Relazione di Fisica Computazionale

Soluzione dell'equazione di Schrödinger unidimensionale per una doppia buca di potenziale

Alex Martinelli

Anthea Boiani

RIASSUNTO

Questo progetto si basa sullo studio, effettuato con due algoritmi diversi, di un potenziale a doppia buca. Prima di procedere con il secondo algoritmo che prevede l'utilizzo della trasformata di Fourier, questo è stato testato su un caso noto: il potenziale armonico, di cui già si conoscono le soluzioni.

INTRODUZIONE

L'equazione di Schrödinger indipendente dal tempo fornisce gli stati stazionari di una particella quantistica di massa m. Il problema in esame si focalizza sulla ricerca degli autovalori e autofunzioni di tale equazione tramite diagonalizzazione di una opportuna matrice che compare nella forma matriciale dell'equazione di Schrödinger, con la condizione che il potenziale utilizzato sia del tipo a doppia buca. Per fare questo verrranno utilizzate due diverse rappresentazioni dell'equazione: la rappresentazione in spazio reale e la rappresentazione sulla base delle onde piane.

DESCRIZIONE DEL PROBLEMA E DEGLI ALGORITMI UTILIZZATI

Il problema viene affrontato nel caso unidimensionale, prendendo come asse di riferimento l'asse x:

$$\frac{1}{2}\left[-\frac{\partial^2}{\partial x^2} + V(x)\right]\phi(x) = E\phi(x).$$

Dove il potenziale risulta nella forma:

$$V(x) = a + bx + cx^2 + dx^4$$

- a rappresenta una traslazione verticale del potenziale.
- $b = 0$ rende le due buche del potenziale simmetriche rispetto all'origine e di tipo armonico per valori di x vicini ai minimi, mentre $b \neq 0$ rompe la loro simmetria.
- $c > 0$, $c < 0$ fa sì che la parabola sia rivolta rispettivamente verso l'alto o verso il basso.
- $d > 0$ permette al potenziale di tendere all'infinito sia per $x \to \infty$ sia per $x \to -\infty$, $d < 0$ fa tendere il potenziale a $-\infty$ sia per $x \to \infty$ sia per $x \to -\infty$.

In questo esperimento sono stati scelti $c < 0, d > 0$ e successivamente confrontati i casi con $b = 0$ e $b \neq 0$.
É anche richiesto che i parametri del potenziale siano scelti in modo tale che esistano almeno due stati ben localizzati nelle rispettive buche.

TASK A

Considerando un intervallo di lunghezza L, costituito da N punti x_i equispaziati di un valore h, è possibile ricorrere all'espressione alle differenze finite:

$$f''(x) = \frac{f(x-h) - 2f(x) + f(x+h)}{h^2}$$

L'espressione finale dell'equazione descritta su una griglia di punti x_k è:

$$-\frac{\phi(x_{k-1}) - 2\phi(x_k) + \phi(x_{k+1})}{h^2} + V(x_k)\phi(x_k) = 2E\phi(x_k).$$

Questa equazione rappresenta un sistema lineare di N equazioni in N incognite, la k-esima equazione (dove $k = 1, 2, \ldots, N$) può essere riscritta nel modo seguente:

$$-\frac{\phi(x_{k-1})}{h^2} + \left(\frac{2}{h^2} + V(x_k)\right)\phi(x_k) - \frac{\phi(x_{k+1})}{h^2} = 2E\phi(x_k).$$

La forma matriciale di questa equazione permette di definire la matrice H.

$$H\phi = 2E\phi$$

La diagonalizzazione della matrice H fornisce gli autovalori E_n ed i rispettivi autovettori $\phi_n(x_k)$, riconducibili alle autofunzioni grazie alla griglia di punti:

$$[H]_{ij} = \begin{pmatrix} \frac{1}{h^2} + \frac{V_1}{2} & -\frac{1}{2h^2} & 0 & \cdots & \cdots & 0 \\ -\frac{1}{2h^2} & \frac{1}{h^2} + \frac{V_2}{2} & -\frac{1}{2h^2} & \ddots & \ddots & \vdots \\ 0 & -\frac{1}{2h^2} & \frac{1}{h^2} + \frac{V_3}{2} & -\frac{1}{2h^2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & -\frac{1}{2h^2} & \frac{1}{h^2} + \frac{V_{N-1}}{2} & -\frac{1}{2h^2} \\ 0 & \cdots & \cdots & 0 & -\frac{1}{2h^2} & \frac{1}{h^2} + \frac{V_N}{2} \end{pmatrix}$$

Questa è una matrice tridiagonale, simmetrica e reale, a causa del fattore moltiplicativo associato all'energia, gli autovalori risultanti saranno il doppio di quelli reali. La diagonalizzazione di H, così come il calcolo degli autovalori e delle autofunzioni, è svolta dalla subroutine `dstevx` presente in una delle librerie `LAPACK`, adatta in questo a caso al tipo di matrice: tridiagonale, simmetrica e reale.
`LAPACK` (Linear Algebra Package) è un insieme di librerie software usate per calcoli scientifici come la diagonalizzazione di matrici di diverso tipo. Ciascuna libreria contiene al proprio interno varie subroutine, ciascuna delle quali tratta uno specifico tipo di matrice. La subroutine `dstevx` richiede che vengano definiti specifici parametri per poter funzionare, l'elenco e il significato di questi è stato riportanto in `Appenice A`
Per compilare il programma principale è necessario usare il compilatore Intel eseguendo i comandi:

```
ifort -check all -o taskA.exe taskA.f90 -qmkl
./taskA.exe
```

La scelta del compilatore Intel è resa necessaria a causa del parametro `ABSTOL`, vi è la necessità di includere `mkl_lapack.fi` per permettere al compilatore di usare la funzione `DLAMCH` utilizzata per definire il parametro `ABSTOL` (che altrimenti non verrebbe trovata), affinché possa migliorare la precisione degli autovalori calcolati dalla subroutine, anche se questo implicherà un tempo di compilazione più alto.

OBBIETTIVI

- Determinare i primi M autovalori di energia piú bassa e le relative autofunzioni, confrontando un caso con $b = 0$ ed uno con $b \neq 0$.

- Confrontare gli stati localizzati nelle buche con quelli trovati nel caso di un potenziale armonico.

CODICE PER LA TASK A

File di input

Per il funzionamento del programma principale, viene utilizzato come file di input un documento di testo che verrà letto durante l'esecuzione del programma stesso. Qui viene riportata la struttura del suddetto file:

```
A
B
N
IL
IU
a
b
c
d
```

Listing 1: -inputA.txt

Dove:

1. l'estremo sinistro dell'intervallo L (`A`)
2. l'estremo destro dell'intervallo L (`B`)
3. il numero di punti da utilizzare nella griglia uniforme dell'intervallo (`N`)
4. l'indice del primo autovalore che il programma deve cercare (`IL`)
5. l'indice dell'ultimo autovalore da determinare (`IU`)
6. termine noto del potenziale (`a`)
7. coefficiente del termine x
8. coefficiente del termine x^2
9. coefficiente del termine x^4

Main Program

Il programma principale è noto come `taskA.f90`, scritto in linguaggio Fortran. Questo legge i dati presenti nel file di input e fornisce alcuni file in output. Il programma viene compilato attraverso il compilatore Intel, generando per convenzione l'eseguibile `taskA.exe`. Si suddivide in 4 parti principali:

1. lettura dei dati dal file `inputA.txt` ;
2. definizione dei vettori da utilizzare nella subroutine `dstevx`;
3. utilizzo della subroutine per il calcolo di autovalori e autovettori;

4. scrittura dei file di output.

L'intero codice è stato riportato in appendice al documento. Si procede con la discussione di alcune parti principali a partire dalla sezione dichiarativa.

```
1
PROGRAM taskA
3 IMPLICIT NONE
INCLUDE 'mkl_lapack.fi'
5 !
DOUBLE PRECISION:: A,B,a1,b1,c1,d1
7 INTEGER :: N
INTEGER :: K,I,J,T,C,Q,R
9 DOUBLE PRECISION :: L,h,S
DOUBLE PRECISION,DIMENSION(:),ALLOCATABLE :: D,X
11 DOUBLE PRECISION,DIMENSION(:),ALLOCATABLE :: E
DOUBLE PRECISION,DIMENSION(:),ALLOCATABLE :: V,v1
13 !
CHARACTER,PARAMETER :: JOBZ = 'V',RANGE = 'I'
15 DOUBLE PRECISION :: VL = 0,VU = 20
INTEGER :: IL,IU,UB
17 DOUBLE PRECISION :: ABSTOL
INTEGER :: M, INFO, LDZ
19 DOUBLE PRECISION,DIMENSION(:),ALLOCATABLE :: W
DOUBLE PRECISION,DIMENSION(:,:),ALLOCATABLE :: Z
21 DOUBLE PRECISION,DIMENSION(:),ALLOCATABLE :: WORK
INTEGER,DIMENSION(:),ALLOCATABLE :: IWORK
23 INTEGER,DIMENSION(:),ALLOCATABLE :: IFAIL
CHARACTER(LEN=*), PARAMETER :: inputA = 'inputA.txt'
25 OPEN(UNIT=7, FILE=inputA)
READ(7,*) A,B,N,IL,IU,a1,b1,c1,d1
27 CLOSE(7)
```

La seguente parte del codice è invece dedicata alla generazione dei vettori e della griglia che verranno utilizzati dalla subroutine dstevx. In particolare il vettore `D` rappresenta la diagonale principale della matrice H mentre `E` rappensenta la diagonale secondaria superiore e inferiore di H.

```
1 ALLOCATE(D(1:N))
ALLOCATE(X(1:N))
3 ALLOCATE(E(1:N-1))
!
5 L = B-A          !larghezza della doppia buca di potenziale
h = L/dble(N-1)          !distanza tra i punti della griglia
7 S = A
X(1) = A
9 X(N) = B
DO K=2,N-1          !creazione della griglia
11     S = S+h
       X(K) = S
13 END DO
```

```
DO I=1,N          !creazione della diagonale principale della matrice
    D(I) = 1/h**2+(a1 + b1*X(I) + c1*(X(I)**2) + d1*(X(I)**4))/2
END DO
    E = (/(-dble(0.5)/h**2,J=1,N-1)/)          !diagonale secondaria della matrice
```

File di output

I file che verranno generati dall'esecuzione del programma sono:

1. `autovalori.out`: contiene gli autovalori calcolati dalla subroutine;
2. `autovettoritaskA.out`: contiene le componenti degli autovettori ortonormalizzati calcolate dalla subroutine;

```
OPEN(23,FILE='autovalori.out')
WRITE(23,*) V
CLOSE(23)

OPEN(27,FILE='autovettoritaskA.out')
DO K=1,N
WRITE(27,*) X(K)  ,Z(K,1)  ,Z(K,2) ,Z(K,3)
end do
```

CASO $b = 0$

La condizione $b = 0$ rende le due buche del potenziale simmetriche rispetto all'origine e di tipo armonico per valori di x vicini ai minimi, conosciamo già le soluzioni analitiche dell'equazione di Shrödinger, le quali dipendono dai polinomi di Hermite e da un esponenziale Gaussiano:

$$\psi_n(x) = \frac{1}{\sqrt{2^n\, n!\sqrt{\pi}}}\, H_n(x)\, e^{-\frac{x^2}{2}}$$

É conveniente considerare l'intervallo di lavoro attorno all'origine, scegliendo una griglia uniforme con un numero dispari di punti. Per l'esperimento numerico si utilizzerà l'intervallo $[-11, 11]$ (per avere una doppia buca abbastanza larga affinchè i grafici delle autofunzioni tendano ad $y = 0$ per i più alti valori di $|x|$), con il numero di punti $N = 5001$. Dunque, nella prima Task si va alla ricerca dei primi M autovalori, ad esempio i primi 20. Il file di input è quello già definito in precedenza e sarà perciò strutturato come indicato in pagina successiva:

```
−11
11
5001
1
20
6.25
0
−0.5
0.01
```

Dopo aver compilato ed eseguito il programma principale si ottengono i seguenti risultati per quanto riguarda gli autovalori:

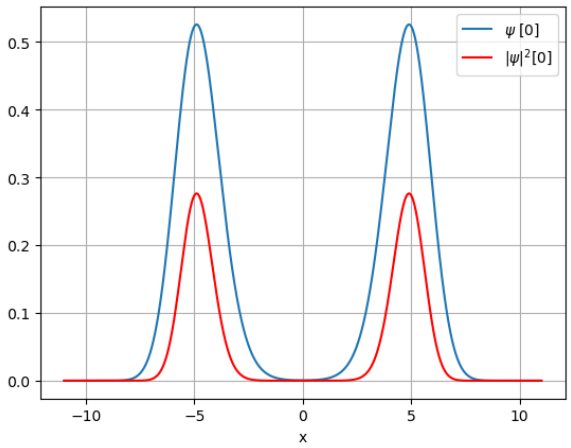
```

 0.489496953830894       0.489497565597048       1.42183664325239
   1.42192924492883        2.26551466086130        2.27064068987056
   2.93460935045914        3.03065475040603        3.45575428586562
   3.79056140959619        4.21241692517833        4.66164222952102
   5.14387673048077        5.65376224986358        6.18919674995959
   6.74826720828969        7.32946659306072        7.93152817915870
   8.55336825213999        9.19404278725960
```

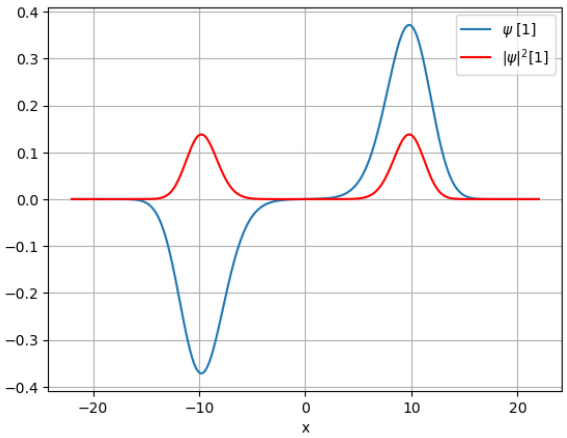
Listing 2: Autovalori task A

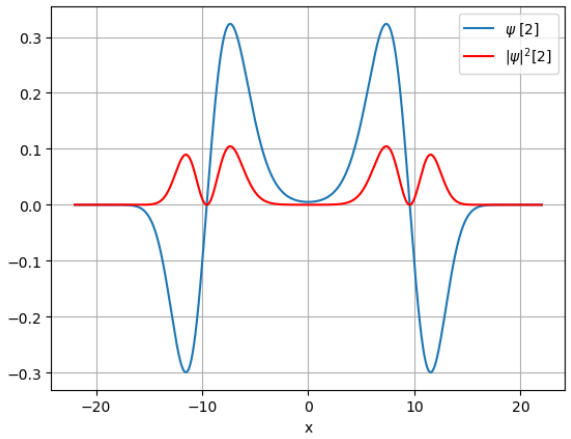
Si prosegue con la rappresentazione delle autofunzioni contenute nel file `autovettoritaskA.out`. A questo scopo, è stato scritto un codice Python attraverso Jupyter Notebook, chiamato `taskAgraphs.ipynb`. Tramite l'esecuzione delle celle, è possibile fornire in output i grafici col numero di punti desiderato a seconda dei dati presenti in `inputA.txt`.

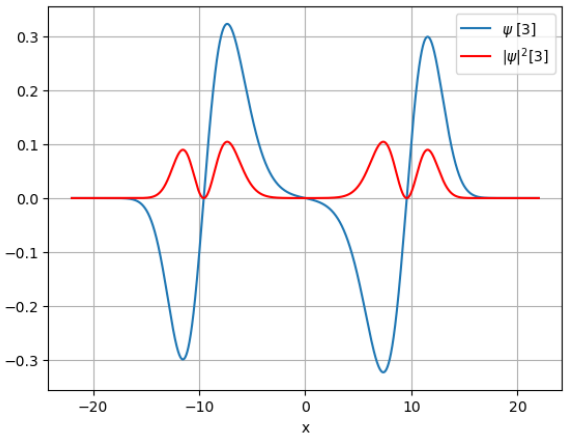
É importante notare come in realtà le autofunzioni ottenute non sono normalizzate, difatti in `taskAgraphs.ipynb` la condizione di normalizzazione, effettuata tramite il Metodo del Trapezio con una funzione specifica di `NumPy`, non fornisce l'unità. Solo gli autovettori della matrice di partenza sono normalizzati, dunque si è anche proceduto alla normalizzazione.

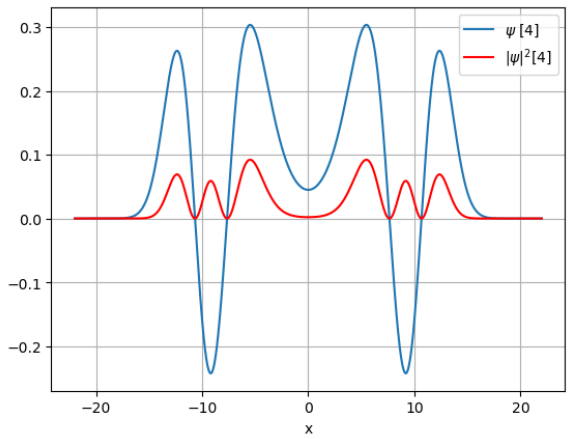
Nelle legende dei grafici vi è indicato il valore del numero quantico n; inoltre col colore blu sono rappresentate le autofunzioni calcolate, col colore rosso le loro densità di probabilità.

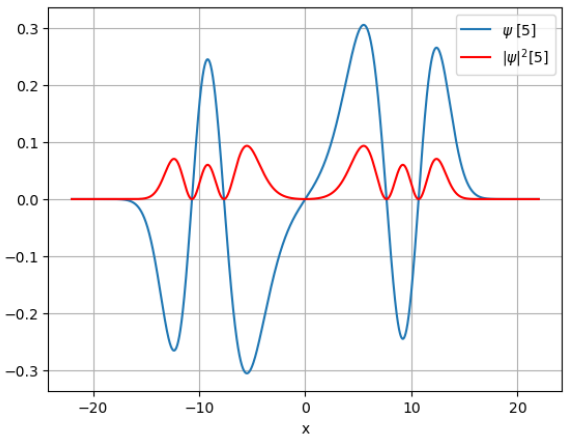

ψ [0]
|ψ|²[0]
x

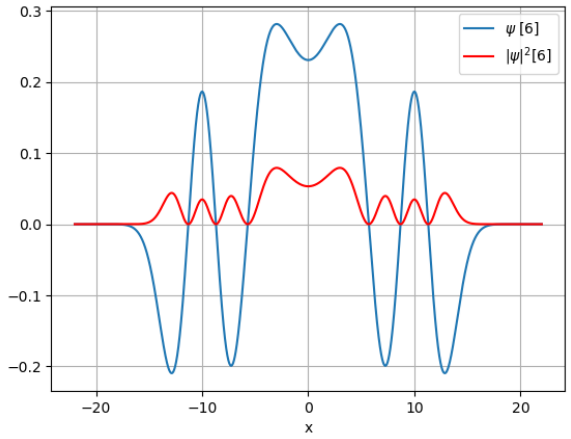

ψ [1]
|ψ|²[1]
x

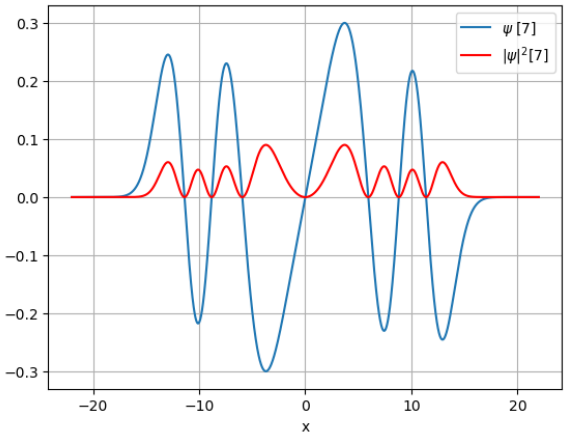

ψ [2]
|ψ|²[2]
x

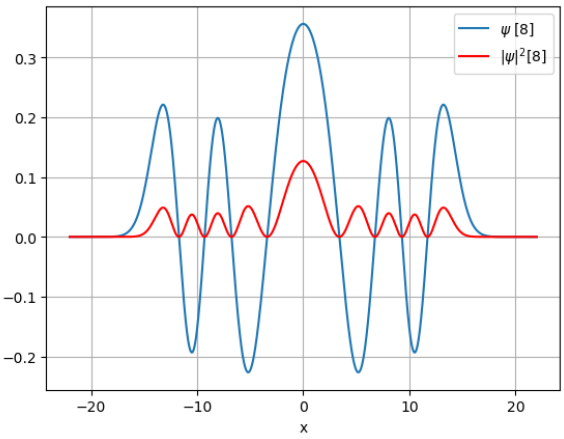

ψ [3]
|ψ|²[3]
x

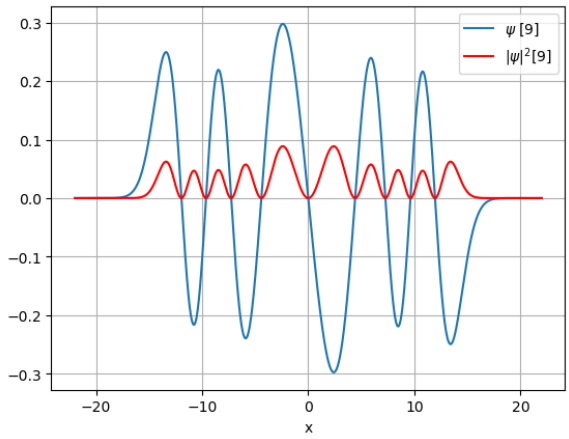

ψ [4]
|ψ|²[4]
x


ψ [5]
|ψ|²[5]
x


ψ [6]
|ψ|²[6]
x


ψ [7]
|ψ|²[7]
x





CASO $b \neq 0$

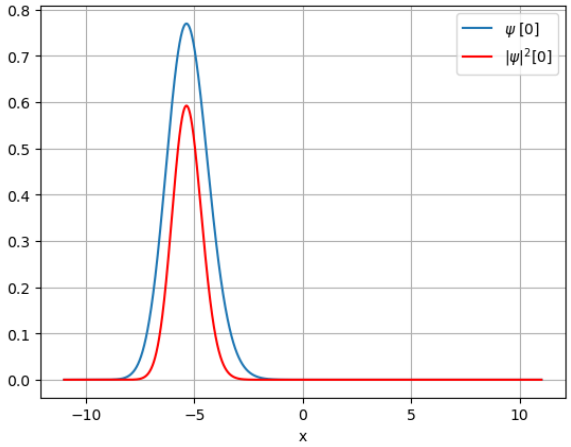
La condizione $b \neq 0$ rende asimmetrica la doppia buca facendo si che una delle due buche sia più profonda dell'altra. La procedura rimane identica a quella appena effettuata, con l'unica differenza che il file `inputA.txt` deve essere modifificato imponendo al parametro b un valore diverso da 0, in questo caso $b = 1$.

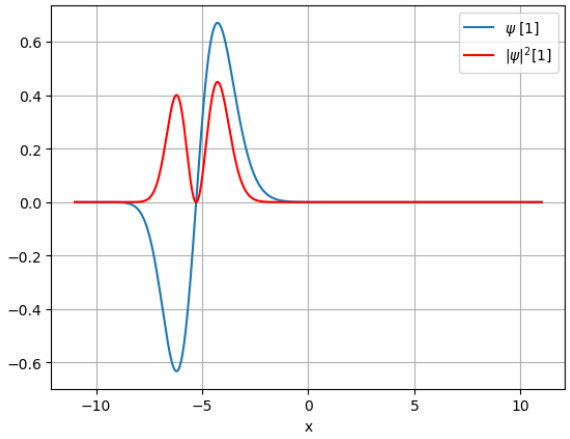
```
−11
11
5001
1
10
6.25
1
−0.5
0.01
```

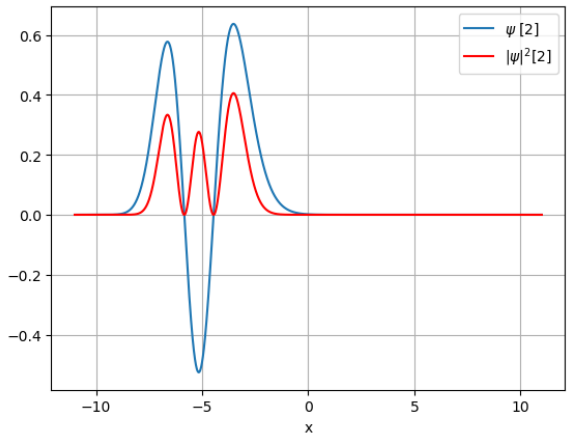
Listing 3: - File di input(`inputA.txt`)

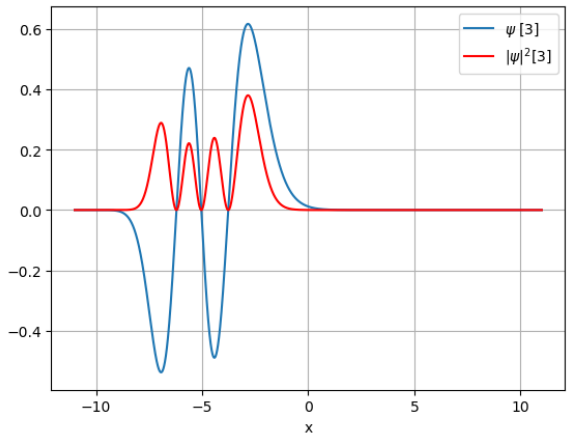
```
−2.05700380238159      −0.973100295383119      5.988343663375417E−002
 1.03388821509543       1.93628501900275        2.73759181869354
 2.74777234960675       3.32584331089856        3.48006953185359
 3.87821133465513       4.25777958963045        4.69688817551105
 5.16773610110067       5.66946117623411        6.19837758498215
 6.75225106008111       7.32924729316255        7.92787912355016
 8.54689879203033       9.18523922360873
```

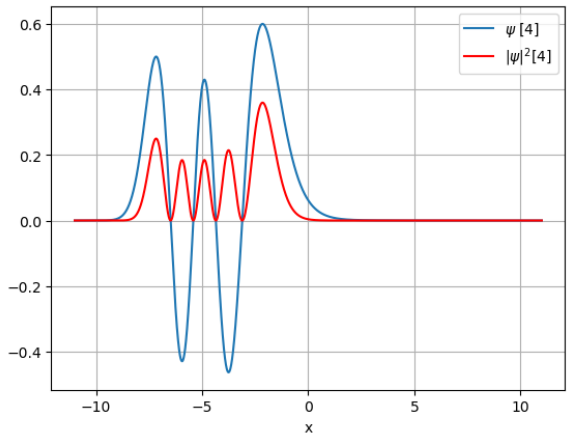
Listing 4: inputA.txt

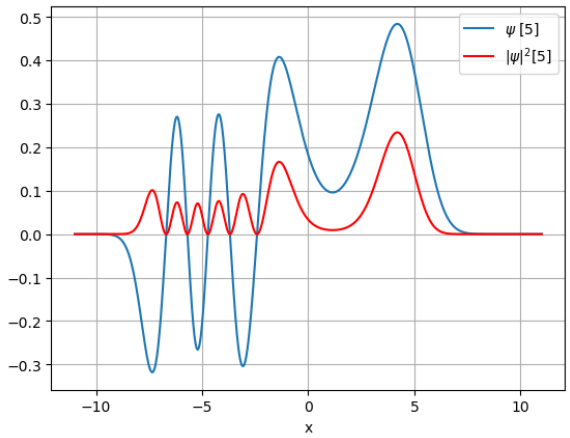

ψ [0]
|ψ|²[0]
x

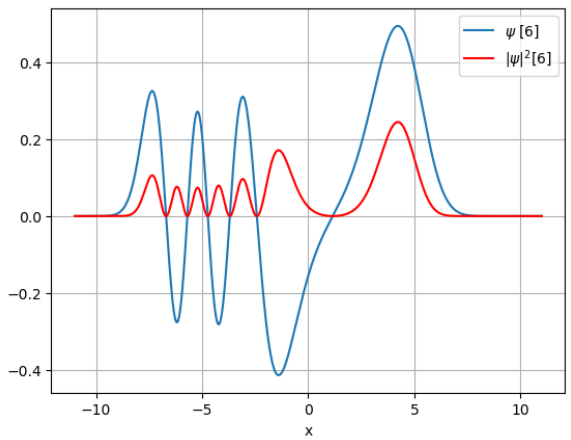

ψ [1]
|ψ|²[1]
x

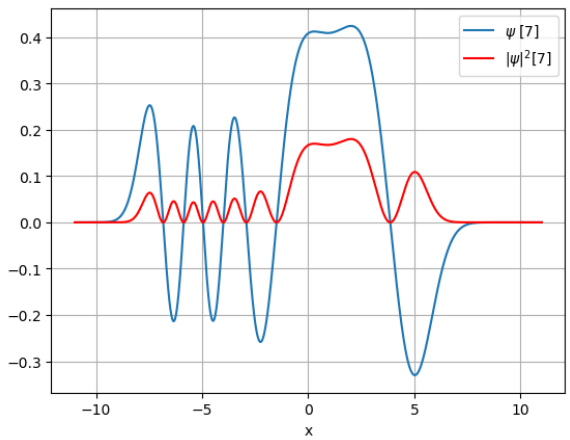

ψ [2]
|ψ|²[2]
x

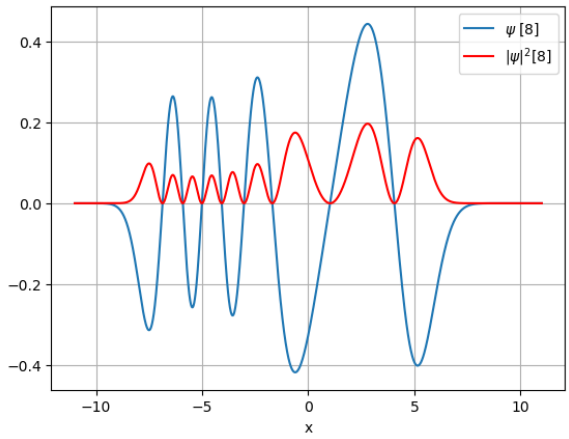

ψ [3]
|ψ|²[3]
x

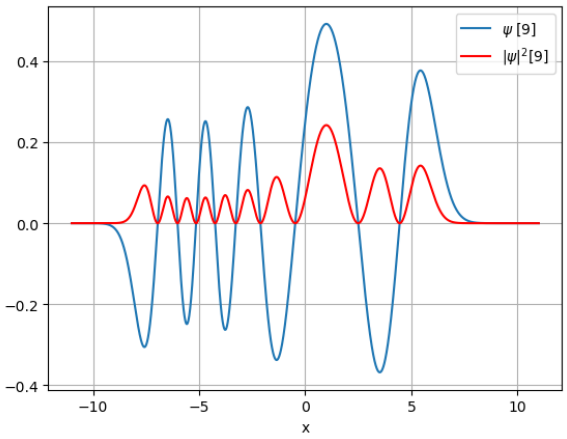

ψ [4]
|ψ|²[4]
x


ψ [5]
|ψ|²[5]
x


ψ [6]
|ψ|²[6]
x


ψ [7]
|ψ|²[7]
x





CONSIDERAZIONI SULLA SCELTA DEI PARAMETRI UTILIZZATI

In questa sezione sono stati cercati i migliori valori dei parametri L(larhezza della doppia buca)) ed N a coefficienti a, b, c, d fissati, i valori trovati sono poi quelli che verranno utilizzati per tutto il resto dell'esperimento. L'obbiettivo è quello di ottenere degli autovalori che raggiungano la convergenza entro una cifra decimale scelta a priori, facendo riferimento in particolar modo all'ultimo autovalore calcolato dall'algoritmo poichè è su questo che ricadono gli errori maggiori. Per la Task A il potenziale a doppia buca è del tipo $V(x) = 6.25 - 0.5x^2 + 0.01x^4$. Questo studio viene eseguito mantenendo costante uno dei due parametri L mentre il secondo varia fino a che non si trova un suo valore che faccia giungere l'ultimo autovalore calcolato a convergenza.Successivamente si userà il valore di L trovato facendo variare l'altro. Per questo esperimento si vuole raggiungere la convergenza sulla quarta cifra decimale. In questa prima tabella il parametro N è fissato ad un valore indicativamente corretto in base al grafico del potenziale, mentre varia L. Per fare questp esperimento consideriamo il decimo autovalore come l'ultimo autovalore calcolato.

L	N	autovalore
[-8,8]	5001	3.79869752612649
[-11,11]	5001	3.79056140959619
[-12,12]	5001	3.79055833344864

Come si nota dalla tabella la convergenza alla quarta cifra dopo la virgola viene raggiunta a partire dal valore $L = [-11, 11]$. É quindi sufficiente usare questo valore di L negli esperimenti numerici. Per $L = [-8, 8]$ invece la convergenza viene raggiunta sulla seconda cifra dopo la virgola.
Ora il parametro $L = [-11, 11]$ rimane fissato mentre varia N.

L	N	autovalore
[-11,11]	201	3.78044753216760
[-11,11]	5001	3.79056140959619
[-11,11]	80001	3.79057752946392

La convergenza alla quarta cifra dopo la virgola la si ottiene già a partire da $N = 5001$. Per $N = 201$ la convergenza ricade sulla prima cifra decimale, mentre per $N = 80001$ rimane sempre sulla quarta. Conviene dunque usare $N = 5001$. Osserviamo inoltre come i valori di N sopra riportati siano tutti numeri dispari,

questo poichè l'algoritmo di calcolo richiede una simmetria della griglia rispetto all'origine. Sulla base delle prove effettuate si può affermare quanto segue:

- Valori piccoli di N oltre a non garantire una buona convergenza fanno si che i grafici delle funzioni d'onda risultino eccessivamente segmentati. La funzione infatti non viene riprodotta sufficientemente bene a causa dei pochi punti della griglia e ad un conseguente Δx grande tra essi.
- il numero di punti condiziona chiaramente il tempo di calcolo, ma soprattutto, in questo caso, la convergenza dei risultati, migliorandola.

RISULTATI E CONCLUSIONI PER LA TASK A

L'esperimento numerico si è concluso correttamente in quanto è stato possibile ricavare i valori dei primi IU autovalori del sistema sia per il caso con $b = 0$ sia per quello con $b \neq 0$, i quali si presentano a due a due confrontbili tra loro entro un certo numero di cifre significative scelte a priori. In particolar modo la prima coppia e la seconda presentano autovalori uguali entro la quarta cifra dopo la virgola; questo suggerisce che gli stati caratterizzati dallo stesso autovalore sono presenti in entrambe le buche di potenziale, entro l'errore appena definito. I grafici lo confermano, ad ogni curva di densità di probabilità ne corrisponde un'altra uguale simmetrica rispetto all'origine, ad indicare la degenerazione degli stati dovuta alla natura del potenziale. Si nota però anche che più ci si allontana dallo stato fondamentale e più gli autovalori divergono a due a due, questo verrà ripreso in seguito nella discussione della degenerazione dell'energia.
Nel caso del potenziale con $b \neq 0$ non è possibile fare le stesse considerazioni. Questi autovalori sono infatti tutti diversi tra loro, infatti il coefficiente $b \neq 0$ fa perdere al sistema la degenerazione dell'energia. Dai grafici corrispondenti ai primi autovalori si nota come i picchi di densità di probabilità siano in numero maggiore per valori negativi della x; questo conferma che, in corrispondenza dei primi autovalori, la probabilità di trovare l'elettrone nella buca più profonda (quella definita per valori negativi di x) è maggiore.

TASK B

Un'equazione differenziale puó essere risolta anche utilizzando una traformata di Fourier scegliendo una base opportuna per lo sviluppo. La rappresentazione in una base é conveniente se la base scelta è in grado di diagonalizzare una parte dell'Hamiltoniana, essendo noti gli elementi diagonali. Inoltre, se la parte non diagonale dell'Hamiltoniana é debole rispetto agli elementi diagonali, la convergenza é rapida. In questo esperimento viene scelta la base di onde piane in quanto é in grado di diagonalizzare l'energia cinetica.
Si consideri un set di funzioni ortonormali $u_k(x)$, dove k é un set di numeri quantici.

$$\int_\Omega u_k^\star(x) u_{k'}(x) dx = \delta_{k,k'}$$

La soluzione viene scritta come combinazione lineare del set di funzioni di base,

$$\phi(x) = \sum_k c_k u_k(x)$$

viene poi sostituita nell'equazione di Schrödinger unidimensionale ed infine viene moltiplicata a sinistra per:

$$\int_\Omega u_{k'}^\star(x) dx$$

In questo modo si ottiene un sistema la cui equazione k'-esima risulta:

$$\sum_k c_k \left[\int_\Omega u^\star_{k'}(x) \left(-\frac{\partial^2}{\partial x^2} \right) u_k(x) dx + \int_\Omega u^\star_{k'}(x) V(x) u_k(x) dx - 2E \int_\Omega u^\star_{k'}(x) u_k(x) dx \right] = 0$$

All'interno delle parentesi quadre i tre integrali che si sommano tra loro rappresentano, nell'ordine, l'energia cinetica $K_{k,k'}$, il potenziale $V_{k,k'}$ e la delta di Kronecker $\delta_{k,k'}$.
Prendendo come base quella delle onde piane:

$$u_k(x) = \frac{1}{\sqrt{L}} e^{ikx}$$

di vettore d'onda k, che assume valori reali multipli di $\frac{2\pi}{L}$, le funzioni di base saranno periodiche di periodo L.

$$K_{k,k'} = k^2 \delta_{k,k'}$$

$$V_{k,k'} = \int_\Omega u^\star_{k'}(x) V(x) u_k(x) dx = \frac{1}{L} \int_{-\frac{L}{2}}^{\frac{L}{2}} V(x) e^{-ix(k'-k)} dx$$

Il calcolo di questo integrale verrà effettuato sfruttando il metodo della trasformata di fourier veloce (FFT).
La k-esima equazione del sistema può quindi essere riscritta in forma matriciale:

$$H \cdot c = 2Ec$$

dove quello tra H e c é un prodotto matrice per colonna, essendo $c = (\dots c_k \dots)^T$.

$$H = \begin{pmatrix} \ddots & & V_{k,k'} \\ & k^2 + V_{k,k'} & \\ V_{k',k} & & \ddots \end{pmatrix}$$

Lo sviluppo in onde piane porta ad una rappresentazione matriciale che ha caratteristiche opposte rispetto alla rappresentazione in spazio reale. La matrice in spazio di Fourier ha dimensioni limitate ma é densa, mentre la matrice in spazio reale é di dimensioni maggiori ma sparsa e tridiagonle. Per questo le routines da utilizzare per la diagonalizzazione saranno differenti nei due casi. La diagonalizzazione di H, così come il calcolo degli autovalori e delle autofunzioni, è svolta dalla subroutine `zheevx` presente in una delle librerie `LAPACK`, adatta in questo a caso al tipo di matrice: hermitiana e densa.
La subroutine `zheevx` richiede che vengano definiti specifici parametri per poter funzionare, l'elenco e il significato di questi è stato riportato in `Appendice B`:

OBBIETTIVI

- Determinare i primi M autovalori di energia piú bassa e le relative autofunzioni, confrontando un caso con $b = 0$ ed uno con $b \neq 0$.

- Confrontare gli stati localizzati nelle buche con quelli trovati nel caso di un potenziale armonico.

Prima di procedere con il caso del potenziale a doppia buca viene testato l'uso della routine che esegue la FFT nel caso del potenziale armonico $V(x) = x^2$, che può essere risolto analiticamente e di cui sono note le soluzioni, così da poterne accertare la validità.
Si definisce $\Delta k = k' - k$, così da poter scrivere:

$$V_{k,k'} = \begin{cases} \frac{(\Delta k^2(\frac{L}{2})^2 - 2)\sin(\Delta k \frac{L}{2}) + \Delta k L \cos(\Delta k \frac{L}{2})}{\Delta k^3 (\frac{L}{2})} & per \;\; k \neq k' \\ \frac{(\frac{L}{2})^2}{3} & per \;\; k = k' \end{cases}$$

Nei casi del potenziale armonico e della doppia buca bisogna compilare, oltre al programma principale, due moduli per il calcolo della trasformata di Fourier veloce e per la diagonalizzazione della matrice.
I moduli, usati per entrambi i potenziali, vanno compilati nel modo seguente:

```
ifort -check all -c trasf.f90
```

```
ifort -check all -c matriceB.f90
```

Il programma principale nel caso del potenziale armonico viene compilato tramite i seguenti comandi:

```
ifort -check all -o PROVA.exe oscarmB.f90 trasf.o matriceB.o -qmkl
./PROVA.exe
```

Mentre nel caso del potenziale a doppia buca:

```
ifort -check all -o taskB.exe trasfdoppiabuca.f90 trasf.o matriceB.o -qmkl
./taskB.exe
```

CODICE PER LA TASK B, CASO ARMONICO

File di input

```
A
B
N
IL
IU
K
```

Listing 5: - Struttura del file di input

In questa seconda task è stato introdotto in input il parametro K che rappresenta il numero di onde piane utilizzate, inoltre a differenza del caso precedente N non rappresenta più il numero di punti ma bensì il numero di intervalli.

Main Program

Il programma principale, noto come `oscarmB.f90`, viene compilato tramite Intel, generando l'eseguibile `PROVA.exe`.
Si divide in 4 parti:

1. lettura del file di input per i dati iniziali;
2. definizione dei vettori da utilizzare e del potenziale armonico;
3. calcolo della FFTW del potenziale (trasformata in avanti);
4. costruzione della matrice da diagonalizzare e chiamata della subroutine di libreria che svolge la diagonalizzazione.

Si passa ora alla discussione di alcune parti importanti del codice.

```
PROGRAM main
    USE fftw_modulo
    USE zheevx_modulo
```

Listing 6: -richiamo dei moduli da utilizzare

Come mostrato in figura le prime righe del codice sono state scritte per ricorrere ai moduli `zheevx_modulo` ed `fftw_modulo`, il primo dei quali svolge la parte riguardante la diagonalizzazione della matrice H, mentre il secondo calcolala traformata di Fourier del potenziale.

```
  L = B-A
        deltax = L/dble(N)
        deltaq = 2*pi/L
        X(1) = -L/2
        Q(1) = -(pi*N)/L
        DO i = 1,N
            X(i+1) = X(i) + deltax
        END DO
        DO g = 1,N
            Q(g+1) = Q(g) + deltaq
        END DO
        V = X**2

        CALL fttw_calcolo_FORWARD(L,N,deltax,V,V_T)
        OPEN(36,FILE='V_T.DAT')
        OPEN(37,FILE='Q.DAT')
        DO P=1,N+1
            WRITE(36,*) DREAL(V_T(P))
            WRITE(37,*) Q(P)
        END DO
        CLOSE(36)
        CLOSE(37)
```

In questo pezzo del codice è dedicato alla definizione dell'intervallo spaziale e della griglia di punti,inoltre vengono creati i Δk (Q nel codice) tra i vettori d'onda k successivi ed il potenziale armonico (V nel codice).

```
        CALL fttw_calcolo_FORWARD(L,N,deltax,V,V_T)
      OPEN(36,FILE='V_T.DAT')
      OPEN(37,FILE='Q.DAT')
      DO P=1,N+1
           WRITE(36,*) DREAL(V_T(P))
           WRITE(37,*) Q(P)
      END DO
      CLOSE(36)
      CLOSE(37)
```

in questa sezione viene calcolato $V_{k,k'}$, che permetterà di costruire la matrice H da diagonalizzare. Per fare ciò viene richiamata la subroutine `fttw_`, contenuta nel modulo `fftw_modulo`, appositamente costruita per calcolare la trasformata di Fourier veloce in avanti del potenziale armonico.

```
! Costruzione della matrice da diagonalizzare
      ALLOCATE(MATRIXA(1:2*K+1,1:2*K+1))
      MATRIXA = 0.0
      DO S=-K,K
           DO C=S,K
                MATRA(K+S+1,K+C+1) = V_T(N/2+1+S-C)*0.5
           END DO
           MATRIXA(K+S+1,K+S+1) = MATRIXA(K+S+1,K+S+1) + 0.5*(S*2*pi/L)**2
      END DO


     CALL zheevx_(MATRIXA,2*K+1,IU,IL)
```

Infine questa sezione crea la matrice H e richiama la subroutine `zheevx_`, contenuta nel modulo `zheevx_modulo`, appositamente costruita per diagonalizzare la matrice in questione.

Modulo per il calcolo della FFTW

Questo modulo(`trasf.f90`), riportato interamente in appendice al documento, può essere diviso in due parti:

1. Preparazione della funzione in esame affinchè la trasformata di Fourier in avanti possa agire su di essa. La manipolazone della funzione avviene affinchè la trasformata venga eseguita correttamente senza nessuno sfasamento del risultato; per questo viene tolto un punto all'estremità del dominio (si passa da $N+1$ ad N punti) poichè la funzione pari si ripete all'estremo.

2. Calcolo della trasformata di Fourier e scrittura del risultato in output nei text file `V_T.txt` e `DeltaK.txt`

Modulo per la diagonalizzazione della Matrice H

Il modulo `matriceB.f90`, riportato interamente in appendice al documento, contiene la subroutine `zheevx_` che, per funzionare, necessita di includere `mkl_lapack.fi` per le ragioni già specificate in precedenza.
Il codice sfrutta la subroutine di libreria `zheevx` per il calcolo degli autovalori e degli autovettori, i risultati di questa operazione vengono poi stampati come output su una serie di text file:

File di output

File genearati in output dal modulo per il calcolo della FFTW:`V_T.txt`: contiene le parti reali di $V_{k,k'}$ calcolate tramite il modulo della trasformata di Fourier; `DeltaK.txt`: contiene tutti i valori di Δk (Q nel codice) tra i vettori d'onda k successivi;

File generati dal modulo della subroutine per la diagonalizzazione:

1. `autovalori.out`: contiene gli autovalori calcolati dalla subroutine;
2. `autovettori.out`: contiene le componenti reali degli elementi della matrice complessa Z;
3. `imm.out`: contiene le componenti immaginarie degli elementi della matrice complessa Z.

RAPPRESENTAZIONE DELLE AUTOFUNZIONI

In questo caso non essendo presente una doppia buca non si deve ricorrere al confronto tra il caso $b = 0$ con quello $b \neq 0$.
Gli autovalori del potenziale armonico sono noti, l'obiettivo è quindi ottenere dei risultati che si avvicinino il più possibile a questi entro un certo numero di cifre scelto a priori. Per la convergenza degli autovalori è stata scelta la terza cifra decimale dopo la virgola. Tale cifra dunque non dovrà variare tra un autovalore ed un altro.
Di seguito il file di input:

```
−10
10
5000
1
4
20
```

In questo caso è stato scelto un numero $k = 20$ di onde piane. Dopo varie prove, con valori di k diversi, si è notato che è grazie a $k = 20$ che si ottiene la convergenza degli autovalori con il maggior numero di cifre decimali dopo la virgola.A differenza del caso precedente, il numero di punti `N=5000` non è in grado di consentire una convergenza degli autovalori entro la quarta cifra dopo la virgola, la quale viene raggiunta solo con circa 40000 punti.

N=8000

```
2
   0.500062513915510          1.50018754174651          2.50031256957751
4    3.50043759740864
```

N=20000

```
   0.500025013914465          1.50007504174335          2.50012506957220
2    3.50017509740118          4.50022512523254
```

N=40000

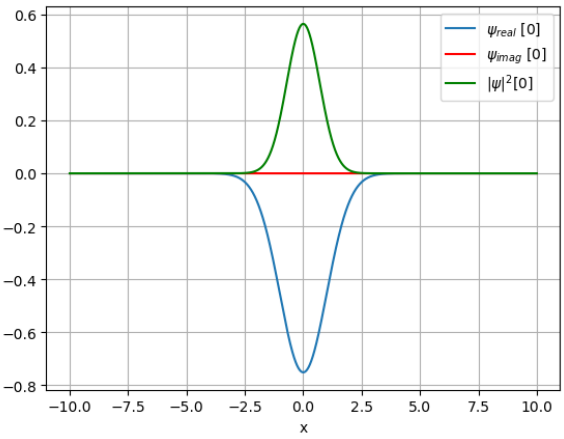
```
   0.500012513914085          1.50003754174225          2.50006256957044
2    3.50008759739885
```

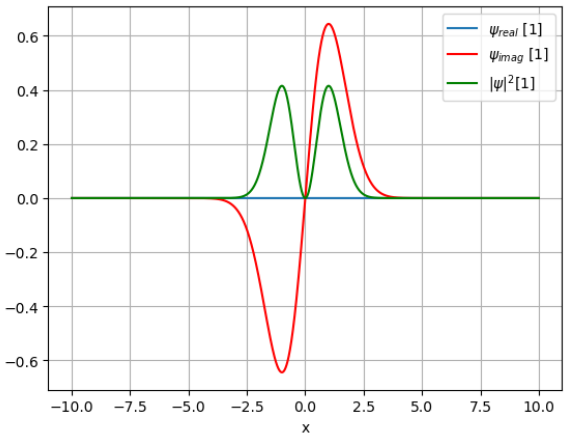
Per non andare a inficiare eccessivamente il tempo di compilazione si è quindi deciso di accettare la convergenze a 3 cifre decimali dopo la virgola e di mantenere N=5000. La compilazione dei moduli e del programma principale porta ai seguenti risultati:

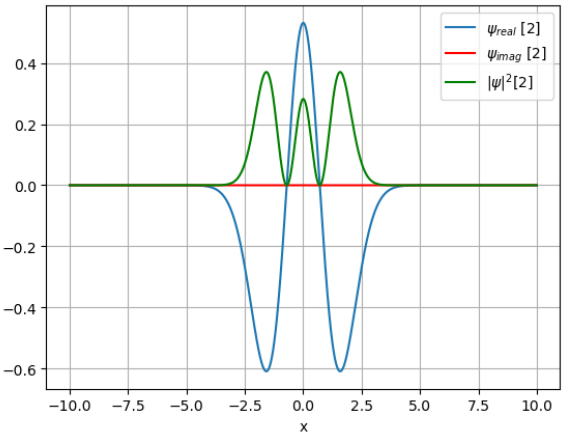
```
   0.500100013916552          1.50030004174965          2.50050006958276
2    3.50070009741599
```

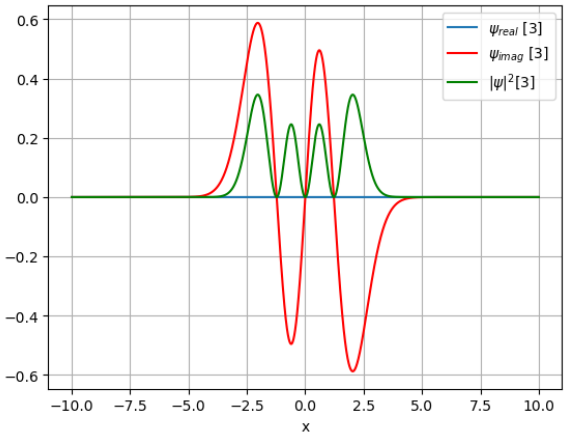
Listing 7: autovalori.out

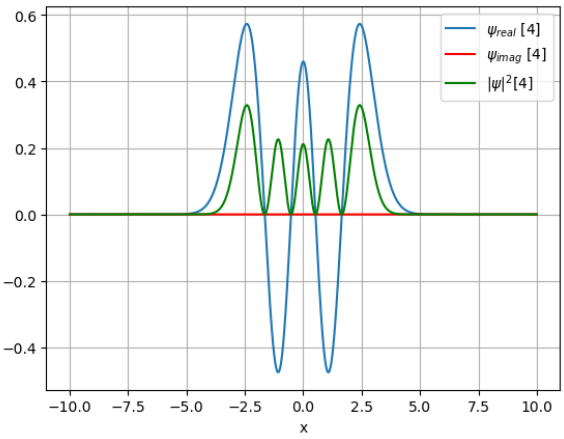
Dove per semplicità di trattazione si è scelto in questo di considerare solo i primi 4 autovalori in quanto per i successivi il numero di punti richiesto sarebbe ancora maggiore. Di seguito vengono riportate le rappresentazioni grafiche delle autofunzioni in cui è indicato il valore del numero quantico n. In blu viene rappresentata la parte reale delle autofunzioni, in rosso la loro parte immaginaria ed infine in verde la densità di probabilità.

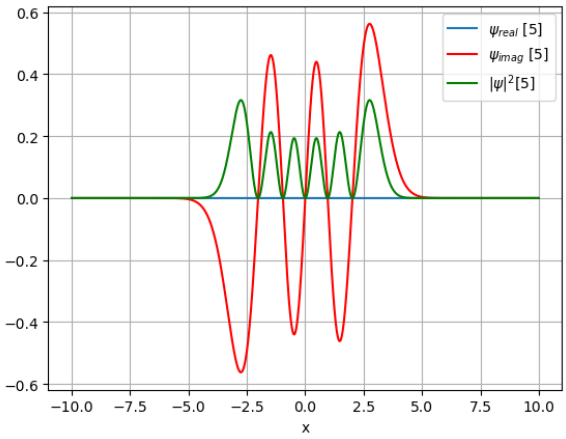

ψ_{real} [0]
ψ_{imag} [0]
$|\psi|^2$[0]
x

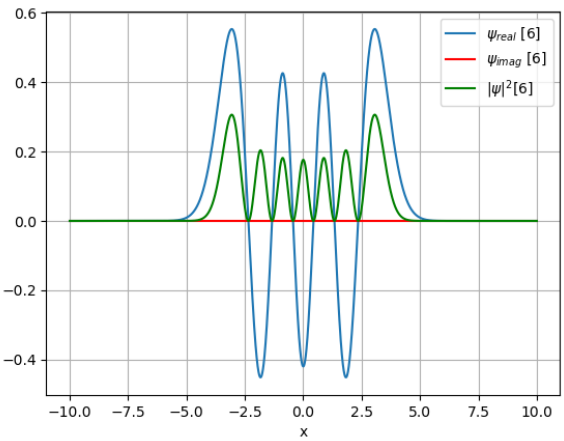

ψ_{real} [1]
ψ_{imag} [1]
$|\psi|^2$[1]
x

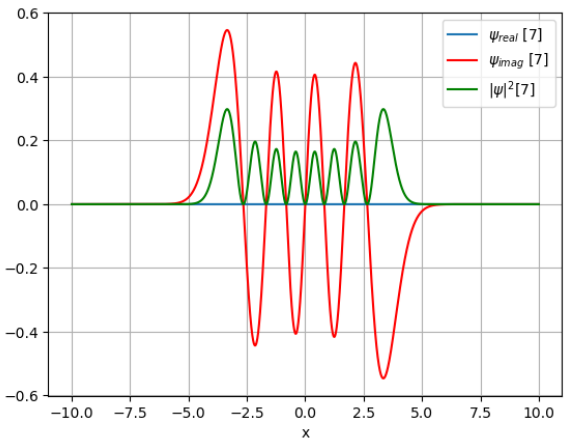

ψ_{real} [2]
ψ_{imag} [2]
$|\psi|^2$[2]
x

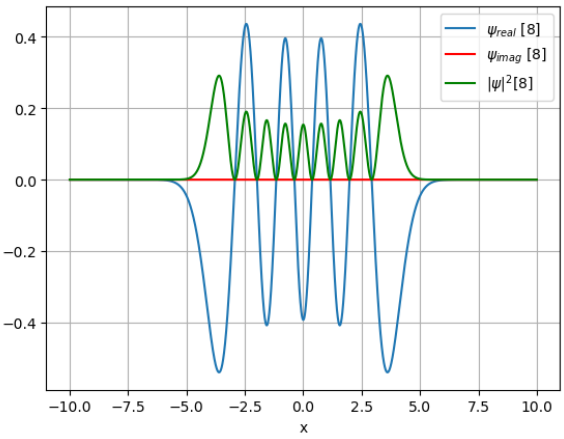

ψ_{real} [3]
ψ_{imag} [3]
$|\psi|^2$[3]
x

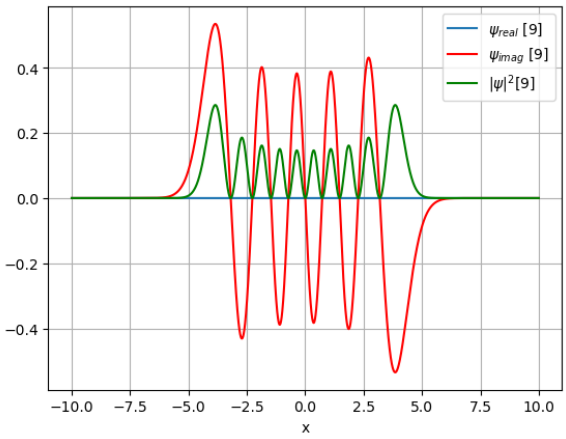

ψ_{real} [4]
ψ_{imag} [4]
$|\psi|^2$[4]
x


ψ_{real} [5]
ψ_{imag} [5]
$|\psi|^2$[5]
x


ψ_{real} [6]
ψ_{imag} [6]
$|\psi|^2$[6]
x


ψ_{real} [7]
ψ_{imag} [7]
$|\psi|^2$[7]
x





DISCUSSIONE DEI RISULTATI E CONCLUSIONI PER LA TASK B, POTENZIALE ARMONICO

Gli autovalori ottenuti sono confrontabili, entro la terza cifra decimale dopo la virgola compresa, con quelli noti del potenziale armonico. L'errore aumenta all'aumentare del numero quantico n. A differenza del caso studiato nella Task A la buca di potenziale è unica e per questo ciascun autovalore compare una sola volta. Come sottolineato in precedenza, tuttavia, a parità di N l'utilizzo della trasformata di fourier veloce porta a una convergenza degli autovalori con un minor numero di cifre decimali dopo la virgola.

CODICE PER LA TASK B, CASO DELLA DOPPIA BUCA

Input file

La lista dei parametri di input è identica a quella nel caso del potenziale armonico, con l'aggiunta dei coefficienti `a,b,c,d` che comportano la presenza di una doppia buca di potenziale.

Programma principale e File di Output

Il programma principale, noto come `trasfdoppiabuca.f90`, viene compilato tramite Intel, generando l'eseguibile `taskB.exe`. Il codice è formalmente identico a quello del potenziale armonico se non per la natura del potenziale che in questo caso presenterà dei termini aggiuntivi che caratterizzano la doppia buca, anche i file di output sono rimasti esattamente gli stessi.
L'intero codice è riportato in appendice al documento.

CASO $b = 0$

In questo caso l'intervallo che determina la larghezza della doppia buca è $[-10, 10]$, suddiviso in $N = 4096$ intervalli. Come prima vengono rappresentati i primi dieci autovalori, ma qui è necessario inserire anche il numero di onde piane che si desidera usare, in questo caso $k = 20$. I coefficienti del potenziale sono gli stessi dell'esperimento numerico in Task A, così da poter confrontare gli autovalori calcolati con questi due metodi differenti. Di seguito il file input:

```
−10
10
5000
1
20
20
6.25
0
−0.5
0.01
```

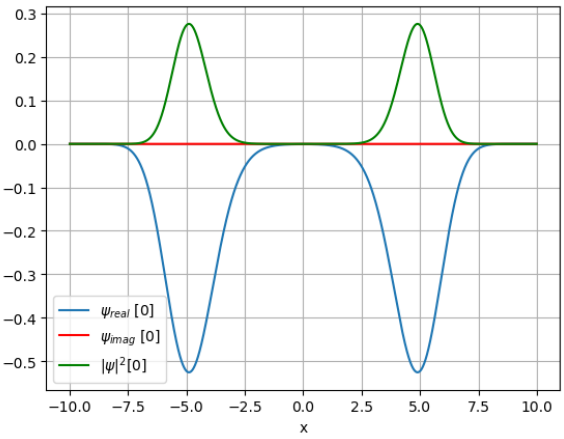
Listing 8: File di input

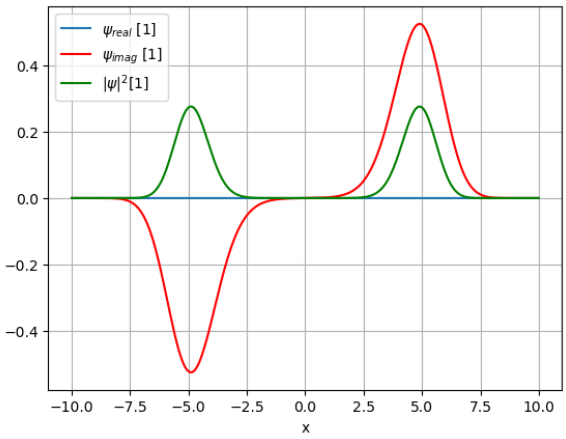
Gli autovalori che si ottengono in output sono:

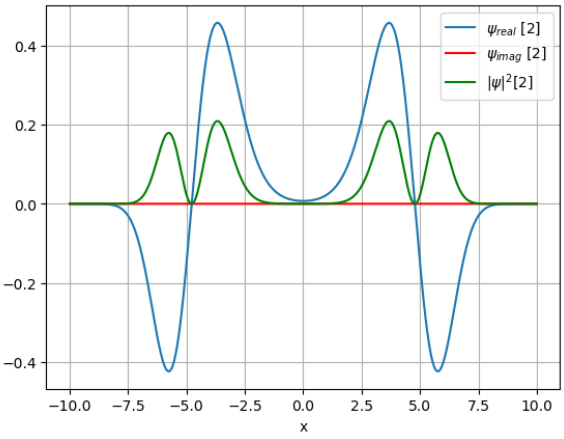
```
 0.489593219139938       0.489593832980470       1.42210569007544
   1.42219858390759        2.26590630928391        2.27104660230674
   2.93497184640785        3.03119338758768        3.45630371987890
   3.79134083313773        4.21334736690910        4.66274729896469
   5.14515667109421        5.65522377968652        6.19084610206164
   6.75011099990546        7.33151141420160        7.93378075195527
   8.55583559369776        9.19673219386149
```

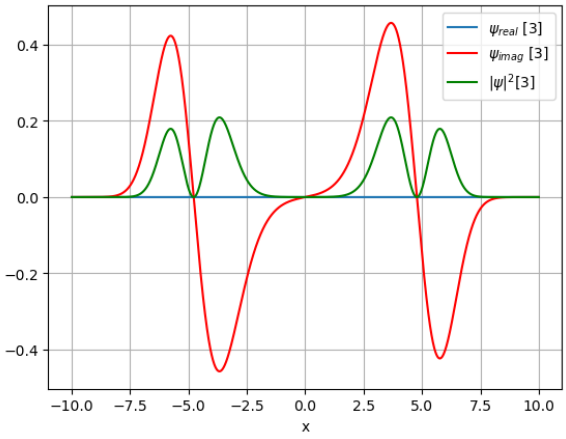
Listing 9: autovalori.out

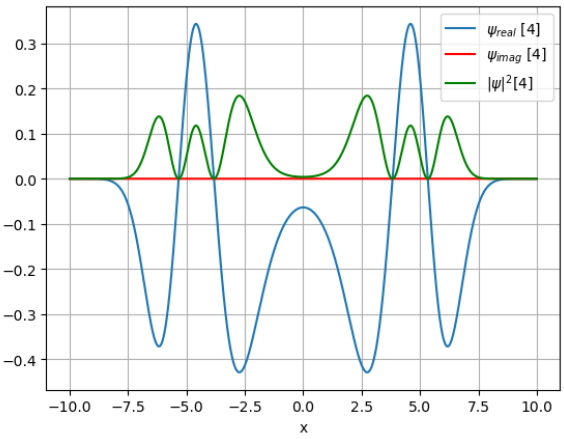
Di seguito la rappresentazione grafica delle autofunzioni in cui è indicato il valore del numero quantico n. In blu viene rappresentata la parte reale delle autofunzioni, in arancione la loro parte immaginaria ed infine in verde la densità di probabilità.

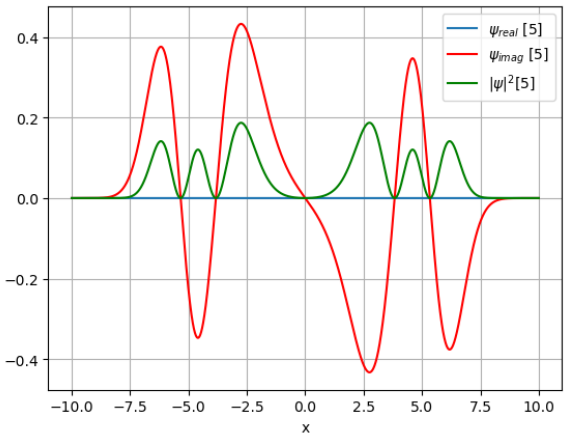

ψ_{real} [0]
ψ_{imag} [0]
$|\psi|^2$[0]
x

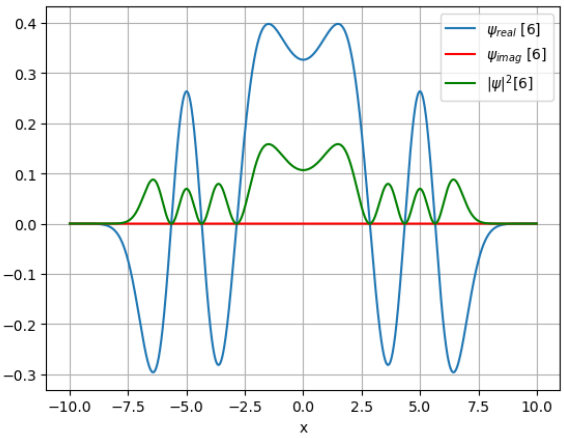

ψ_{real} [1]
ψ_{imag} [1]
$|\psi|^2$[1]
x

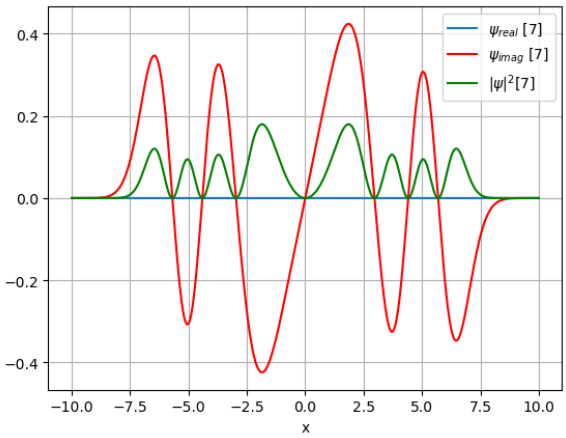

ψ_{real} [2]
ψ_{imag} [2]
$|\psi|^2$[2]
x

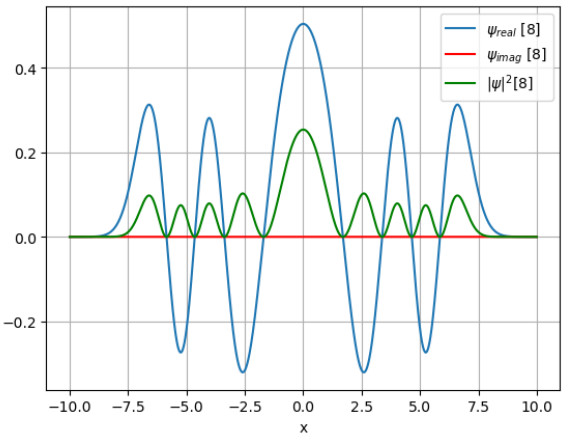

ψ_{real} [3]
ψ_{imag} [3]
$|\psi|^2$[3]
x

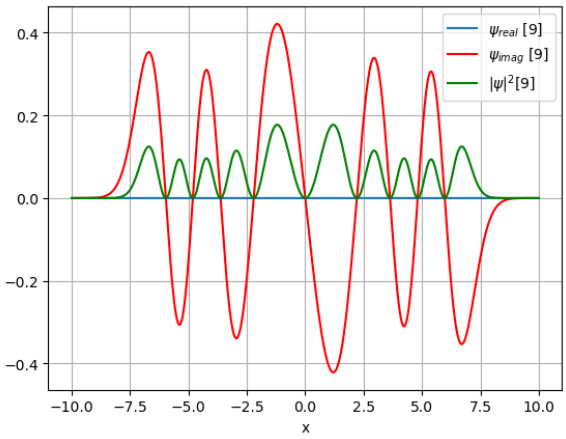

ψ_{real} [4]
ψ_{imag} [4]
$|\psi|^2$[4]
x


ψ_{real} [5]
ψ_{imag} [5]
$|\psi|^2$[5]
x


ψ_{real} [6]
ψ_{imag} [6]
$|\psi|^2$[6]
x


ψ_{real} [7]
ψ_{imag} [7]
$|\psi|^2$[7]
x





Confronto autovalori e grafici delle autofunzioni di Task A e Task B, caso $b = 0$

Gli autovalori di questa doppia buca sono stati calcolati diversamente nelle due task A e B. Come si può notare dalle tabelle sotto riportate gli autovalori corrispondono tra loro entro almeno la terza cifra decimale dopo la virgola per quanto riguarda i primi 5 autovalori, passando alla seconda cifra per quelli successivi. Questo indica che i due algoritmi di calcolo possono essere usati indifferentemente sole se si vuole lavorare con una precisione che ricade al massimo entro 2 o 3 cifre decimali dopo la virgola. Come ulteriore conferma si notino i grafici delle autofunzioni che mostrano, tra un metodo e l'altro, delle densità di probabilità pressochè identiche tra loro, per n fissato.

```
0.489496953830894      0.489497565597048      1.42183664325239
  1.42192924492883       2.26551466086130       2.27064068987056
  2.93460935045914       3.03065475040603       3.45575428586562
  3.79056140959619       4.21241692517833       4.66164222952102
  5.14387673048077       5.65376224986358       6.18919674995959
  6.74826720828969       7.32946659306072       7.93152817915870
  8.55336825213999       9.19404278725960
```

Listing 10: Autovalori Task A

```
  0.489593219139938      0.489593832980470       1.42210569007544
  1.42219858390759       2.26590630928391       2.27104660230674
  2.93497184640785       3.03119338758768       3.45630371987890
  3.79134083313773       4.21334736690910       4.66274729896469
  5.14515667109421       5.65522377968652       6.19084610206164
  6.75011099990546       7.33151141420160       7.93378075195527
  8.55583559369776       9.19673219386149
```

Listing 11: Autovalori Task B

CASO $b \neq 0$

In questo caso abbiamo b=1, i restanti parametri di input rimangono invariati.

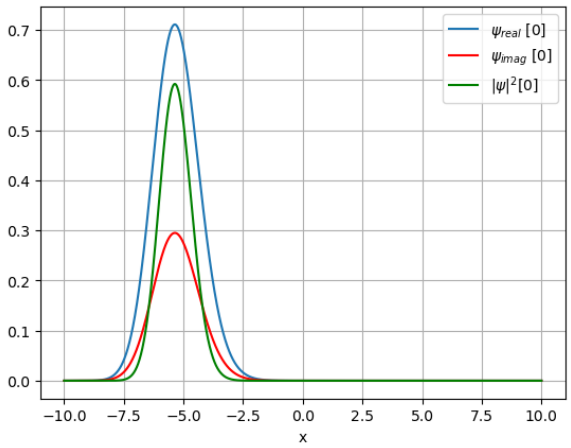
```
−10
10
5000
1
10
20
6.25
1
−0.5
0.01
```

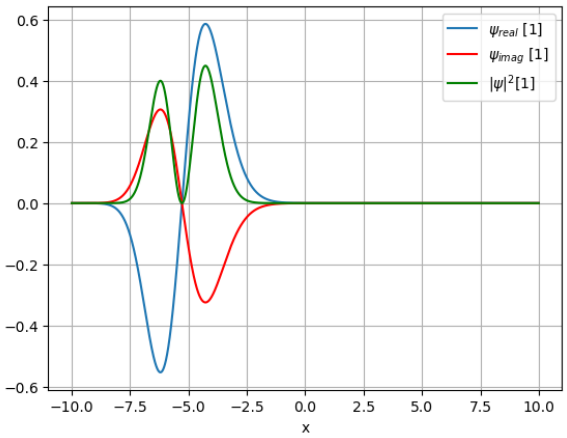
Listing 12: Input file

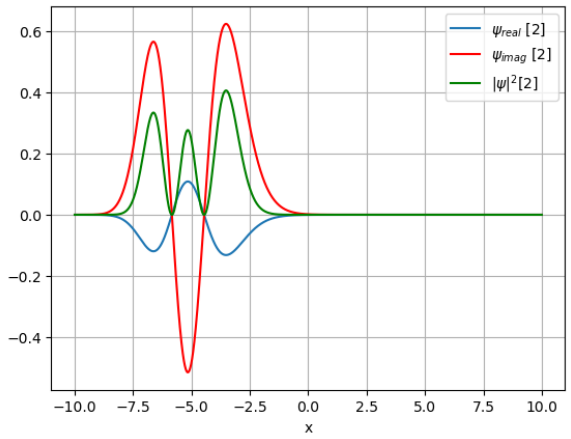
```
−2.05689303626143      −0.972779749556995      6.039337150382251E−002
   1.03456009321336        1.93707830546628        2.73801794114073
   2.74825698139023        3.32621548035866        3.48069110957680
   3.87890749821765        4.25868991306138        4.69796902920886
   5.16899909982927        5.67090954674772        6.20001738117135
   6.75408882420579        7.33129055480599        7.93013741003988
   8.54938616542476        9.18797977807983
```

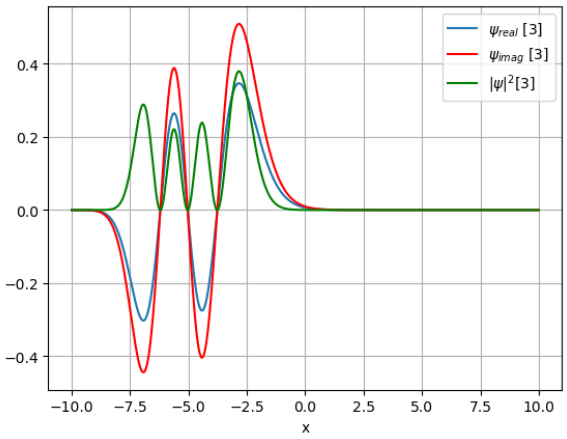
Listing 13: autovalori.out

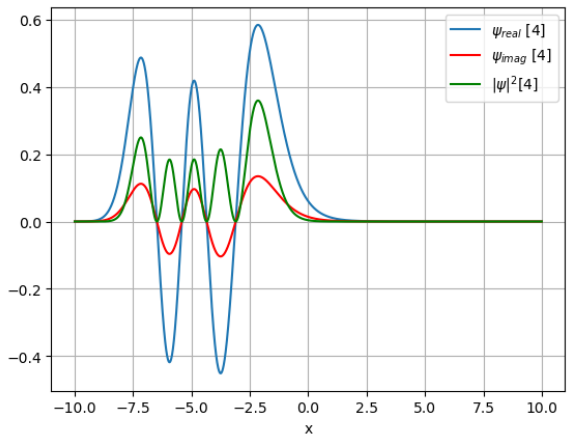
Di seguito i grafici delle funzioni d'onda. La legenda è la stessa utilizzata in precedenza.

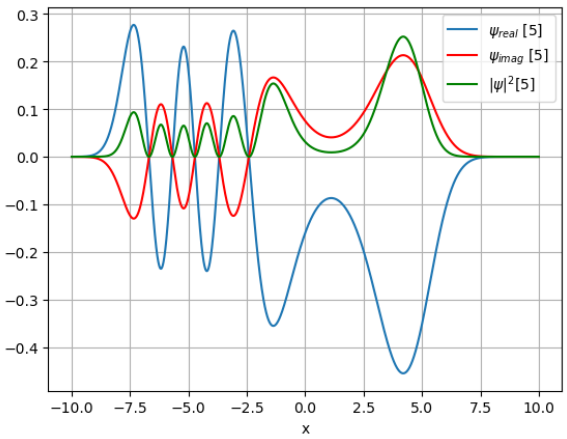

ψ_{real} [0]
ψ_{imag} [0]
$|\psi|^2$[0]
x

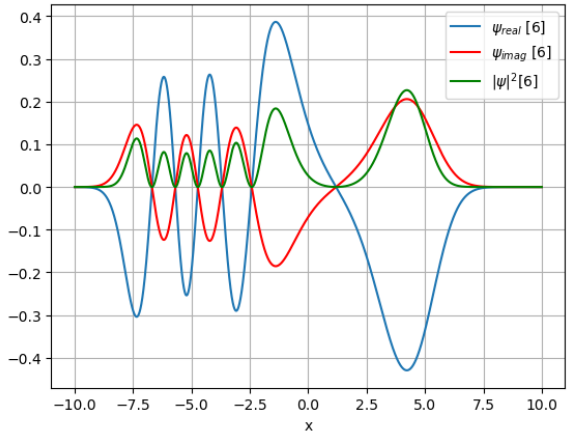

ψ_{real} [1]
ψ_{imag} [1]
$|\psi|^2$[1]
x

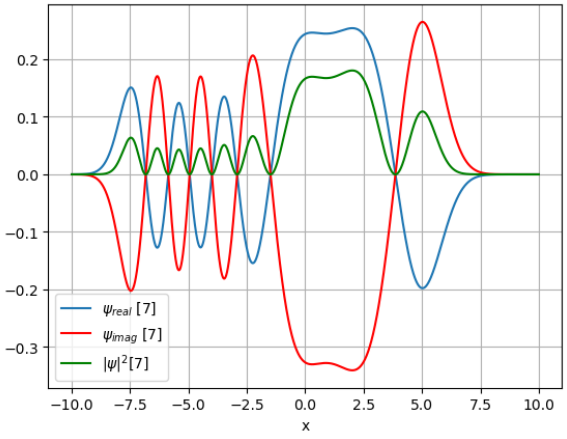

ψ_{real} [2]
ψ_{imag} [2]
$|\psi|^2$[2]
x

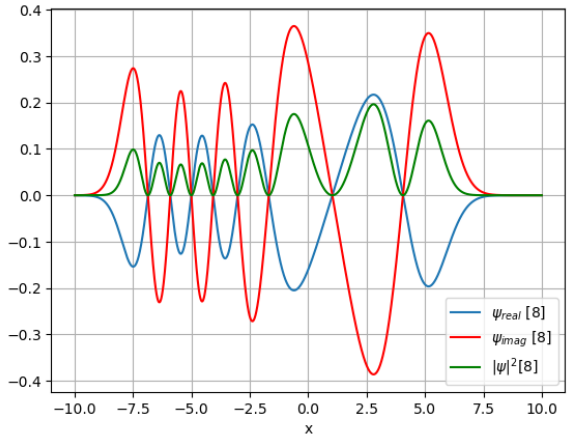

ψ_{real} [3]
ψ_{imag} [3]
$|\psi|^2$[3]
x

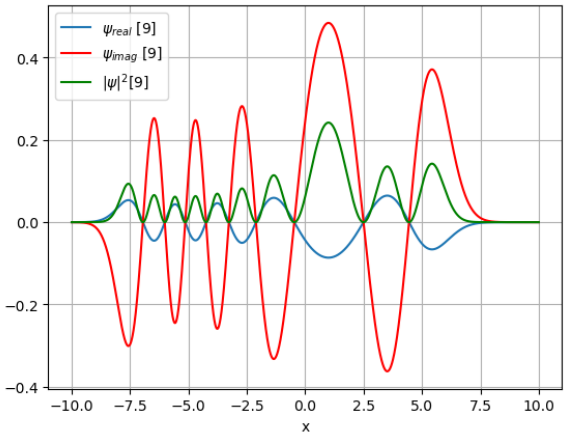

ψ_{real} [4]
ψ_{imag} [4]
$|\psi|^2$[4]
x


ψ_{real} [5]
ψ_{imag} [5]
$|\psi|^2$[5]
x


ψ_{real} [6]
ψ_{imag} [6]
$|\psi|^2$[6]
x


ψ_{real} [7]
ψ_{imag} [7]
$|\psi|^2$[7]
x





Confronto autovalori e grafici delle autofunzioni di Task A e Task B, caso $b \neq 0$

Dalle tabelle sotto riportate gli autovalori calcolati con i due diversi algoritmi sono uguali tra loro entro la seconda cifra decimale dopo la virgola per quanto riguarda i primi 2 autovalori, passando alla prima cifra per i successivi. Dunque i due algoritmi di calcolo possono essere usati indistintamente esclusivamente se si vuole lavorare con una precisione che ricade sulla prima o seconda cifra decimale. Si notino inoltre i grafici delle autofunzioni che mostrano, tra un metodo e l'altro, densità di probabilità pressochè identiche tra loro, per n fissato.

```
1  −2.05673756327815    −0.970696101005160    7.020608247330527E−002
   1.06203757689582     1.99140177370646      2.74224523314842
3  2.82382626665276     3.35198348408085      3.54537653383522
   3.94605308491373
```

Listing 14: Autovalori Task A

```
1  −2.05689303626143    −0.972779749556995    6.039337150382251E−002
   1.03456009321336     1.93707830546628      2.73801794114073
3  2.74825698139023     3.32621548035866      3.48069110957680
   3.87890749821765
```

Listing 15: Autovalori Task B

RISULTATI E CONCLUSIONI PER LA TASK B

Sia per il caso $b = 0$ sia per quello $b \neq 0$ è possibile trarre le stesse conclusioni discusse per la Task A. Nel primo caso riguardano la simmetria del problema e la presenza degli stati in entrambe le buche a causa degli autovalori uguali a due a due entro un certo numero di cifre decimali; in particolar modo le prime coppie presentano autovalori uguali entro la terza cifra decimale. Le successive coppie di autovalori però non presenteranno autovalori del tutto uguali,questo argomento verrà trattato nel paragrafo successivo. Per quanto riguarda $b \neq 0$ anche in questo caso non essendoci una simmetria del potenziale si perde completamente la degenerazione dell'energia.

CONCLUSIONI

In conclusione si sono determinati gli autostati dell'equazione di Schrödinger per entrambi i tipi di potenziale considerati, con due algoritmi differenti, giungendo risultati validi in entrambi i casi. I due algoritmi sono stati poi confrontati e propongono risultati compatibili tra loro entro le prime cifre decimali. E' quindi possibile usare un metodo piuttosto che un altro pur rimanendo all'interno dell'errore stabilito a priori.
Infine durante la ricerca dei coefficienti del potenziale è stato osservato come i primi autovalori sembrino essere uguali a coppie, diversificandosi crescendo lungo la barriera che separa le due buche, in questo caso il sistema sembrerebbe comportarsi come due oscillatori armonici disaccoppiati.

Di seguito vengono riportati ineteramente tutti i codici utilizzati nella soluzione del problema

TASK A

```
2
PROGRAM taskA
4 IMPLICIT NONE
INCLUDE 'mkl_lapack.fi'
6 !
DOUBLE PRECISION:: A,B,a1,b1,c1,d1
8 INTEGER :: N
INTEGER :: K,I,J,T,C,Q,R
10 DOUBLE PRECISION :: L,h,S
DOUBLE PRECISION,DIMENSION(:),ALLOCATABLE :: D,X
12 DOUBLE PRECISION,DIMENSION(:),ALLOCATABLE :: E
DOUBLE PRECISION,DIMENSION(:),ALLOCATABLE :: V,v1
14 !
CHARACTER(LEN=*),PARAMETER :: format='(E15.5,200(",",E15.5))'
16 CHARACTER,PARAMETER :: JOBZ = 'V',RANGE = 'I'

18
DOUBLE PRECISION :: VL = 0,VU = 20
20 INTEGER :: IL,IU,UB
DOUBLE PRECISION :: ABSTOL
22 INTEGER :: M, INFO, LDZ
DOUBLE PRECISION,DIMENSION(:),ALLOCATABLE :: W
24 DOUBLE PRECISION,DIMENSION(:,:),ALLOCATABLE :: Z
DOUBLE PRECISION,DIMENSION(:),ALLOCATABLE :: WORK
26 INTEGER,DIMENSION(:),ALLOCATABLE :: IWORK
INTEGER,DIMENSION(:),ALLOCATABLE :: IFAIL
28
CHARACTER(LEN=*), PARAMETER :: inputA = 'inputA.txt'
30 OPEN(UNIT=7, FILE=inputA)
    READ(7,*) A,B,N,IL,IU,a1,b1,c1,d1
32     CLOSE(7)
!
34 !
! definizione dei vettori che verranno utilizzati nella subroutine.
36 !
!
38 ALLOCATE(D(1:N))
ALLOCATE(X(1:N))
40 ALLOCATE(E(1:N-1))
!
42 L = B-A          !larghezza della doppia buca di potenziale
h = L/dble(N-1)          !distanza tra i punti della griglia
44 S = A
X(1) = A
46 X(N) = B
```

```
DO K=2,N-1          !creazione della griglia
     S = S+h
     X(K) = S
END DO
DO I=1,N          !creazione della diagonale principale della matrice
     D(I) = 1/h**2+(a1 + b1*X(I) + c1*(X(I)**2) + d1*(X(I)**4))/2
END DO
     E = (/(-dble(0.5)/h**2,J=1,N-1)/)          !diagonale secondaria della matrice
!
!
! Utilizzo della subroutine per il calcolo di autovalori e autovettori.
!
!
M=IU-IL+1
LDZ = N
UB = IU + 5       !Per avere un upper bound da definire in precedenza
!ALLOCATE(Z(LDZ,1:UB))
ALLOCATE(Z(LDZ,max(1,M)))
ALLOCATE(W(1:N))
ALLOCATE(WORK(1:5*N))
ALLOCATE(IWORK(1:5*N))
ALLOCATE(IFAIL(1:N))
ABSTOL = 2 * DLAMCH('S')
CALL dstevx(JOBZ,RANGE,N,D,E,VL,VU,IL,IU,ABSTOL,M,W,Z,LDZ,WORK,IWORK,IFAIL,INFO)
ALLOCATE(V(1:M))
V = (/(W(C),C=1,M)/)          !autovalori
IF(M .NE. (IU-IL+1)) THEN
     PRINT *, "Il numero di valori trovati non corrisponde a quello cercato."
END IF
!
!
!    i file di output del programma.
!
!
OPEN(23,FILE='autovalori.out')
WRITE(23,*) V
CLOSE(23)

!
OPEN(27,FILE='autovettoritaskA.out')

DO Q=1,LDZ
WRITE(27,FMT=format) (Z(Q,R), R=1,M)
end do
CLOSE(27)
END PROGRAM taskA
```

TASK B, MAIN PROGRAM, POTENZIALE ARMONICO

```
PROGRAM oscarmB
   USE fftw
    USE modulozheevx
    IMPLICIT NONE
!
    DOUBLE PRECISION :: A,B,L,deltax,deltaq,O,h
    INTEGER :: IL,IU,N,K,i,T,C,S,P,g
    DOUBLE PRECISION, PARAMETER :: pi=ACOS(-1.0)
    DOUBLE COMPLEX,DIMENSION(:), ALLOCATABLE :: V,V_T
    DOUBLE PRECISION,DIMENSION(:), ALLOCATABLE :: X,Q,W
    DOUBLE COMPLEX,DIMENSION(:,:), ALLOCATABLE :: MATRIXA
    CHARACTER(LEN=*), PARAMETER :: inputB1 = 'inputB1.txt'
    CHARACTER(LEN=*),PARAMETER :: format='(E15.5,200(",",E15.5))'

    OPEN(UNIT=35, FILE=inputB1) ! Lettura del file di input.


    READ(35,*) A,B,N,IL,IU,K
    CLOSE(35)
        ALLOCATE(X(1:N+1))
        ALLOCATE(Q(1:N+1))
        ALLOCATE(V(1:N+1))
        ALLOCATE(V_T(1:N+1))

!
! Definizione dei vettori utili e del potenziale da utilizzare.
!
        L = B-A
        deltax = L/dble(N)
        deltaq = 2*pi/L
        X(1) = -L/2
        Q(1) = -(pi*N)/L
        DO i = 1,N
            X(i+1) = X(i) + deltax
        END DO
        DO g = 1,N
            Q(g+1) = Q(g) + deltaq
        END DO
        V = X**2

        CALL fttw_(L,N,deltax,V,V_T)
        OPEN(36,FILE='V_T.txt')
        OPEN(37,FILE='DeltaK.txt')
        DO P=1,N+1
            WRITE(36,*) DREAL(V_T(P))
            WRITE(37,*) Q(P)
        END DO
        CLOSE(36)
        CLOSE(37)
```

```
50
52 ! Costruzione della matrice da diagonalizzare
        ALLOCATE(MATRIXA(1:2*K+1,1:2*K+1))
54      MATRIXA = 0.0
        DO S=-K,K
56          DO C=S,K
                MATRIXA(K+S+1,K+C+1) = V_T(N/2+1+S-C)*0.5
58          END DO
            MATRIXA(K+S+1,K+S+1) = MATRIXA(K+S+1,K+S+1) + 0.5*(S*2*pi/L)**2
60      END DO

62
       CALL zheevx_(MATRIXA,2*K+1,IU,IL)
64

66 END PROGRAM oscarmB
```

TASK B, MAIN PROGRAM, DOPPIA BUCA

```
1  PROGRAM oscarmB
       USE fftw
3      USE modulozheevx
       IMPLICIT NONE
5  !
       DOUBLE PRECISION :: A,B,L,deltax,deltaq,h,a1,b1,c1,d1
7      INTEGER :: IL,IU,N,K,i,T,C,S,P,g
       DOUBLE PRECISION, PARAMETER :: pi=ACOS(-1.0)
9      DOUBLE COMPLEX,DIMENSION(:), ALLOCATABLE :: V,V_T
       DOUBLE PRECISION,DIMENSION(:), ALLOCATABLE :: X,Q,W
11     DOUBLE COMPLEX,DIMENSION(:,:), ALLOCATABLE :: MATRIXA
       CHARACTER(LEN=*), PARAMETER :: inputB1 = 'inputB2.txt'
13     CHARACTER(LEN=*),PARAMETER :: format='(E15.5,200(",",E15.5))'


15



17     OPEN(UNIT=35, FILE=inputB1) ! Lettura del file di input.
       READ(35,*) A,B,N,IL,IU,K,a1,b1,c1,d1
19     CLOSE(35)
       ALLOCATE(X(1:N+1))
21     ALLOCATE(Q(1:N+1))
       ALLOCATE(V(1:N+1))
23     ALLOCATE(V_T(1:N+1))

25 !
   ! Calcolo dei vettori utili e del potenziale da utilizzare.
27 !
       L = B-A
```

```
29      deltax = L/dble(N)
        deltaq = 2*pi/L
31      X(1) = -L/2
        Q(1) = -(pi*N)/L
33      DO i = 1,N
        X(i+1) = X(i) + deltax
35      END DO
        DO g = 1,N
37      Q(g+1) = Q(g) + deltaq
        END DO
39      V = a1+b1*X+ c1*(X**2)+d1*(X**4)
  ! esecuzione della trasformata
41      CALL fttw_(L,N,deltax,V,V_T)
        OPEN(36,FILE='V_T.txt')
43      OPEN(37,FILE='DeltaK.txt')
        DO P=1,N+1
45      WRITE(36,*) DREAL(V_T(P))
        WRITE(37,*) Q(P)
47      END DO
        CLOSE(36)
49      CLOSE(37)

51  ! Costruzione della matrice da diagonalizzare
        ALLOCATE(MATRIXA(1:2*K+1,1:2*K+1))
53      MATRIXA = 0.0
        DO S=-K,K
55      DO C=S,K
        MATRIXA(K+S+1,K+C+1) = V_T(N/2+1+S-C)*0.5
57      END DO
        MATRIXA(K+S+1,K+S+1) = MATRIXA(K+S+1,K+S+1) + 0.5*(S*2*pi/L)**2
59      END DO

61      CALL zheevx_(MATRIXA,2*K+1,IU,IL)


63
  END PROGRAM oscarmB
```

MODULO PER IL CALCOLO DELLA FFTW

```
MODULE fftw
    IMPLICIT NONE
!
    CONTAINS
!
        SUBROUTINE fttw_(L,N,deltax,A,ff3)
            IMPLICIT NONE
            INCLUDE 'fftw3.f'
            DOUBLE COMPLEX, DIMENSION(1:N+1), INTENT(IN) :: A
            DOUBLE COMPLEX, DIMENSION(1:N+1) :: ff1,ff2,ff4,ff5,ff6
            DOUBLE COMPLEX, DIMENSION(1:N+1), INTENT(OUT) :: ff3

            INTEGER(KIND=8) :: plan1,plan2
            DOUBLE PRECISION, INTENT(IN) :: L,deltax
            INTEGER, INTENT(IN) :: N
!
                ff1(N/2+2:) = A(1:N/2)
                ff1(1:N/2+1) = A(N/2+1:)
!
                CALL dfftw_plan_dft_1d(plan1,N+1,ff1,ff2,FFTW_forward,fftw_estimate)
                CALL dfftw_execute(plan1,ff1,ff2)
                CALL dfftw_destroy_plan(plan1)
                ff2 = ff2/dble(N+1)
                ff3(N/2+1:) = ff2(1:N/2+1)
                ff3(1:N/2) = ff2(N/2+2:)
!
                ff4(N/2+2:) = ff3(1:N/2)
                ff4(1:N/2+1) = ff3(N/2+1:)

                CALL dfftw_plan_dft_1d(plan2,N+1,ff4,ff5,FFTW_backward,fftw_estimate)
                CALL dfftw_execute(plan2,ff4,ff5)
                CALL dfftw_destroy_plan(plan2)
                ff6(N/2+1:) = ff5(1:N/2+1)
                ff6(1:N/2) = ff5(N/2+2:)


        END SUBROUTINE fttw_
!
END MODULE fftw
```

MODULO PER LA DIAGONALIZZAZIONE

```
MODULE modulozheevx
    IMPLICIT NONE
!
    CONTAINS
        SUBROUTINE zheevx_(MATRIXA,N,IU,IL)
        IMPLICIT NONE
        INCLUDE 'mkl_lapack.fi'
!
        INTEGER,INTENT(IN) :: N,IU,IL
        DOUBLE COMPLEX,DIMENSION(N,N),INTENT(IN) :: MATRIXA
        DOUBLE PRECISION,DIMENSION(:),ALLOCATABLE :: EV
        INTEGER :: Y,R
!
        CHARACTER,PARAMETER :: JOBZ = 'V',RANGE = 'I',UPLO = 'U'
        DOUBLE PRECISION :: VL = 0,VU = 10
        INTEGER :: UB
        DOUBLE PRECISION :: ABSTOL
        INTEGER :: INFO, LDZ, LDA, LWORK, M
        DOUBLE PRECISION,DIMENSION(1:N) :: W
        DOUBLE COMPLEX,DIMENSION(:,:),ALLOCATABLE :: Z
        DOUBLE COMPLEX,DIMENSION(1:2*N) :: WORK
        INTEGER,DIMENSION(1:5*N) :: IWORK
        DOUBLE PRECISION,DIMENSION(1:7*N) :: RWORK
        INTEGER,DIMENSION(1:N) :: IFAIL
!
        CHARACTER(LEN=*),PARAMETER :: format='(E15.5,200(",",E15.5))'
!
! Utilizzo della subroutine Zheevx
!
        LWORK = 2*N
        LDA = N
        LDZ = N
        UB = IU + 5      !upper bound
        ABSTOL = 2 * DLAMCH('S')
        ALLOCATE(Z(LDZ,1:UB))
        CALL zheevx(JOBZ,RANGE,UPLO,N,MATRIXA,LDA,VL,VU,IL,IU,ABSTOL,M,W,Z,LDZ,WORK,LWORK,
    RWORK,IWORK,IFAIL,INFO)

        ALLOCATE(EV(1:M))
        EV = W(1:M)

!
! Scrittura dei file di output.
!
        OPEN(23,FILE='autovalori.out')
        WRITE(23,*) EV
        CLOSE(23)
!
```

```
49          OPEN(27,FILE='autovettori.out')
            DO Y=1,LDZ
51              WRITE(27,FMT=format) (REAL(Z(Y,R)), R=1,M)
            END DO
53          CLOSE(27)
  !
55          OPEN(28,FILE='imm.out')
            DO Y=1,LDZ
57              WRITE(28,FMT=format) (AIMAG(Z(Y,R)), R=1,M)
            END DO
59          CLOSE(28)
  !
61          END SUBROUTINE zheevx_
  !
63 END MODULE modulozheevx
```

APPENDICE A

Parametri di inout per la routine Dstevx utilizzata nella task A:

- `JOBZ`: parametro input di tipo carattere; se JOBZ = 'V', come in questo caso, fa si che vengano calcolati autovalori ed autovettori. Se JOBZ = 'N' fa si che vengano calcolati solamente gli autovalori.

- `RANGE`: parametro inpur di tipo carattere; se RANGE = 'I', come in questo caso, calcola gli autovalori dal IL-esimo al IU-esimo. Se RANGE = 'V', calcola gli autovalori nell'intervallo semiaperto (VL, VU]. Se RANGE = 'A' calcola tutti gli autovalori.

- `N`: parametro input di tipo intero; rappresenta l'ordine della matrice.

- `D`: array di input ed output in doppia precisione di ordine N; in input rappresenta gli N elementi della diagonale principale di Q. In output potrebbe essere moltiplicato per una costante scelta al fine di evitare l'over/underflow durante il calcolo degli autovalori.

- `E`: array di input ed output in doppia precisione di ordine (N-1); in input rappresenta gli (N-1) elementi di una delle due simmetriche diagonali secondarie. In output potrebbe essere moltiplicato per una costante scelta al fine di evitare l'over/underflow durante il calcolo degli autovalori.

- `VL`: parametro di input in doppia precisione; poiché RANGE = 'I' è possibile associare a VL un qualsiasi valore reale in doppia precisione (per via del RANGE che esclude il suo utilizzo). Se RANGE = 'V' rappresenta l'estremo inferiore dell'intervallo a cui appartengono gli autovalori. VL $<$ VU .

- `VU`: parametro di input in doppia precisione; poiché RANGE = 'I' è possibile associare a VU un qualsiasi valore reale in doppia precisione (per via del RANGE che esclude il suo utilizzo). Se RANGE = 'V' rappresenta l'estremo superiore dell'intervallo a cui appartengono gli autovalori. VL $<$ VU .

- `IL`: parametro di input di tipo intero; poiché RANGE = 'I', IL rappresenta l'indice del più piccolo tra gli autovalori che devono essere calcolati. $1 \leq$ IL $\leq$ IU $\leq$ N se N > 0 ; IL = 1 , IU = 0 se N = 0. Se RANGE = 'A' oppure 'V' attribuire ad IL un qualunque valore intero.

- `IU`: parametro di input di tipo intero; poiché RANGE = 'I', IU rappresenta l'indice del più grande tra gli autovalori che devono essere calcolati. $1 \leq$ IL $\leq$ IU $\leq$ N se N > 0 ; IL = 1 , IU = 0 se N = 0. Se RANGE = 'A' oppure 'V' attribuire ad IU un qualunque valore intero.

- `ABSTOL`: parametro di input in doppia precisione; rappresenta la tolleranza dell'errore assoluto associato a ciascun autovalore. Un autovalore è definito convergente quando risiede in un intervallo [a,b] di ampiezza minore o uguale ad: (ABSTOL + EPS) * max($|a|, |b|$), dove EPS è la precisione di macchina. Se ABSTOL risulta minore o uguale a zero allora sarà sostituito da: EPS *$|T|$, dove $|T|$ è la norma-1 della matrice tridiagonale Q. Se INFO > 0 , valore che indica la non convergenza di alcuni autovettori, gli autovalori saranno calcolati con più precisione ponendo: ABSTOL = 2 * DLAMCH('S') . La funzione DLAMCH, presa dalla libreria `LAPACK`, migliora la precisione degli autovalori calcolati dalla subroutine.

- `M`: parametro di output di tipo intero; rappresenta il numero totale degli autovalori calcolati. $0 \leq$ M $\leq$ N. Poiché RANGE= 'I' si ha che M = IU - IL + 1. Se RANGE = 'A' si ha che M = N.

- `W`: array di output in doppia precisione di dimensione N; i suoi primi M elementi sono gli M autovalori cercati ed in ordine crescente.

- `Z`: array di output in doppia precisione con dimensione (LDZ, max(1,M)). Poiché JOBZ = 'V', se INFO = 0, le prime M colonne di Z contengono gli autovettori ortonormali della matrice Q, corrispondenti agli autovalori selezionati, con la i-esima colonna di Z che rappresenta gli autovettori contenuti in W(i). Se un autovettore fallisce la sua convergenza (INFO > 0), allora quella colonna di Z contiene la più recente approssimazione dell'autovettore; il suo indice è contenuto in IFAIL. Se JOBZ = 'N' attribuire a Z un qualunque array in doppia precisione.
  Attenzione: assicurarsi che almeno max(1,M) colonne siano presenti nell'array Z. Se RANGE = 'V', l'esatto valore di M è ignoto a priori e deve essere usato un estremo superiore.

- `LDZ`: parametro di input di tipo intero; rappresenta la dimensione dell'array Z. LDZ $\geq$ 1 e poiché JOBZ = 'V' si ha che LDZ $\geq$ max(1,N).

- `WORK`: array di output in doppia precisione; ha dimensione 5*N.

- `IWORK`: array di output di tipo intero; ha dimensione 5*N.

- `IFAIL`: array di output di tipo intero con dimensione N; poiché JOBZ = 'V', se INFO = 0 allora i primi M elementi di IFAIL sono 0. Se INFO > 0 allora contiene gli indici degli autovettori che non hanno raggiunto la convergenza. Se JOBZ = 'N' attribuire ad IFAIL un qualunque array intero di dimensione N.

- `INFO`: parametro di output di tipo intero; se INFO = 0 non sono stati commessi errori; se INFO = -i < 0 allora l'i-esimo argomento presenta un valore non corretto; se INFO = i > 0 allora un numero i di autovettori non ha raggiunto la convergenza e i loro indici sono immagazzinati nell'array IFAIL.

APPENDICE B

Parametri di input per la routine zheevx usata nella task B:

- `JOBZ`: parametro input di tipo carattere; se JOBZ = 'V', come in questo caso, fa si che vengano calcolati autovalori ed autovettori. Se JOBZ = 'N' fa si che vengano calcolati solamente gli autovalori.

- `RANGE`: parametro input di tipo carattere; se RANGE = 'I', come in questo caso, calcola gli autovalori dal IL-esimo al IU-esimo. Se RANGE = 'V', calcola gli autovalori nell'intervallo semiaperto (VL, VU]. Se RANGE = 'A' calcola tutti gli autovalori.

- `UPLO`: parametro input di tipo carattere; se UPLO = 'U', viene immagazzinato il triangolo superiore di A. Se UPLO = 'L' viene immagazzinato il triangolo inferiore di A.

- `N`: parametro input di tipo intero; rappresenta l'ordine della matrice. N $\geq$ 0.

- `A`: array di input ed output, complesso in doppia precisione di dimensioni (LDA,N); in input rappresenta la matrice hermitiana A. Se UPLO = 'U', la parte triangolare superiore di dimensioni (N,N) di A contiene la parte triangolare superiore della matrice hermitiana A. Se UPLO = 'L', la parte triangolare inferiore

di dimensioni (N,N) di A contiene la parte triangolare inferiore della matrice hermitiana A. In output viene distrutto il triangolo inferiore (se UPLO = 'L') oppure il triangolo superiore (se UPLO = 'U') di A, con anche la diagonale.

- `LDA`: parametro di input di tipo intero; rappresenta il numero di righe della matrice A. LDA $\geq$ max(1,N).

- `VL`: parametro di input in doppia precisione; poiché RANGE = 'I' è possibile associare a VL un qualsiasi valore reale in doppia precisione (per via del RANGE che esclude il suo utilizzo). Se RANGE = 'V' rappresenta l'estremo inferiore dell'intervallo a cui appartengono gli autovalori. VL < VU .

- `VU`: parametro di input in doppia precisione; poiché RANGE = 'I' è possibile associare a VU un qualsiasi valore reale in doppia precisione (per via del RANGE che esclude il suo utilizzo). Se RANGE = 'V' rappresenta l'estremo superiore dell'intervallo a cui appartengono gli autovalori. VL < VU .

- `IL`: parametro di input di tipo intero; poiché RANGE = 'I', IL rappresenta l'indice del più piccolo tra gli autovalori che devono essere calcolati. $1 \leq$ IL $\leq$ IU $\leq$ N se N > 0 ; IL = 1 , IU = 0 se N = 0. Se RANGE = 'A' oppure 'V' attribuire ad IL un qualunque valore intero.

- `IU`: parametro di input di tipo intero; poiché RANGE = 'I', IU rappresenta l'indice del più grande tra gli autovalori che devono essere calcolati. $1 \leq$ IL $\leq$ IU $\leq$ N se N > 0 ; IL = 1 , IU = 0 se N = 0. Se RANGE = 'A' oppure 'V' attribuire ad IU un qualunque valore intero.

- `ABSTOL`: parametro di input in doppia precisione; rappresenta la tolleranza dell'errore assoluto associato a ciascun autovalore. Un autovalore è definito convergente quando risiede in un intervallo [a,b] di ampiezza minore o uguale ad: (ABSTOL + EPS) * max($|a|, |b|$), dove EPS è la precisione di macchina. Se ABSTOL risulta minore o uguale a zero allora sarà sostituito da: EPS *$|T|$, dove $|T|$ è la norma-1 della matrice tridiagonale ottenuta A ad una forma tridiagonale. Se INFO > 0 , valore che indica la non convergenza di alcuni autovettori, gli autovalori saranno calcolati con più precisione ponendo: ABSTOL = 2 * DLAMCH('S') . La funzione DLAMCH, presa dalla libreria `LAPACK`, migliora la precisione degli autovalori calcolati dalla subroutine.

- `M`: parametro di output di tipo intero; rappresenta il numero totale degli autovalori calcolati. $0 \leq$ M $\leq$ N. Poiché RANGE= 'I' si ha che M = IU - IL + 1. Se RANGE = 'A' si ha che M = N.

- `W`: array di output in doppia precisione di dimensione N; i suoi primi M elementi sono gli M autovalori cercati ed in ordine crescente.

- `Z`: array di output complesso in doppia precisione con dimensione (LDZ, max(1,M)). Poiché JOBZ = 'V', se INFO = 0, le prime M colonne di Z contengono gli autovettori ortonormali della matrice A, corrispondenti agli autovalori selezionati, con la i-esima colonna di Z che rappresenta gli autovettori contenuti in W(i). Se un autovettore fallisce la sua convergenza (INFO > 0), allora quella colonna di Z contiene la più recente approssimazione dell'autovettore; il suo indice è contenuto in IFAIL. Se JOBZ = 'N' attribuire a Z un qualunque array in doppia precisione.
  Attenzione: assicurarsi che almeno max(1,M) colonne siano presenti nell'array Z. Se RANGE = 'V', l'esatto valore di M è ignoto a priori e deve essere usato un estremo superiore.

- `LDZ`: parametro di input di tipo intero; rappresenta la dimensione dell'array Z. LDZ $\geq$ 1 e poiché JOBZ = 'V' si ha che LDZ $\geq$ max(1,N).

- `WORK`: array di output complesso in doppia precisione, con dimensione (max(1,LWORK)). In uscita, se INFO = 0, WORK(1) restituisce il valore di LWORK ottimale.

- `LWORK`: parametro di input di tipo intero; rappresenta la lunghezza dell'array WORK. LWORK $\geq$ 1 quando N $\leq$ 1; negli altri casi assume valore 2*N. Per un'efficienza ottimale, LWORK $\geq$ (NB+1)*N, dove NB rappresenta il massimo del blocksize per ZHETRD e per ZUNMTR, come restituito da ILAENV. Se LWORK = -1, viene definito un workspace query; la routine calcola solamente la dimensione ottimale dell'array WORK, restituisce questo valore come prima entrata dell'array WORK e XERBLA non fornisce nessun messagio di errore relativo ad LWORK.

- `RWORK`: array di output in doppia precisione; ha dimensione 7*N.

- `IWORK`: array di output di tipo intero; ha dimensione 5*N.

- `IFAIL`: array di output di tipo intero con dimensione N; poiché JOBZ = 'V', se INFO = 0 allora i primi M elementi di IFAIL sono 0. Se INFO $>$ 0 allora contiene gli indici degli autovettori che non hanno raggiunto la convergenza. Se JOBZ = 'N' attribuire ad IFAIL un qualunque array intero di dimensione N.

- `INFO`: parametro di output di tipo intero; se INFO = 0 non sono stati commessi errori; se INFO = -i $<$ 0 allora l'i-esimo argomento presenta un valore non corretto; se INFO = i $>$ 0 allora un numero i di autovettori non ha raggiunto la convergenza e i loro indici sono immagazzinati nell'array IFAIL.