipo:

$$\psi_{mn}(x,y) = \phi_m(x)\phi_n(y), \tag{1.13}$$

con

$$\phi_k(\xi) = \begin{cases} \sqrt{\frac{2}{\pi}}\cos(k\xi), & \text{per } k \text{ dispari} \\ \sqrt{\frac{2}{\pi}}\sin(k\xi), & \text{per } k \text{ pari} \end{cases} . \tag{1.14}$$

I corrispondenti autovalori dell'energia[5] sono dati da $E_{mn} = m^2+n^2$. Infatti:

$$H\psi_{mn}(x,y) = \left(-\frac{\partial^2}{\partial x^2} - \frac{\partial^2}{\partial y^2}\right)\phi_m(x)\phi_n(y) = \\ m^2\phi_m(x)\phi_n(y) + n^2\phi_m(x)\phi_n(y) = (m^2+n^2)\psi_{mn}(x,y).$$

Ciascuna delle autofunzioni ha la seguente evoluzione temporale:

$$\Psi_{mn}(x,y;t) = \psi_{mn}(x,y)e^{-iE_{mn}t} = \phi_m(x)\phi_n(y)e^{-i(m^2+n^2)t}. \tag{1.15}$$

Per la simulazione prenderemo in considerazione alcune semplici combinazioni lineari (finite) di autostati dell'energia, cosa resa possibile dallo spettro discreto dell'operatore hamiltoniano.

[5] Almeno quando $m \neq n$, si tratta chiaramente di livelli degeneri: è sufficiente scambiare gli indici per avere due stati ψ_{mn} e ψ_{nm} linearmente indipendenti ma relativi alla stessa energia, il che riflette la simmetria del sistema.

Capitolo 2

Le simulazioni

2.1 Il problema della rappresentazione grafica delle funzioni complesse

È tipico rappresentare le funzioni d'onda quantistiche mediante il grafico del quadrato del modulo, sebbene in questo modo si perda una parte dell'informazione. Il problema può essere risolto cercando di superare le difficoltà tecniche relative alla rappresentazione dei suoi valori complessi.

Un numero complesso può essere individuato dalla sua parte reale e da quella immaginaria oppure dal modulo e dall'argomento. Sono comunque necessarie due dimensioni spaziali per rappresentarlo graficamente. Sono possibili diverse soluzioni, a seconda della geometria e della dimensionalità del sistema fisico.

Lo stato dinamico di una particella quantistica in una dimensione può essere descritto da una funzione complessa definita sull'asse reale. Tale funzione può essere rappresentata mediante un grafico tridimensionale il cui asse x corrisponde alla variabile indipendente, mentre gli assi y e z sono utilizzati per rappresentare le parti reale e immaginaria del suo valore. Alcuni esempî verranno illustrati nel prosieguo del capitolo (§2.2.3).

Per un sistema bidimensionale si avrebbe bisogno di quattro dimensioni: in

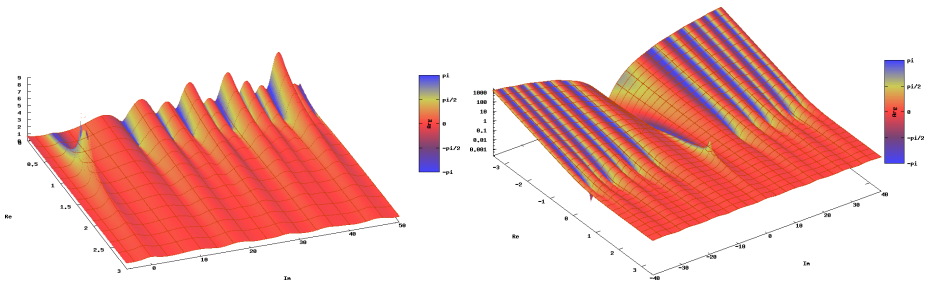


Figura 2.1: *Funzione Zeta di Riemann* $\zeta(s) = \sum_{n=1}^{\infty} \frac{1}{n^s}$, con $s = x + iy$ nei dominî $(x, y) \in [0, 3] \times [-5, 50]$ e $(x, y) \in \left[-\frac{7}{2}, \frac{7}{2}\right] \times [-40, 40]$ (in quest'ultimo caso in scala logaritmica).

realtà è possibile realizzare il grafico tridimensionale di $|\Psi(x, y)|^2$ e servirsi di una scala di colori per rappresentare l'argomento complesso. È necessario che gli estremi di questa scala siano identificati — corrispondano cioè allo stesso colore — in modo da evitare una errata percezione di discontinuità dell'argomento nel passaggio da $-\frac{\pi}{2}$ a $\frac{\pi}{2}$. Questa tecnica può essere utilizzata anche per rappresentare funzioni complesse di variabile complessa (funzioni analitiche, funzioni speciali etc.), di cui si fornisce un esempio in Fig. 2.1.

2.2 Barriera di potenziale unidimensionale ed effetto tunnel

2.2.1 Calcolo simbolico con *Maxima*

Maxima[5] è un *Computer Algebra System* a sorgente aperto[8] con il quale calcoleremo, per via esatta e simbolica, i parametri della (1.11) e, per via approssimata e numerica, una sovrapposizione di autofunzioni per costruire un pacchetto d'onda e studiarne l'evoluzione temporale.

Maxima è un programma *a riga di comando*, per il quale sono disponibili alcuni *frontend* grafici, come *wxMaxima*[9].

Di seguito illustreremo i passaggi concettualmente più importanti dello script utilizzato, riportato in A.1.1, cominciando dalla parte *simbolica*.

Esprimiamo innanzi tutto in codice *Maxima* la (1.11) e la sua derivata:

```
/*
 * general solutions for potential barrier (0<E<V)
 */
psi1(x) := exp(%i*x*sqrt(2*m*E)/hbar) + A2*exp(-%i*x*sqrt(2*m*E)/hbar);
psi2(x) := C1*exp(x*sqrt((V-E)*m)/hbar) + C2*exp(-x*sqrt((V-E)*m)/hbar);
psi3(x) := B1*exp(%i*x*sqrt(2*m*E)/hbar);
/*
 * and their derivatives
 */
Dpsi1(x) := diff(psi1(x),x);
Dpsi2(x) := diff(psi2(x),x);
Dpsi3(x) := diff(psi3(x),x);
```

Impostiamo e risolviamo il sistema (1.12):

```
/*
 * Regularity condition on wavefunction and derivative
 */
reg12 : psi1(x1)=psi2(x1);
reg23 : psi2(x2)=psi3(x2);
Dreg12 : Dpsi1(x1)=Dpsi2(x1);
Dreg23 : Dpsi2(x2)=Dpsi3(x2);
/*
 * Simbolically solve the liner equations in A2,C1,C2,B1
 */
linsolve([reg12,reg23,Dreg12,Dreg23],[A2,C1,C2,B1]);
```

È importante che *Maxima* risolva il sistema simbolicamente poiché le soluzioni ottenute sono funzioni di E e di altri parametri del problema che dobbiamo ancora fissare. Ad esempio, alla richiesta[1]:

```
tex(psi1(x));
```

si ottiene il seguente output[2]:

$$e^{\frac{\sqrt{2}\,i\,\sqrt{m}\,x\,\sqrt{E}}{\hbar}}- \left(e^{-\frac{\sqrt{2}\,i\,\sqrt{m}\,x\,\sqrt{E}}{\hbar}}\left((-V-E)\,e^{\frac{2\,\sqrt{m}\,x_2\,\sqrt{V-E}}{\hbar}+\frac{2\,\sqrt{2}\,i\,\sqrt{m}\,x_1\,\sqrt{E}}{\hbar}}+(V+E)\,e^{\frac{2\,\sqrt{m}\,x_1\,\sqrt{V-E}}{\hbar}+\frac{2\,\sqrt{2}\,i\,\sqrt{m}\,x_1\,\sqrt{E}}{\hbar}}\right)\right)/ \left(\left(2\,\sqrt{2}\,i\,\sqrt{E}\,\sqrt{V-E}-V+3\,E\right)e^{\frac{2\,\sqrt{m}\,x_2\,\sqrt{V-E}}{\hbar}}+\left(2\,\sqrt{2}\,i\,\sqrt{E}\,\sqrt{V-E}+V-3\,E\right)e^{\frac{2\,\sqrt{m}\,x_1\,\sqrt{V-E}}{\hbar}}\right).$$

Come si vede, questa è una espressione complicata, generale e non semplificata, che possiamo considerare come un elemento intermedio e interno alla elaborazione complessiva.

[1] Del tutto ridondante se si esegue il codice di §A.1.1 in modalità non interattiva, ma utile a scopo esplicativo o di *debug*.

[2] Sfruttiamo la possibilità, per *Maxima*, di produrre codice TEX che ci consente, con piccole modifiche, di formattarne i risultati nel presente documento.

Con ovvio significato dei simboli possiamo ora scrivere

$$\psi_E(x) = \begin{cases} e^{ik_E x} + A_2(E)e^{-ik_E x} & \text{per } x < x_1 \\ B_1(E)e^{\chi_E x} + B_2(E)e^{-\chi_E x} & \text{per } x \in (x_1, x_2) \\ C_1(E)e^{ik_E x} & \text{per } x > x_2 \end{cases},$$

per considerare l'evoluzione temporale di ciascuna autofunzione,

$$\Psi_E(x;t) = \psi_E(x)e^{-\frac{iEt}{\hbar}},$$

in termini di codice *Maxima*:

```
/* time evolution of each eigenfunction */
Epsi1(x,E,t) := psi1(x)*exp(-%i*E*t/hbar);
Epsi2(x,E,t) := psi2(x)*exp(-%i*E*t/hbar);
Epsi3(x,E,t) := psi3(x)*exp(-%i*E*t/hbar);
```

2.2.2 Calcolo numerico con *Maxima*

A questo punto, completata la parte simbolica della elaborazione, possiamo fissare i parametri del problema. Una scelta opportuna può rendere più o meno evidenti taluni effetti fisici nel risultato finale.

```
m       : 2;
hbar    : 0.1;
V       : 1;
x1      : 5.0;
x2      : 5.1;
```

Vogliamo ora costruire un pacchetto d'onda localizzato, sovrapponendo autofunzioni relative a un intervallo di energie strettamente incluso in $(0, V)$:

$$\Psi(x;t) = \int_{E_0-\Delta E}^{E_0+\Delta E} f(E)\Psi_E(x;t)\,dE \tag{2.1}$$

dove, per la nostra simulazione si è scelto:

```
E0      : 0.5;
sigmaE  : 0.1;
deltaE  : 1.5 * sigmaE;
/*
 * Make a superposition of eigenfunctions modulated by f(E).
 * A gaussian is a good choice to obtain a well localized packet.
 */
f(E) := exp(-(E-E0)^2/sigmaE^2);
```

Non resta che valutare numericamente la (2.1) in un reticolo discreto di valori per x, t, e con un opportuno passo.

```
/* boundaries for plotting ... */
x0                  : -2;
x3                  : 10;
xstep               : 0.0141;
...
ti                  : 1.30;
tf                  : 14.8;
tstep               : 0.03750;
Emin                : E0 - deltaE;
Emax                : E0 + deltaE;
...
```

Il seguente frammento si riferisce all'integrazione numerica nel tratto $x < x_1$ (codice analogo è eseguito per le altre due regioni):

```
...
  /* The superposition (wavepacket) is constructed  */
  for x: x0 step xstep thru x1 do (
    result_re : quad_qags(  /*numerically integrate real... */
      realpart(f(E)*Epsi1(x,E,t)),
      E,
      Emin,
      Emax,
      epsrel,
      limitq
    ),
    result_im : quad_qags( /*...and imaginary part*/
      imagpart(f(E)*Epsi1(x,E,t)),
      E,
      Emin,
      Emax,
      epsrel,
      limitq
    ),
    print(x, "= x < x1; t =", t, " results:", result_re, result_im),
    printf(stream,format,t,x,result_re[1],result_im[1])
  ),
...
```

Abbiamo omesso alcuni dettagli tecnici per i quali si rimanda al codice completo in §A.1.1 oltre che, naturalmente, alla documentazione di *Maxima*[5].

I risultati del calcolo sono scritti nella forma

$$t \qquad x \qquad \Re(\Psi(x;t)) \qquad \Im(\Psi(x;t))$$

in un file di testo che possa esser letto da *Gnuplot*[4], per poi realizzare immagini e animazioni.

```
...
stream: openw(output_file);
format: "~f ~f ~f ~f~%"; /* float */
...
  /* The superposition (wavepacket) is constructed  */
  for x: x0 step xstep thru x1 do (
...
    printf(stream,format,t,x,result_re[1],result_im[1])
...
```

2.2.3 Rappresentazione grafica

Lo script *Perl* riportato in §A.1.2 integra diverse componenti software, realizza una sequenza di grafici con *Gnuplot*[4] e ne ricava file di animazioni mediante diversi codificatori video. È stato sviluppato in ambiente Unix, ma con qualche accorgimento è possibile il *porting* verso altri sistemi operativi.

Per conoscerne la sintassi, eseguiamolo senza argomenti:

```
$ ./tunnel-plot.pl
Usage: ./tunnel-plot.pl {file.dat} [ size-X size-Y ]
```

Bisogna fornire dunque il file di dati prodotto da *Maxima* ed eventualmente le dimensioni desiderate per i grafici di funzione. I parametri scelti in §2.2.2 possono portare a una esecuzione che può durare diverse ore, ma ci consentono di avere dati sufficienti a produrre immagini ad una definizione paragonabile alla massima disponibile sullo schermo di un moderno personal computer.

Per prima cosa, lo script suddivide il file in ingresso in parti più piccole, ognuna delle quali contiene dati che si riferiscono ad uno stesso istante di tempo: in pratica, ciascuna di esse servirà a produrre un singolo "fotogramma".

```
...
open(FH,"<$infile");
#
while(<FH>) {
    chomp;
    if (m/^\s*(\d*\.\d+)/) {
        $line = $_;
        $t = $1;
        $n = sprintf("%07.4f",$t);
        $file = "$datadir/$n$ext";
        unless ($handles{"$n"}) {
            message("splitting: $n$ext ...  \r");
            open($handles{"$n"},">$file") or die ("Couldn't open $file: $!\n");
```

```
        }
        $fh = $handles{"$n"};
        print $fh "$line\n";
    }
}
...
```

Successivamente viene invocata l'esecuzione di *Gnuplot*[4] per produrre i grafici tridimensionali:

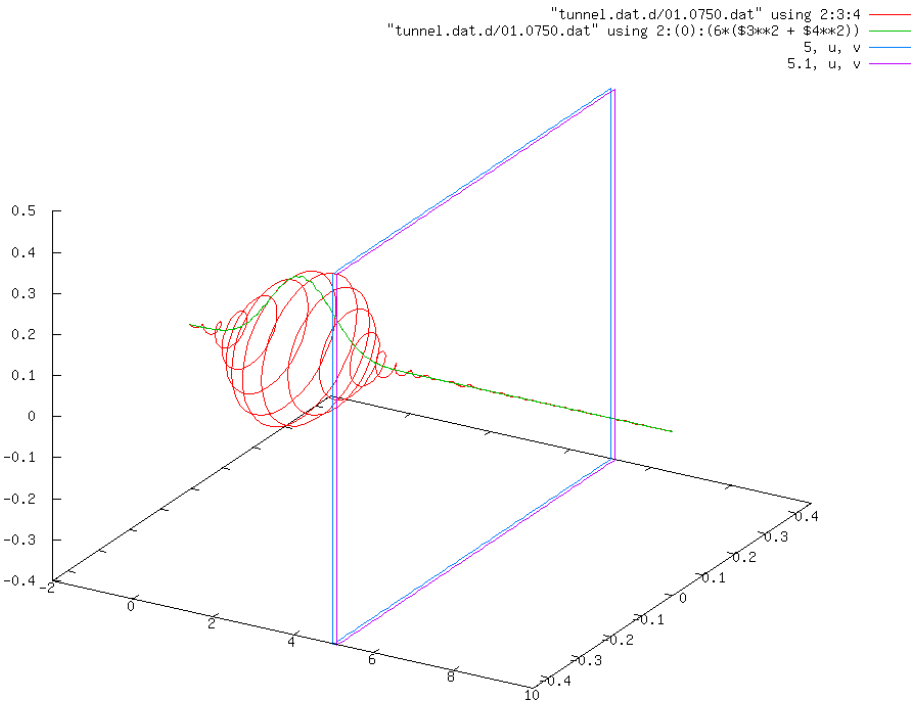
```
...
foreach $n (sort(keys(%handles))) {
    $fh = $handles{$n};
    close($fh);
    message("plotting: $n.png ...   \r");
    print GNUPLOT "set output \"$imgdir/$n.png\"\n";
    print GNUPLOT "splot \"$datadir/$n$ext\" using 2:3:4 w lines, ".
    " \"$datadir/$n$ext\" using 2:(0):($smscale*(\$3**2 + \$4**2)) w lines,".
    " $x1, u, v w lines, $x2, u, v w lines\n";
}
...
```

nei quali l'asse x è riferito alla variabile indipendente, mentre gli assi y e z rappresentano rispettivamente, per ogni istante t, la parte reale e immaginaria di $\Psi(x;t)$.

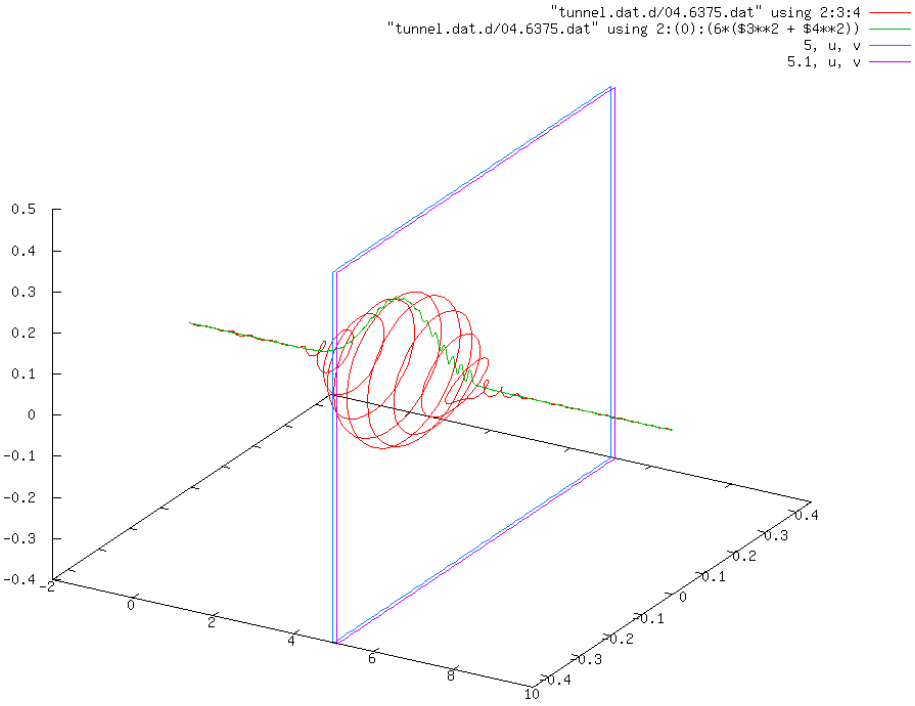
Alcuni esempî sono riportati nelle Figure 2.2-2.5 dove, per raffronto, è riportata anche la usuale rappresentazione del modulo quadro (nel piano xz, in verde). In Fig. 2.2 è mostrato l'avvicinamento della particella alla barriera, con evidente analogia al caso classico. In Fig. 2.3 si vede come la forma d'onda sia alterata dall'impatto con la stessa. Nelle Figure 2.4 e 2.5 è mostrato l'*effetto tunnel*. In un analogo sistema classico la particella tornerebbe indietro, mentre in questo sistema quantistico la funzione d'onda sembra "dividersi in due". Dal punto di vista statistico ciò significa che esiste una probabilità non nulla di trovare la particella oltre la barriera.

Con i parametri scelti disponiamo di 369 immagini che, alla frequenza tipica di venticinque *frames* al secondo, ci consentono di produrre una animazione della durata di 14.76 secondi.

```
...
    system("$mencoder $uri -o $lossless -ovc lavc -lavcopts vcodec=ffv1");
...
# codice per la conversione ad altri formati...
...
```

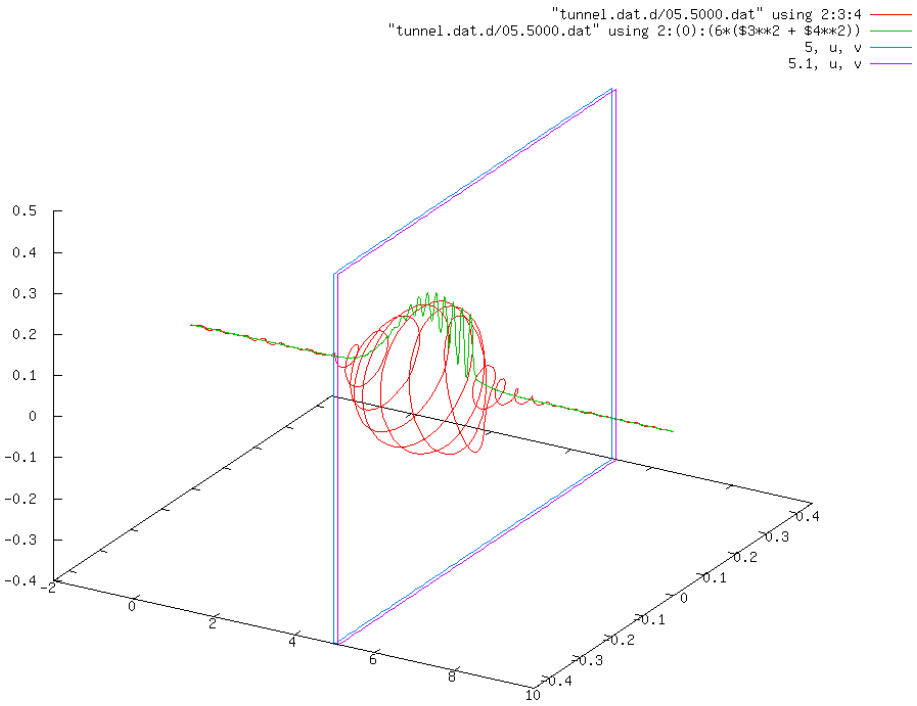


(a)

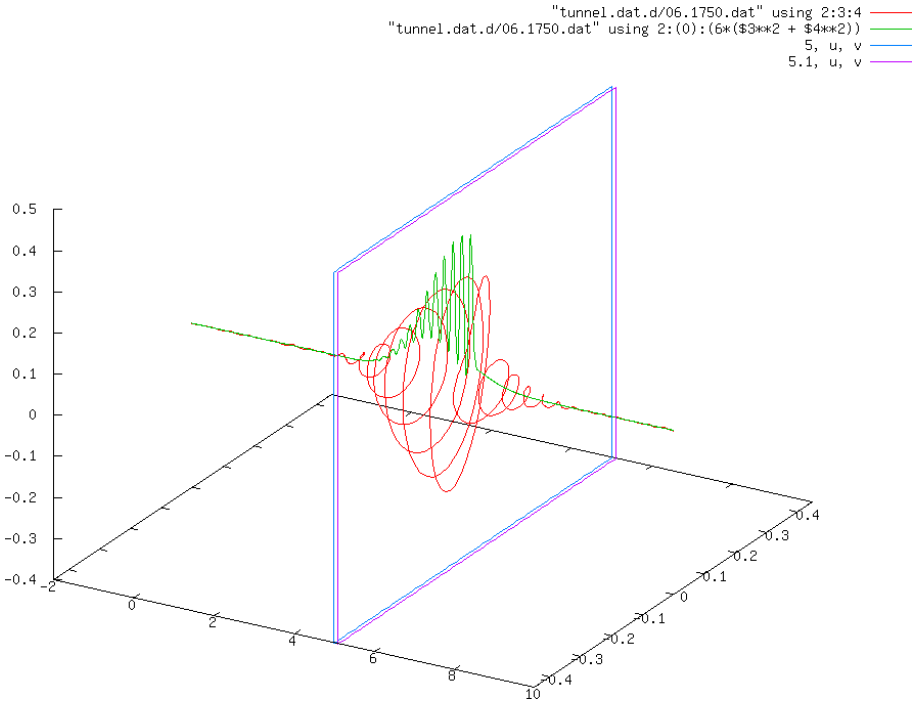


(b)

Figura 2.2: La particella si dirige verso la barriera.

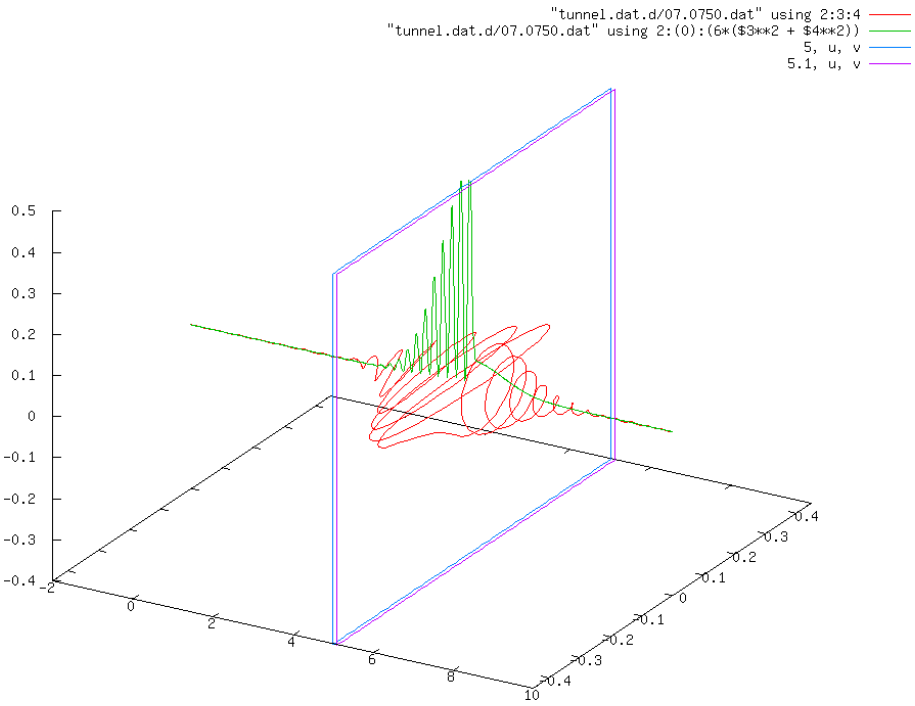


(a)

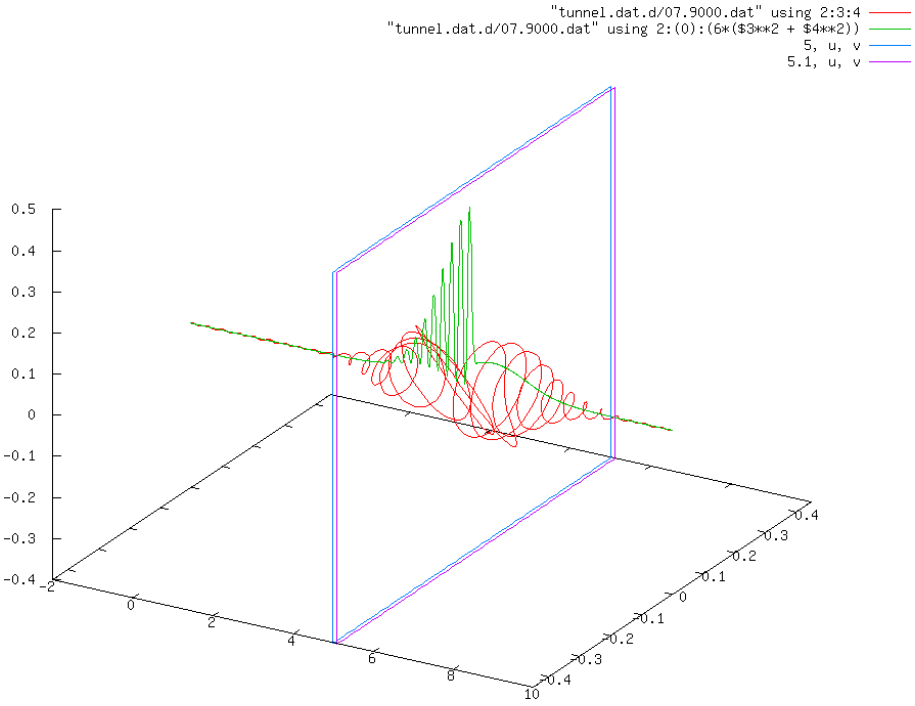


(b)

Figura 2.3: Impatto con la barriera.

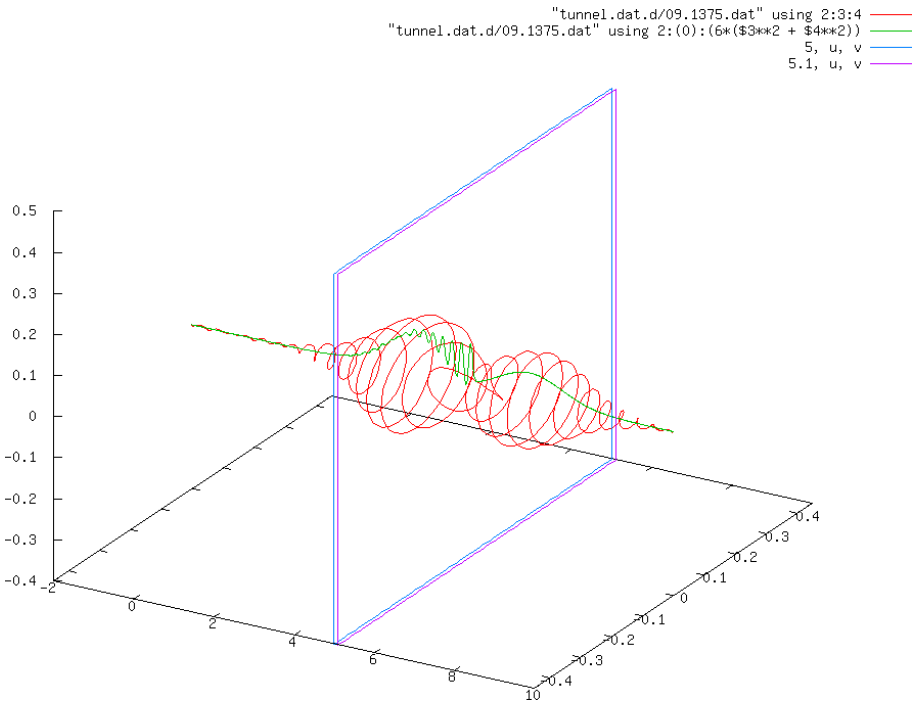


(a)

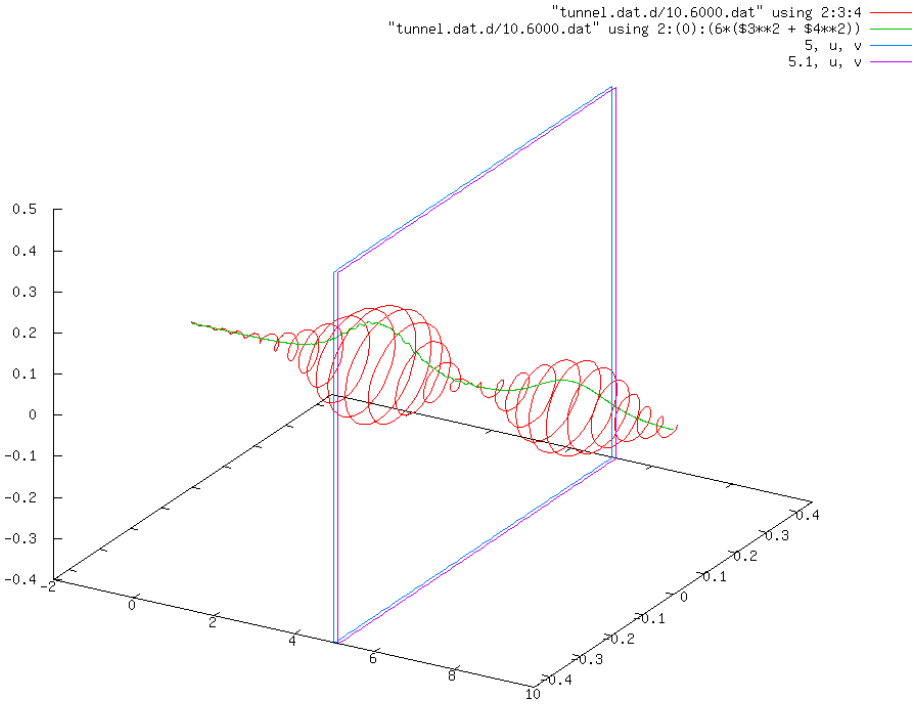


(b)

Figura 2.4: Effetto tunnel.



(a)



(b)

Figura 2.5: Vi è una probabilità che la particella “sia passata attraverso”.

Nello script in §A.1.1 è presente una sezione con i seguenti altri parametri, *commentati*:

```
/*
x0                  : 3.0;
x3                  : 6.5;
ti                  : 5;
tf                  : 12;
tstep               : 0.007;
...
*/
```

che ci consentirebbero, se sostituiti ai parametri cui ci siamo riferiti sinora, di ottenere una animazione molto più dettagliata, nel tempo, di quanto accade in prossimità della barriera.

Tutte le animazioni, i grafici, i file di dati, il codice *Perl* e *Maxima* sono disponibili alla pagina web `http://people.na.infn.it/~pq-qp/pages/simulations.html`, parte del *portale* didattico "$PQ - QP$"[3].

2.3 Buca di potenziale bidimensionale a pareti infinite

Il calcolo delle funzioni d'onda è, in questo caso, abbastanza semplice da non richiedere l'aiuto di metodi numerici o di strumenti informatici[3], dei quali ci serviremo solo per la rappresentazione grafica.

[3] Ciò non toglie che, anche nell'affrontare questo problema, *Maxima*[5] sia stato un utile strumento di verifica, ad esempio per controllare le condizioni di normalizzazione.

2.3.1 Combinazioni lineari finite

Sceglieremo ora alcune combinazioni lineari, opportunamente normalizzate, delle funzioni (1.15):

$$\sqrt{\frac{1}{5}}\Psi_{11} + \sqrt{\frac{2}{5}}\Psi_{21} + \sqrt{\frac{2}{5}}\Psi_{12}, \tag{2.2a}$$

$$\sqrt{\frac{1}{5}}\Psi_{11} + \sqrt{\frac{2}{5}}\Psi_{21} + i\sqrt{\frac{2}{5}}\Psi_{12}, \tag{2.2b}$$

$$\sqrt{\frac{1}{10}}\Psi_{11} + \sqrt{\frac{3}{10}}\Psi_{21} + i\sqrt{\frac{3}{10}}\Psi_{12} + \sqrt{\frac{3}{20}}\Psi_{32} + i\sqrt{\frac{3}{20}}\Psi_{23}, \tag{2.2c}$$

$$\sqrt{\frac{3}{10}}\Psi_{11} + \sqrt{\frac{3}{10}}\Psi_{22} + \sqrt{\frac{1}{5}}\Psi_{52} + i\sqrt{\frac{1}{5}}\Psi_{25}. \tag{2.2d}$$

Ci occorrerà salvare in un file ciascuna di queste definizioni. Ad esempio, l'ultima delle (2.2), tenuto conto della (1.15) e della (1.14), nella sintassi di *Gnuplot*[4] diventa:

```
set zrange[0:.80]

psi(x,y)=(2./pi)*(\
     cos(1*x)*cos(1*y)*exp(- 2*i*t)          *sqrt(.30)+\
     sin(2*x)*sin(2*y)*exp(- 8*i*t)          *sqrt(.30)+\
     cos(5*x)*sin(2*y)*exp(-29*i*t)          *sqrt(.20)+\
     sin(2*x)*cos(5*y)*exp(-29*i*t) *i       *sqrt(.20)\
)
```

2.3.2 Simulazioni

Lo script in §A.2.1 accetta come argomento il nome o il percorso di un file del tipo appena descritto e riportato in §A.2.4. Questo file contiene la definizione della funzione d'onda e l'impostazione del *range* di valori per la rappresentazione grafica.

Lo script crea una directory nella quale salverà le immagini prodotte; carica un "preambolo" di impostazioni per *Gnuplot* (si veda §A.2.2); dopo di che itera il disegno dei grafici su un insieme discreto di valori del tempo t.

```
for($t=$t_i;$t<=$t_f;$t+=$step) {
     $n=sprintf("%07.4f",$t);
     $progress=sprintf("%02d", (($t-$t_i)/($t_f-$t_i))*100 );
```

```
    print STDERR "plotting @ t=$n  $progress\%     \r";
    print GNUPLOT "
        set title \"t=$n\" offset 0,-1
        t=$t
        set output \"$dir/frame.$n.$format\"
        load \"$plot\"
    ";
}
```

Nel precedente frammento di codice, `$plot` è il nome di un ulteriore file (v. §A.2.3) contenente le istruzioni con cui *Gnuplot* effettua il disegno vero e proprio.

```
# pseudodata special file '++'
# requires Gnuplot 4.3 or higher (currently under development,
# available via CVS only -- @ 2008-06-25)
#
# http://gnuplot.sourceforge.net/demo_4.3/heatmaps.3.png
#
# http://sourceforge.net/tracker/index.php?func=detail&
# aid=1872528&group_id=2055&atid=302055
#

splot '++' using 1:2:(abs(psi($1,$2))**2):(arg(psi($1,$2))) w pm3d at s
```

L'ultima istruzione, grazie ad una recente caratteristica, consente l'uso diretto del colore per rappresentare l'argomento complesso della funzione d'onda.[4]

Con le impostazioni scelte, al termine dell'esecuzione troviamo nella directory creata oltre 1500 immagini, sufficienti a produrre un'animazione [5] della durata di circa un minuto allo standard di 25 frame/s.

2.3.3 Immagini ed esempî

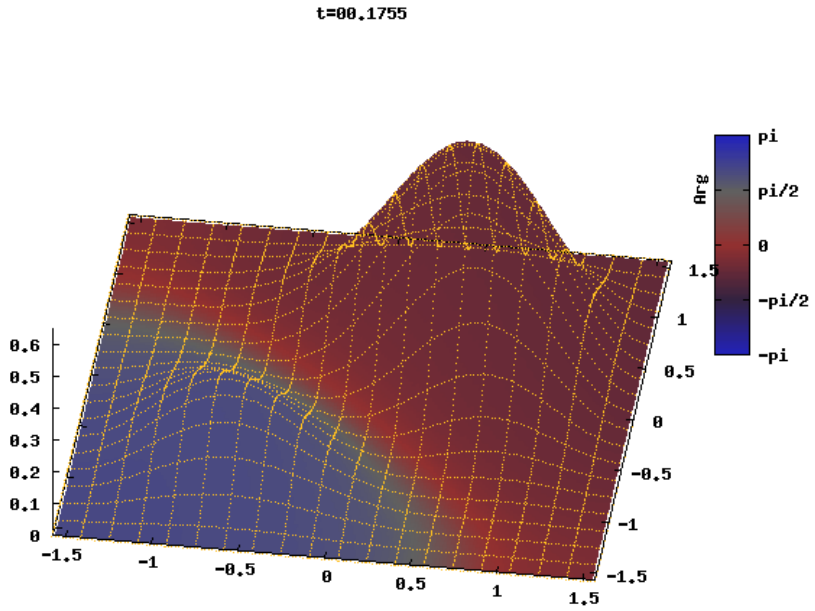
Il comportamento della (2.2a) in Figura 2.6 ricorda quello di una particella che "va avanti e indietro" lungo la diagonale del quadrato $[-\frac{\pi}{2}, \frac{\pi}{2}]^2$.

La (2.2b), rappresentata in Figura 2.7, è diversa dalla (2.2a) solo per un fattore i (unità immaginaria) nel coefficiente relativo a Ψ_{12}, ma il comportamento è molto diverso: la particella sembra "ruotare" attorno al centro della buca.

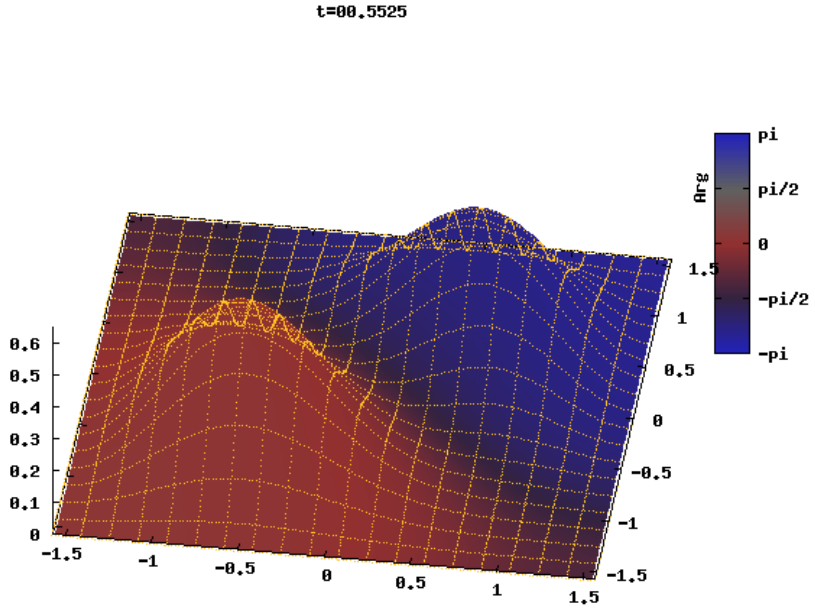
La (2.2c) in Figura 2.8 mostra un comportamento misto.

[4] Nelle precedenti versioni di *Gnuplot* sarebbe stato necessario creare un *file di dati* con un programma esterno.

[5] Per convertire una sequenza di immagini in un file video si possono usare strumenti quali *MEncoder*/*MPlayer* `http://www.mplayerhq.hu/`.

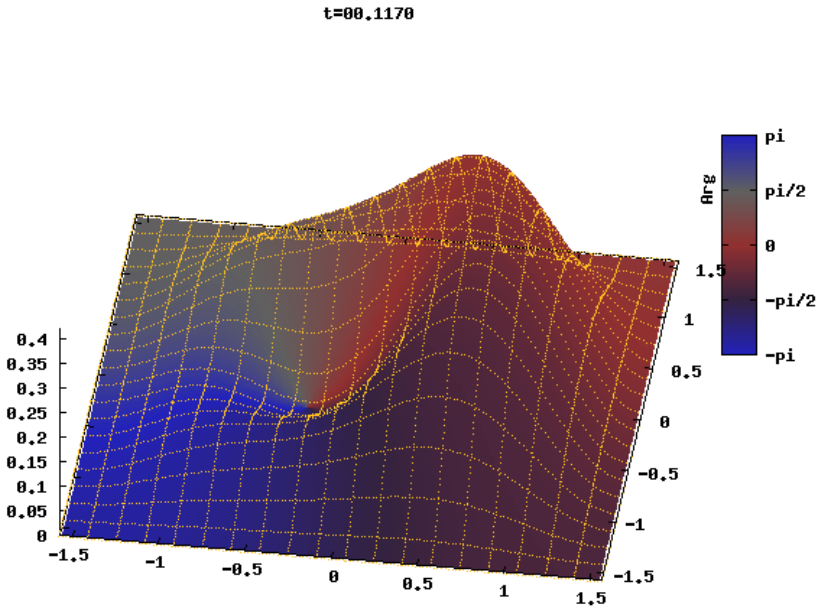


(a)

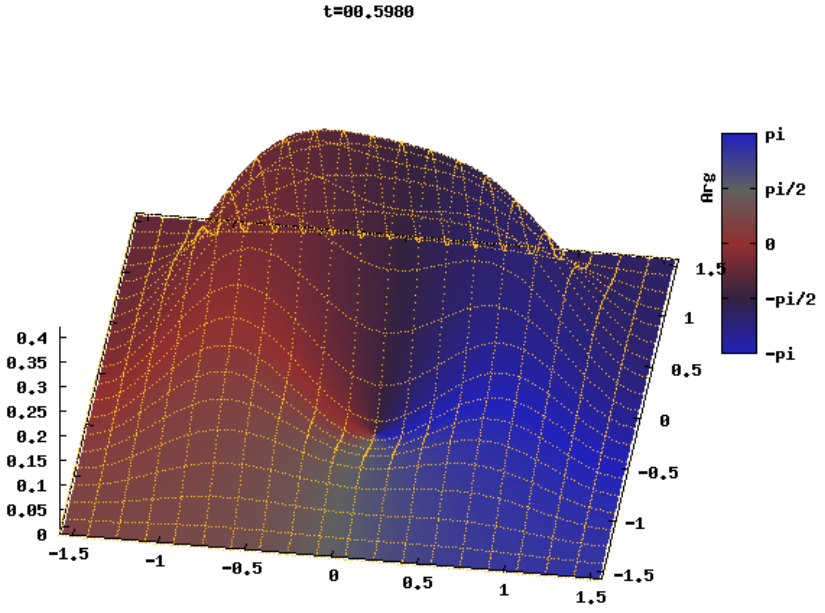


(b)

Figura 2.6: Buca 2D: $\sqrt{\frac{1}{5}}\Psi_{11} + \sqrt{\frac{2}{5}}\Psi_{21} + \sqrt{\frac{2}{5}}\Psi_{12}$.



(a)



(b)

Figura 2.7: Buca 2D: $\sqrt{\frac{1}{5}}\Psi_{11} + \sqrt{\frac{2}{5}}\Psi_{21} + i\sqrt{\frac{2}{5}}\Psi_{12}$.

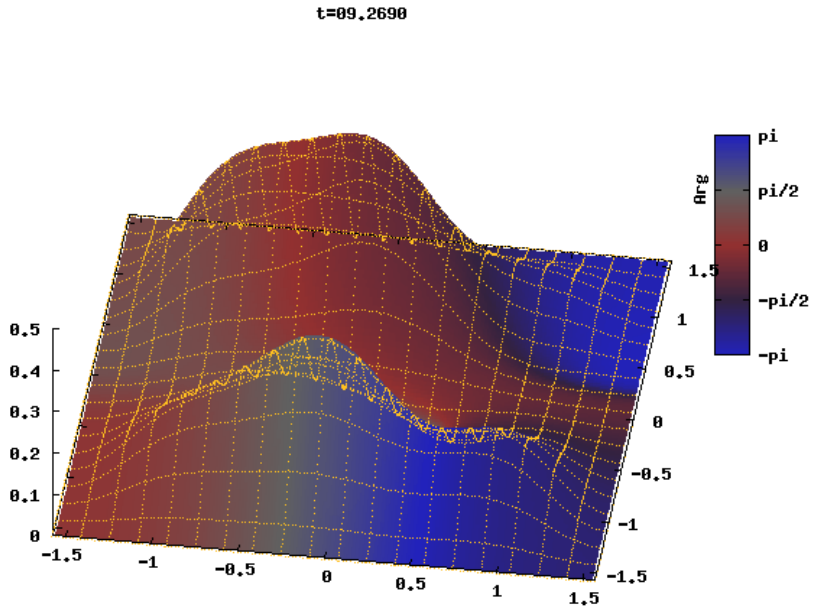


Figura 2.8: Buca 2D: $\sqrt{\frac{1}{10}}\Psi_{11} + \sqrt{\frac{3}{10}}\Psi_{21} + i\sqrt{\frac{3}{10}}\Psi_{12} + \sqrt{\frac{3}{20}}\Psi_{32} + i\sqrt{\frac{3}{20}}\Psi_{23}$.

La (2.2d), rappresentata in Figura 2.9, esibisce invece proprietà squisitamente ondulatorie.

Anche per queste simulazioni, tutto il materiale multimediale, il codice sorgente ed eventuale documentazione aggiuntiva sono disponibili su [3], alla pagina delle *Simulazioni*.

2.3.4 Possibili sviluppi

Combinando un piccolo numero di autofunzioni è difficile ottenere qualcosa che ricordi un comportamento classico. Poiché il raffronto fra il comportamento ondulatorio e quello corpuscolare può essere molto istruttivo, potremmo immaginare di invertire l'approccio al problema: considerare cioè un pacchetto d'onda ben localizzato in un arbitrario punto della buca con le opportune proprietà, e in seguito calcolarne i coefficienti di Fourier rispetto alla base (1.13), magari con l'ausilio di un programma per il calcolo. A questo punto si potrà simularne l'evoluzione temporale in modo analogo (a meno di accorgimenti tecnici) a quanto fatto per le (2.2).

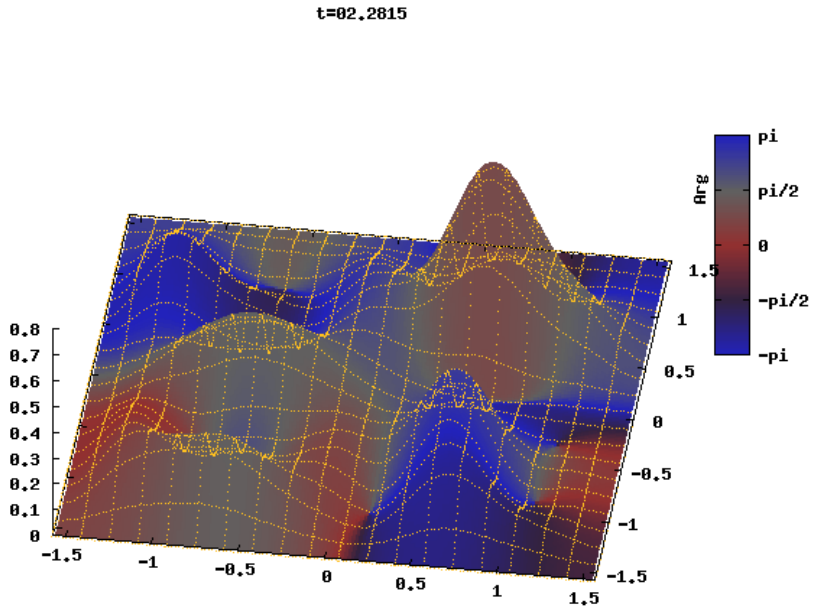


Figura 2.9: Buca 2D: $\sqrt{\frac{3}{10}}\Psi_{11} + \sqrt{\frac{3}{10}}\Psi_{22} + \sqrt{\frac{1}{5}}\Psi_{52} + i\sqrt{\frac{1}{5}}\Psi_{25}$.

Capitolo 3

Conclusioni

Abbiamo mostrato come sia possibile superare, con opportuni accorgimenti, le difficoltà nella rappresentazione grafica delle funzioni a valori complessi; rappresentazione importantissima in Meccanica Quantistica giacché, come è noto, la funzione d'onda *deve essere complessa* se si vuole che l'equazione di Schrödinger — equazione differenziale del primo ordine rispetto al tempo — abbia soluzioni ondulatorie [6]. Solo gli autostati dell'energia sono descrivibili da funzioni reali: sono stati stazionarî e la loro evoluzione temporale è banale in quanto l'argomento complesso cambia globalmente con un'unica frequenza $\omega = E/\hbar$.

Più complicata, per certi versi più interessante, e necessariamente descritta da funzioni complesse, è invece l'evoluzione di stati non stazionarî, che abbiamo simulato considerando sovrapposizioni discrete o continue di autofunzioni relative a energie diverse. L'evoluzione di tali stati rende evidenti sia la propagazione ondosa che le "traiettorie" di pacchetti localizzati.

Il presente lavoro è suscettibile ovviamente di sviluppi ulteriori, come si è già visto in §2.3.4.

L'evoluzione più naturale è la simulazione di sistemi a tre dimensioni, disegnando ad esempio le superfici di livello di $|\Psi|^2$ o le linee di forza della corrente di probabilità $\mathbf{j} = \frac{\hbar}{2mi}\left(\Psi^*\nabla\Psi - \Psi\nabla\Psi^*\right)$, mentre l'argomento complesso della Ψ può essere rappresentato con una tavolozza di colori in modo analogo a quanto

visto per la buca bidimensionale.

Vale la pena di osservare, infine, che proprio l'uso del colore costituisce un problema di accessibilità per le persone con limitazioni visive, il che può motivarci a sviluppare rappresentazioni alternative oltre che ad approfondire la vasta problematica legata alla comunicazione dei risultati scientifici.

Ringraziamenti

Grazie ai miei relatori, per il dettaglio e per la visione d'insieme. È stata un'esperienza istruttiva e divertente.

Grazie ai ragazzi che sotto la guida di Ofelia Pisanti e Fedele Lizzi hanno lavorato a *PQ-QP* [3]. In modo particolare Deborah Pallotti e Mariano Barbieri che hanno aperto la strada alle simulazioni descritte in queste pagine. A Mariano devo soprattutto l'opera paziente di *fine tuning* relativa alla scelta ottimale dei parametri per l'effetto tunnel.

Grazie a questa bella comunità di fisici napoletani; a chi ha condiviso l'entusiasmo degli inizî e a chi questo entusiasmo, d'un tratto, l'ha visto e l'ha fatto rinascere.

Grazie in modo speciale ai miei genitori, per l'amore e la pazienza infiniti; e a tutta la mia famiglia per la passione con cui mi segue.

Grazie agli amici, in senso lato e in senso stretto, dentro e fuori l'Università. Grazie a Luigi, Rossana, Rosario, Tiziano, Chiara, Massimo, Sergio, Vittorio, Michele, Ciro, Luca, Gianluca, Antonello, Serena, Lello, Ettore, Francesco, Antonio, Cristina, Felicia, Carmine, Anna, Diego, Patrizia, Laura, Nunzia, Tiziana, Giampaolo, Nello, Emanuele, Igor, Antonella, Mario, Guido, Giuseppe, Paolo, Sandra, Claudia, Nico, Daniela, Simona, Sabino, Roberto, Antonia, Maria Antonietta, Orazio, Marco; ciascuno per un motivo diverso. Grazie agli omonimi, che mi aiutano ad accorciare la lista e soprattutto grazie a tutti coloro di cui mi sono dimenticato.

Riferimenti bibliografici e telematici

[1] P. Caldirola, R. Cirelli, e G. M. Prosperi. *Introduzione alla Fisica Teorica*, capitolo VI.A.3. UTET, Torino, 2000.

[2] P. Caldirola, R. Cirelli, e G. M. Prosperi. *Introduzione alla Fisica Teorica*, capitolo VIII. UTET, Torino, 2000.

[3] Corso di Laurea in Fisica; Università degli Studi di Napoli "Federico II". Portale didattico PQ-QP. `http://people.na.infn.it/~pq-qp/`.

[4] Gnuplot. `http://www.gnuplot.info/`.

[5] Maxima. `http://maxima.sourceforge.net/`.

[6] E. Merzbacher. *Quantum Mechanics*, capitolo 2.1. John Wiley & Sons, New York, 1961.

[7] J. Von Neumann. *Mathematical Foundations of Quantum Mechanics*. Princeton University Press, 1996.

[8] The Open Source Initiative. `http://www.opensource.org/`.

[9] wxMaxima. `http://wxmaxima.sourceforge.net/`.

Appendice A

Codice

A.1 Barriera di potenziale in una dimensione ed effetto tunnel

A.1.1 `tunnel.mxm`: codice *Maxima* per il calcolo numerico e simbolico

```
/* A symbolic+numeric simulation of the tunnel quantum effect */
/* Copyright (C) 2007 Guido De Rosa */
/* License: MIT */

globalsolve : true;
assume(E>0);
assume(m>0);
assume(E<V);

/* general solutions for potential barrier */
psi1(x) := exp(%i*x*sqrt(2*m*E)/hbar) + A2*exp(-%i*x*sqrt(2*m*E)/hbar);
psi2(x) := C1*exp(x*sqrt((V-E)*m)/hbar) + C2*exp(-x*sqrt((V-E)*m)/hbar);
psi3(x) := B1*exp(%i*x*sqrt(2*m*E)/hbar);

/* and their derivatives */
Dpsi1(x) := diff(psi1(x),x);
Dpsi2(x) := diff(psi2(x),x);
Dpsi3(x) := diff(psi3(x),x);

/* regularity condition on wavefunction and derivative */
reg12 : psi1(x1)=psi2(x1);
reg23 : psi2(x2)=psi3(x2);
Dreg12 : Dpsi1(x1)=Dpsi2(x1);
Dreg23 : Dpsi2(x2)=Dpsi3(x2);

/* Simbolically solve the liner equations in A2,C1,C2,B1*/
linsolve([reg12,reg23,Dreg12,Dreg23],[A2,C1,C2,B1]);

psi1(x);
psi2(x);
psi3(x);

```

```
33 /* time evolution of each eigenfunction */
34 Epsi1(x,E,t) := psi1(x)*exp(-%i*E*t/hbar);
35 Epsi2(x,E,t) := psi2(x)*exp(-%i*E*t/hbar);
36 Epsi3(x,E,t) := psi3(x)*exp(-%i*E*t/hbar);
37
38 /* set the parameters of our problem */
39 m       : 2;
40 hbar    : 0.1;
41 V       : 1;
42 x1      : 5.0;
43 x2      : 5.1;
44 E0      : 0.5;
45 sigmaE  : 0.1;
46 deltaE  : 1.5 * sigmaE;
47
48 /* a generalized 'Fourier Transform' to make a superposition of
49  * eigenfunctions by integrating f(E)*Epsi[1|2|3](x,E,t) over an interval
50  * from E = E0-deltaE to E = E0+deltaE . A guaussian is a good choice
51  * to obtaina a well localized packet
52  */
53 f(E) := exp(-(E-E0)^2/sigmaE^2);
54
55 assume(E<V);
56
57 /* boundaries for plotting ... */
58 x0                : -2;
59 x3                : 10;
60 xstep             : 0.0141;
61 epsrel            : 0.2;  /* max relative error for numeric integration
62                              *(quadpack): in practice, very rarely error
63                              * is more than 5-10%; accuracy is good enough
64                              * and we don't need to slowdown calculation
65                              * any more...
66                              */
67 limitq            : 5;    /* max number of subintervals for numeric
68                             /*integration (quadpack) [ignored?] */
69 ti                : 1.30;
70 tf                : 14.8;
71 tstep             : 0.03750;
72 Emin              : E0 - deltaE;
73 Emax              : E0 + deltaE;
74 prefix            : "tunnel-"; /* output filename prefix */
75
76 /* For detailed pictures near the barrier, I used these settings instead: */
77 /*
78 x0                : 3.0;
79 x3                : 6.5;
80 ti                : 5;
81 tf                : 12;
82 tstep             : 0.007;
83 prefix            : "barrier-";
84 */
85
86 /* write data to a file... to be read by a Perl script which then prepares
87  * chunks of data and commands for Gnuplot
88  * TODO: resume from a previously interrupted calculation.
89  *
90  * Instead, choose filename with a timestamp to avoid unwanted overwitings
91  */
92 output_file : sconc(prefix,string(absolute_real_time()),".dat");
93 stream: openw(output_file);
94 format: "~f ~f ~f ~f~%"; /* float */
```

```
95
96  for t:ti step tstep thru tf do (
97
98    printf(stream,"~%"),
99
100   /* The superposition (wavepacket) is constructed  */
101   for x: x0 step xstep thru x1 do (
102     result_re : quad_qags(  /*numerically integrate real... */
103       realpart(f(E)*Epsi1(x,E,t)),
104       E,
105       Emin,
106       Emax,
107       epsrel,
108       limitq
109     ),
110     result_im : quad_qags( /*...and imaginary part*/
111       imagpart(f(E)*Epsi1(x,E,t)),
112       E,
113       Emin,
114       Emax,
115       epsrel,
116       limitq
117     ),
118     print(x, "= x < x1; t =", t, " results:", result_re, result_im),
119     printf(stream,format,t,x,result_re[1],result_im[1])
120   ),
121   for x: x1 step xstep thru x2 do (
122     result_re : quad_qags(
123       realpart(f(E)*Epsi2(x,E,t)),
124       E,
125       Emin,
126       Emax,
127       epsrel,
128       limitq
129     ),
130     result_im : quad_qags(
131       imagpart(f(E)*Epsi2(x,E,t)),
132       E,
133       Emin,
134       Emax,
135       epsrel,
136       limitq
137     ),
138     print(x, "= x in (x1,x2); t =",t," results:", result_re, result_im),
139     printf(stream,format,t,x,result_re[1],result_im[1])
140   ),
141   for x: x2 step xstep thru x3 do (
142     result_re : quad_qags(
143       realpart(f(E)*Epsi3(x,E,t)),
144       E,
145       Emin,
146       Emax,
147       epsrel,
148       limitq
149     ),
150     result_im : quad_qags(
151       imagpart(f(E)*Epsi3(x,E,t)),
152       E,
153       Emin,
154       Emax,
155       epsrel,
156       limitq
```

```
157     ),
158     print(x, " = x > x2; t=",t," results:", result_re, result_im),
159     printf(stream,format,t,x,result_re[1],result_im[1])
160   )
161
162 );
```

A.1.2 tunnel-plot.pl: script per la rappresentazione grafica e le animazioni

```
1  #!/usr/bin/perl -w
2
3  # Copyright (C) 2007, Guido De Rosa <guido_derosa@libero.it>
4  # License: MIT
5
6  # Split one big (Maxima->Gnuplot) data file by "time"
7  # i.e.: one file, one plot, one frame for animation...
8
9  #use Data::Dumper;
10 use File::Basename;
11 use IO::Handle;
12
13 use strict;
14
15 # Configuration
16 my $gnuplot     =   "gnuplot"; # or full path to the command
17 my ($x1,$x2)    =   (5, 5.1); # barrier
18 my $xrange      =   "[-2:10]";
19 my $yrange      =   "[-.45:.45]";
20 my $zrange      =   "[-.4:.5]";
21 my $smscale     =    6; # scaling factor plotting |psi(x)|^2 on z axis
22
23 autoflush STDOUT 1;
24 autoflush STDERR 1;
25
26 my ($infile, $sizex, $sizey) = @ARGV or
27 die ("Usage: $0 {file.dat} [ size-X size-Y ]\n");
28
29 my ($name,$path,$ext) = fileparse($infile,".[a-zA-Z0-9]*");
30 $name =~ s/\.[^\.]+$//; # remove extension
31
32 my $datadir     =   "$infile.d";
33 my $imgdir;
34 if ($sizex and $sizey) {
35     $imgdir      =   "$datadir/img-$sizex"."x$sizey";
36 } else {
37     $imgdir      =   "$datadir/img";
38 }
39
40 my %handles = ();
41 my ($t, $n, $line, $fh, $file);
42
43 sub preamble ;
44 sub videncode ;
45 sub message ;
46
47 -d $datadir or mkdir $datadir;
48 -d $imgdir or mkdir $imgdir;
49
50 open(FH,"<$infile");
```

```
51
52  while(<FH>) {
53      chomp;
54      if (m/^\s*(\d*\.\d+)/) {
55          $line = $_;
56          $t = $1;
57          $n = sprintf("%07.4f",$t);
58          $file = "$datadir/$n$ext";
59          unless ($handles{"$n"}) {
60              message("splitting: $n$ext ... \r");
61              open($handles{"$n"},">$file") or die ("Couldn't open $file:
62  $!\n");
63              #print Dumper(\%handles),"\n";
64          }
65          $fh = $handles{"$n"};
66          print $fh "$line\n";
67      }
68  }
69
70  message("\n");
71
72  open(GNUPLOT,"|$gnuplot");
73
74  autoflush GNUPLOT 1;
75
76  preamble();
77
78  foreach $n (sort(keys(%handles))) {
79      $fh = $handles{$n};
80      close($fh);
81      message("plotting: $n.png ... \r");
82      print GNUPLOT "set output \"$imgdir/$n.png\"\n";
83      print GNUPLOT "splot \"$datadir/$n$ext\" using 2:3:4 w lines, ".
84      "\"$datadir/$n$ext\" using 2:(0):($smscale*(\$3**2 + \$4**2)) w lin
85  es,".
86      " $x1, u, v w lines, $x2, u, v w lines\n";
87  }
88
89  close(GNUPLOT);
90
91  videncode();
92
93  sub preamble {
94      print GNUPLOT "reset\n";
95      if ($sizex and $sizey) {
96          print GNUPLOT "set terminal png crop small size $sizex, $sizey\n
97  ";
98      } else {
99          print GNUPLOT "set terminal png crop small\n";
100     }
101     print GNUPLOT <<EOF
102 set parametric
103 set ticslevel 0
104 set xrange$xrange
105 set yrange$yrange
106 set zrange$zrange
107 set urange$yrange
108 set vrange$zrange
109 set isosamples 2
110
111 EOF
112 }
```

```
113
114 sub videncode {
115     # You will need mplayer and ffmpeg2theora
116     # http://www.mplayerhq.hu/  http://www.v2v.cc/~j/ffmpeg2theora/
117
118     message("\nencoding␣videos...␣");
119
120     my $mencoder = "mencoder";
121     my $mplayer = "mplayer";
122     my $ffmpeg2theora = "ffmpeg2theora";
123     my $ffmpeg = "ffmpeg";
124     my $videodir;
125     my $mpverbosity = ""; # "-msglevel all=-1:avsync=5";
126
127     if ($sizex and $sizey) {
128         $videodir = "$datadir/video-$sizex"."x$sizey";
129     } else {
130         $videodir = "$datadir/video";
131     }
132
133     my $uri             = "mf://$imgdir/*.png";
134     my $lossless        = "$videodir/$name-lossless.avi";
135     my $ogghq           = "$videodir/$name-hq.ogg";
136     my $ogglq           = "$videodir/$name-lq.ogg";
137     my $mpeg            = "$videodir/$name.mpg";
138     my $mpeglq          = "$videodir/$name-lq.mpg";
139     my $mp4             = "$videodir/$name.mp4";
140     my $mp4lq           = "$videodir/$name-lq.mp4";
141     my $avi             = "$videodir/$name.avi";
142     my $flv             = "$videodir/$name.flv";
143
144     my $theoralq        = 56; # low bitrate for theora
145
146     mkdir "$videodir";
147
148     # encode an intermadiate, lossless file (better solution than
149     # writing to a VERY BIG yuv file, or using named pipes):
150     system("$mencoder␣$uri␣-o␣$lossless␣-ovc␣lavc␣-lavcopts␣vcodec=ffv1
151 $mpverbosity");
152
153     # encode OGG/Theora video
154     # 0different qualities
155     system("$ffmpeg2theora␣-o␣$ogghq␣-S␣0␣-v␣6␣--optimize␣$lossless");
156     system("$ffmpeg2theora␣-o␣$ogglq␣-S␣0␣-V␣$theoralq␣--optimize␣$lossl
157 ess");
158
159     # MPEG1 and MPEG4
160     system("$ffmpeg␣-y␣-i␣$lossless␣$mpeg");
161     system("$ffmpeg␣-y␣-i␣$lossless␣$mp4");
162     system("$ffmpeg␣-y␣-b␣1␣-i␣$lossless␣$mpeglq");
163     system("$ffmpeg␣-y␣-b␣1␣-i␣$lossless␣$mp4lq");
164
165     # Optional, proprietary formats/codecs:
166
167     # encode a "Microsoft-friendly" AVI/M$MPEG4v2 video
168     # http://lists.mplayerhq.hu/pipermail/mencoder-users/
169     # 2006-November/004562.html
170     system("$mencoder␣$uri␣-ovc␣lavc␣-lavcopts␣vcodec=msmpeg4v2:mbd=2:ke
171 yint=5:subcmp=2:dia=2:mv0:autoaspect␣-ofps␣24␣-o␣$avi␣$mpverbosity");
172
173     # encode a Macromedia Flash Video: never tested with a flash player
174
```

```
175     #system("$mencoder $uri -o $flv -ovc lavc -lavcopts vcodec=flv -of l
176 #avf -lavfopts format=flv:i_certify_that_my_video_stream_does_not_use_b_f
177 #rames");
178 
179 }
180 
181 sub message {
182     #print STDERR @_;
183     print @_;
184 }
```

A.2 Buca di potenziale bidimensionale

A.2.1 buca2d.anima.pl: script Perl

```
1  #!/usr/bin/perl -w
2  
3  use IO::Handle;
4  
5  autoflush STDOUT 1;
6  autoflush STDERR 1;
7  
8  # TODO: use strict; better style..
9  
10 $gnuplot="gnuplot"; # or complete path
11 
12 $preamble="buca2d.preamble.gpi";
13 $function=$ARGV[0];
14 $dir="$function.d";
15 $plot="buca2d.plot.gpi";
16 $format='png';
17 $t_i=0;
18 $t_f=10;
19 $step=0.0065;
20 
21 mkdir $dir;
22 
23 open(GNUPLOT,"|$gnuplot");
24 
25 autoflush GNUPLOT 1;
26 
27 print GNUPLOT "load␣\"$preamble\"\n";
28 print GNUPLOT "load␣\"$function\"\n";
29 
30 for($t=$t_i;$t<=$t_f;$t+=$step) {
31     $n=sprintf("%07.4f",$t);
32     $progress=sprintf("%02d", (($t-$t_i)/($t_f-$t_i))*100 );
33     print STDERR "plotting␣@␣t=$n␣␣$progress\%␣␣␣␣\r";
34     print GNUPLOT "
35 ␣␣␣␣␣␣␣␣set␣title␣\"t=$n\"␣offset␣0,-1
36 ␣␣␣␣␣␣␣␣t=$t
37 ␣␣␣␣␣␣␣␣set␣output␣\"$dir/frame.$n.$format\"
38 ␣␣␣␣␣␣␣␣load␣\"$plot\"
39 ␣␣␣␣";
40 }
41 print GNUPLOT "\n";
42 
43 close(GNUPLOT);
```

A.2.2 buca2d.preamble.gpi: impostazioni *Gnuplot*

```
reset
unset key
#set size square
set terminal png #crop
i={0,1}
#phi(x,y)=arg(x+i*y)
#r(x,y)=sqrt(x*x+y*y)

set palette defined (\
    -pi "#2222BB", -pi/2 "#332240", 0 "#923030", pi/2 "#606060", pi "#2222BB")
#set palette defined (\
#    -pi "#2222BB",                   0 "#923030", pi/2 "#707040", pi "#2222BB")

set ticslevel 0
set view 30,8,1.1,1.1
set cbrange[-pi:pi]
set xrange[-pi/2:pi/2]
set yrange[-pi/2:pi/2]

set cbtics ("-pi" -pi, "-pi/2" -pi/2, "0" 0, "pi/2" pi/2, "pi" pi)
set cblabel "Arg"  offset -12.5,3
```

A.2.3 buca2d.plot.gpi: codice *Gnuplot* per i grafici

```
set multiplot
set samples 150
set isosamples 150
       #set hidden3d
       set pm3d corners2color c1
       #set style line 100 lt 2 lw 0.5
       #unset hidden3d
       #set surf


# pseudodata special file '++'
# requires Gnuplot 4.3 or higher (currently under development,
# available via CVS only -- @ 2008-06-25)
#
# http://gnuplot.sourceforge.net/demo_4.3/heatmaps.3.png
# http://sourceforge.net/tracker/index.php?func=detail&
#aid=1872528&group_id=2055&atid=302055

splot '++' using 1:2:(abs(psi($1,$2))**2):(arg(psi($1,$2))) w pm3d at s

set isosamples 20
#set hidden3d
unset ztics
set style line 2 lw 0.1
splot abs(psi(x,y))**2 w dots lt 7
unset multiplot
set ztics
```

A.2.4 myfunc.gpi: esempio di definizione di funzione d'onda nella sintassi di *Gnuplot*

```

```

```
2  set zrange[0:.80]
3
4  psi(x,y)=(2./pi)*(\
5       cos(1*x)*cos(1*y)*exp(- 2*i*t)       *sqrt(.30)+\
6       sin(2*x)*sin(2*y)*exp(- 8*i*t)       *sqrt(.30)+\
7       cos(5*x)*sin(2*y)*exp(-29*i*t)       *sqrt(.20)+\
8       sin(2*x)*cos(5*y)*exp(-29*i*t) *i    *sqrt(.20)\
9  )
```