Determinazione degli autostati del potenziale di Morse con tecniche di diagonalizzazione numerica

Christian Aoufia
Alessandra Grieco

(Dated: 22 giugno 2021)

Utilizzando la libreria Fortran LAPACK, il progetto propone la risoluzione dell'equazione di Schrödinger per il potenziale di Morse attraverso la diagonalizzazione di una matrice opportuna. Verranno utilizzate sia la rappresentazione in spazio reale che in una base di onde piane, attraverso la Fast Fourier Transform (FFT) e la libreria FFTW.

INTRODUZIONE

Obiettivo del progetto è lo studio di un sistema quantistico monodimensionale su cui agisce un potenziale di Morse attraverso la determinazione numerica degli autovalori e autovettori dell'operatore hamiltoniano associato al sistema. Questo tipo di potenziale è di grande importanza in fisica atomica poiché permette di modellizzare l'interazione tra i nuclei in una molecola biatomica e ricavarne quindi i livelli vibrazionali. In questo progetto verranno determinati i primi M autovalori e autovettori della seguente equazione di Schrödinger (qui riscalata per essere adimensionale):

$$\frac{1}{2}\left[-\frac{\partial^2}{\partial x^2} + v(x)\right]\Phi(x) = E(x)\Phi(x), \tag{1}$$

dove v è il potenziale di Morse

$$v(x) = (1 - e^{-\alpha(x-x_0)})^2, \tag{2}$$

utilizzando sia la rappresentazione in spazio reale (Task A) che quella su una base di onde piane (Task B). Quest'ultima scelta di funzioni di base è particolarmente conveniente prima di tutto perché diagonalizza una parte dell'hamiltoniana (l'energia cinetica) e in secondo luogo poiché è disponibile un algoritmo molto efficiente per il metodo FFT.

TASK A

DEFINIZIONE DEL SISTEMA DA INTEGRARE NUMERICAMENTE E DESCRIZIONE DEGLI ALGORITMI SELEZIONATI

Obiettivo di questa task è risolvere l'equazione (1) in spazio reale. Per renderla implementabile in codice Fortran, riscriviamo la derivata seconda con la sua espressione alle differenze finite:

$$f'' = \frac{f(x-h) - 2f(x) + f(x+h)}{h^2}, \tag{3}$$

Utilizziamo quindi una griglia di N punti x_i dato un intervallo spaziale di lunghezza L: l'equazione di Schrödinger sulla griglia diventa, scrivendo $\Phi_k = \Phi(x_k)$:

$$-\frac{\Phi_{k-1} - 2\Phi_k + \Phi_{k+1}}{h^2} + v_k\Phi_k - = 2E\Phi_k, \tag{4}$$

dove h è l'intervallo tra punti successivi e v_k è il potenziale di Morse calcolato sui punti della griglia. La (4), raccogliendo i termini omologhi in k, diventa:

$$-\frac{1}{h^2}\Phi_{k-1} + \left(\frac{2}{h^2} + v_k\right)\Phi_k - \frac{1}{h^2}\Phi_{k+1} = 2E\Phi_k, \tag{5}$$

che è un sistema di N equazioni lineari agli autovalori nelle N incognite Φ_k, quindi accoppia i termini in k, $k-1$ e $k+1$. Questo sistema è esprimibile in forma matriciale come:

$$\mathbf{H}\Phi = 2E\Phi \tag{6}$$

dove $\Phi = (\Phi_1, \Phi_2, ..., \Phi_N)^T$ e H è una matrice NxN simmetrica e tridiagonale:

$$[\mathbf{H}]_{(i,j)} = \begin{cases} -1/h^2, & \text{se } j = i-1 \\ 2/h^2 + v_k, & \text{se } j = i \\ -1/h^2, & \text{se } j = i+1 \end{cases}. \tag{7}$$

Il problema si riduce quindi alla diagonalizzazione di $\mathbf{H}$, ovvero alla determinazione dei suoi primi M autovalori e autovettori sulla griglia x_k. Sarà inoltre variato il potenziale di Morse cambiando il parametro α nell'intervallo [0.5 , 2] e analizzata la differenza tra gli autovalori e autovettori ottenuti, anche per quanto riguarda il numero di stati legati. A tal proposito, risulta conveniente riscrivere il potenziale di Morse (sia per questa task che per la task B) traslandolo di -1 rispetto alle ordinate. In questo modo, il potenziale va a zero all'infinito, quindi gli stati legati risultano immediatamente evidenti in quanto associati ad autovalori negativi:

$$v(x) = e^{-2\alpha(x-x_0)} - 2e^{-\alpha(x-x_0)}. \tag{8}$$

La lunghezza dell'intervallo [0, L], preso totalmente positivo in quanto il potenziale di Morse non presenta particolari simmetrie, verrà scelta abbastanza grande in modo che il potenziale sia andato a zero significativamente alla fine dell'intervallo. Il parametro x_0, che rappresenta il minimo del potenziale, verrà quindi scelto in modo che sia abbastanza lontano sia da 0 che da L, altrimenti la diagonalizzazione viene compromessa dalla presenza

di una barriera di potenziale infinita che non è presente nel problema fisico. Verrà inoltre variato il numero di punti N e scelto un N abbastanza alto (compatibilmente con L) attraverso il seguente criterio di arresto: se la differenza calcolata tra gli autovalori per la griglia N e quelli per la griglia $N - N_{step}$ (dove N_{step} è il passo di variazione di N) è minore di una certa tolleranza, allora viene scelto N come numero di punti per la griglia.

IL CODICE FORTRAN

Il codice è strutturato in 4 files: i due moduli Fortran `mod_costr.f90` e `mod_result.f90`, il programma principale `main.f90` (che prende in input `input.txt`) e la parte in Python che gestisce la grafica con Matplotlib `graphics.py`. Di seguito verranno analizzati i primi tre files in particolare, che contengono l'algoritmo vero e proprio.

A. Descrizione file di input

```
40 0.00001 ! Lunghezza intervallo, tolleranza sugli autovalori
64 1024 32 6 ! Nstart, Nstop, Nstep, M (numero autovalori e autovettori da calcolare)
0.5 2.0 0.5 5 ! a_start, a_stop, a_step, x_0
'eval.txt' 'evec.txt' 'err.txt' ! nomi file di output
```

Listing 1 File di input commentato.

B. Modulo `mod_costr.f90`

Il modulo contiene due `subrountine` e una `function`. La `subroutine grid` si occupa di creare la griglia degli N punti x_k e la relativa ampiezza dell'intervallo tra due punti h, avendo come input N e L. La `function v(a,x,x_0)` prende in input i parametri del potenziale di Morse e la griglia e costruisce il potenziale stesso. La `subroutine ham_cr` invece prende in input h e il potenziale e costruisce la matrice $\mathbf{H}$ calcolando solo diagonale e diagonale superiore/inferiore poiché essa è tridiagonale e simmetrica.

C. Modulo `mod_resolve.f90`

Questo modulo, composto dalla sola `subroutine resolve`, si occupa di calcolare i primi M autovalori e autovettori di $\mathbf{H}$ utilizzando `mod_costr` e la libreria LAPACK. La `subroutine` prende in input N, M, L, le unità su cui scrivere i risultati, la tolleranza sugli autovalori e i parametri del potenziale di Morse. Dopo aver allocato tutti i vettori dichiarati, la chiamata a `grid`, `ham_cr` e il calcolo del potenziale, vengono calcolati gli autovalori tramite la routine LAPACK `DSTEBZ` [1], che è ottimizzata per matrici tridiagonali simmetriche. La routine prende in input anche una tolleranza per l'algoritmo interno, che ragionevolmente impostiamo uguale alla tolleranza del criterio di arresto.
Successivamente, poiché la routine sarà inserita in un programma `main` per essere iterata, viene calcolato la

[1] `https://www.netlib.org/lapack/explore-html/d9/db0/a18657_ga28f88843da09a0ee400daf46caaabec6.html`

differenza degli autovalori del passo corrente (divisi per 2, come si vede dalla (6)) rispetto a quelli calcolati al passo precedente (si parte con `eval_last` = 0): solo se la differenza è minore della tolleranza vengono calcolati gli autovettori corrispondenti con la routine `DSTEIN` [2] (che prende in input gli autovalori, oltre ad **H**, quindi bisogna avere cura di moltiplicare per 2 il vettore in input) e stampati autovalori e autovettori (normalizzati utilizzando la formula trapezoidale per il calcolo degli integrali) rispettivamente su `eval.txt` e `evec.txt`.

```
!CONVERGENZA
IF (err <= tol) THEN
  isdone = .true.

  WRITE(fm_eval,'(a,i2,a)') "(",M,"f10.5)"   !formati
  WRITE(fm_evec,'(a,i4,a)') "(",N,"f10.5)"

  WRITE(ueval, fm_eval) eval

  !DSTEIN(N,D,E,M,W,IBLOCK,ISPLIT,Z,LDZ,WORK,IWORK,IFAIL,INFO)
  CALL dstein(N,ham_i,ham_i_1,M,2*eval,ib,is,evec,N,work1,iwork,ifail,info)

  WRITE(uevec, fm_evec) (x(o), o = 1,N)

  DO p = 1,M !CALCOLO NORMA
      norm = 0
      DO i = 2,N
          norm =  norm + L/(2*dble(N))*(evec(i-1,p)**2+evec(i,p)**2)
      END DO
      evec(:,p) = evec(:,p) / sqrt(norm) !NORMALIZZAZIONE AUTOVETTORI

      WRITE(uevec, fm_evec) evec(:,p)
  END DO

  RETURN
END IF
```

Listing 2 Snippet di `mod_resolve.f90`

D. Programma principale `main.f90`

Il programma principale si occupa di leggere il file di input e di calcolare, tramite `resolve`, gli autovalori e gli autovettori al variare del parametro α del potenziale di Morse, e di N. Gli autovalori vengono calcolati per ogni N, mentre il calcolo degli autovettori e la stampa su files avviene, come visto prima, solo per il primo valore di N che soddisfa il criterio di arresto. Viene inoltre implementato un contatore degli steps che dà informazione su quanti steps su N sono necessari prima della convergenza. Se arriva ad N_{stop} senza convergenza, il programma stampa su terminale un messaggio di *warning*.

[2] `https://www.netlib.org/lapack/explore-html/da/dba/group__double_o_t_h_e_rcomputational_ga215c9e229f4b54fed9993f58285aba8a.html`

E. Descrizione dei file di output

Oltre ai due file presentati di seguito, `resolve` ha come output anche un file `err.txt` che tiene traccia della differenza tra gli autovalori calcolati ai vari passi. Il suo output verrà commentato nella sezione di esperimenti numerici.

```
 -0.28125   -0.03125    0.00686    0.02616    0.05617    0.09589 ! Primi M autovalori
       30         7595  ! Numero di step, N
```

Listing 3 Descrizione di `eval.txt`

Il file `evec.txt` è da leggersi insieme a `input.txt` e a `eval.txt` per associare i corrispondenti valori di α ed N. Per ogni valore di α, vengono stampate $M+1$ righe: la prima contiene la griglia x_k, le altre i primi M autovettori.

```
 0.00000   0.00527   0.01053   0.01580   0.02107   0.02634   0.03160   0.03687 [...]
  39.97893  39.98420  39.98947  39.99473  40.00000 ! griglia

 0.00000   0.00000   0.00000   0.00000   0.00000   0.00000   0.00000   0.00000 [...]
  0.00000   0.00000   0.00000   0.00000   0.00000 !primo autovettore
```

Listing 4 Descrizione di `evec.txt`

ESPERIMENTI NUMERICI

L'esperimento numerico è stato svolto con il file di input in Listing 1. La giustificazione del criterio di arresto descritto nei paragrafi precedenti viene dal fatto che la differenza tra gli autovalori cala più che esponenzialmente all'aumentare di N: il problema numerico è quindi ben condizionato e converge.

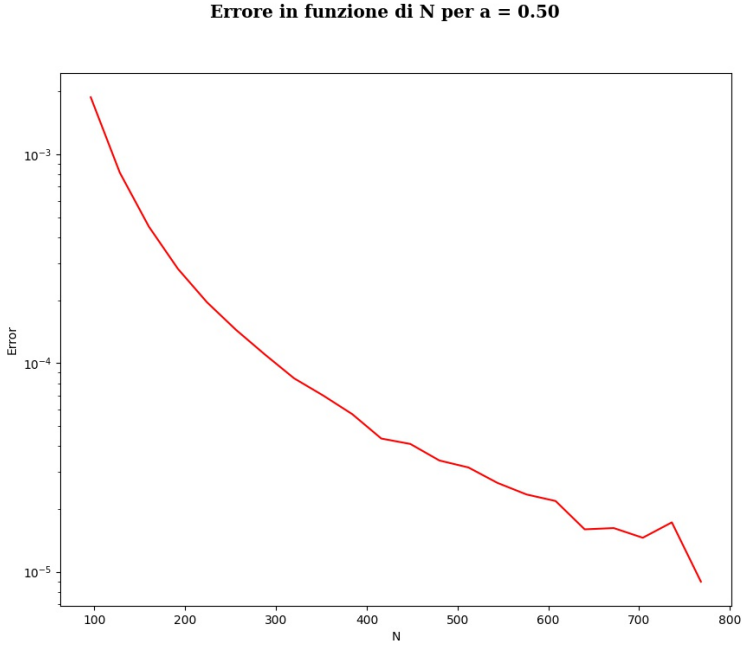


Figura 1 Grafico della differenza degli autovalori calcolati al passo N e al passo $N - N_{step}$, in funzione di N. La scala logaritmica mette in evidenza l'andamento più che esponenziale.

Di seguito riportiamo i grafici delle prime $M = 6$ autofunzioni e relativi autovalori per quattro valori di α :

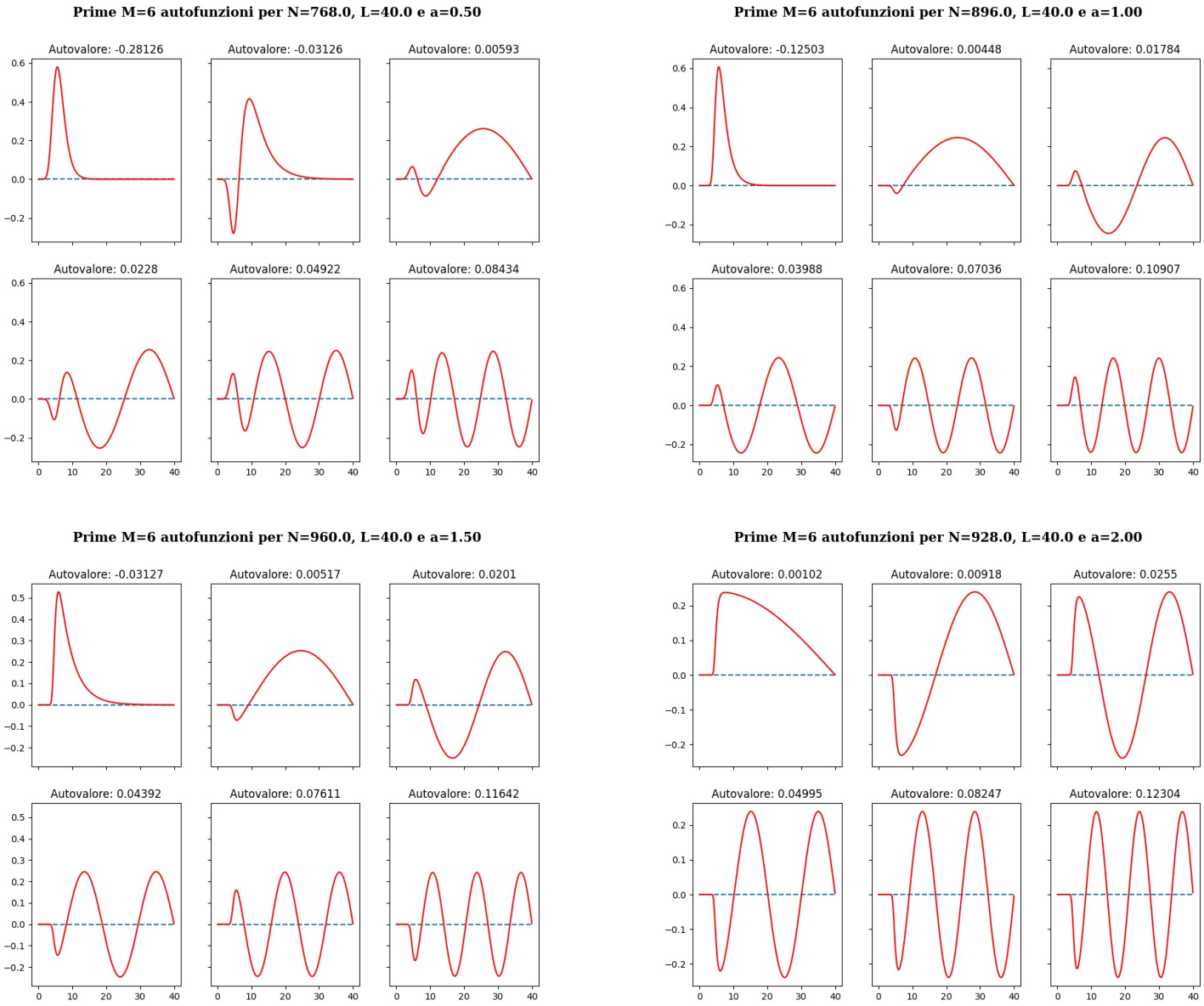


Figura 2 Autovalori e autovettori per il potenziale di Morse al variare di α, rappresentando l'equazione (1) in spazio reale.

Vista la forma del potenziale e il fatto che il problema numerico è risolto in un intervallo finito, gli stati di scattering vanno a zero diversamente a $L = 40$ che a $L = 0$ (dove vanno a zero esponenzialmente) poiché l'intervallo finito impone una barriera di potenziale infinita, che a $L = 0$ è mitigata dalla presenza della barriera del potenziale di Morse (quindi le autofunzioni vanno a zero prima di raggiungere la barriera infinita). Si nota inoltre che il numero di stati legati diminuisce al diminuire della larghezza della buca di potenziale, che è legata ad α (più è grande, meno il potenziale è largo): per $\alpha = 2$ non ci sono stati legati.

TASK B

DEFINIZIONE DEL SISTEMA DA INTEGRARE NUMERICAMENTE E DESCRIZIONE DEGLI ALGORITMI SELEZIONATI

Obbiettivo di questa seconda parte del progetto consiste nella risoluzione dell'equazione (1) nello spazio degli impulsi. In generale, per risolvere l'equazione di Schrödinger la si può proiettare su una base opportuna. In questo caso, per la sua generalità, scegliamo una base di onde piane per l'intervallo finito [0,L]:

$$u_k(x) = \frac{1}{\sqrt{L}} e^{ikx}, \tag{9}$$

dove k è un multiplo intero di $2\pi/L$. Le funzioni scelte sono un set ortonormale completo per lo spazio di Hilbert, quindi una generica autofunzione si può scrivere:

$$\phi(x) = \sum_k c_k u_k(x). \tag{10}$$

Inserendo nell'equazione (1) e moltiplicando a sinistra per $\int dx u^*_{k'}(x)$ otteniamo:

$$\sum_k c_k \left[K_{k,k'} + v_{k,k'}\right] = 2E \sum_k c_k \delta_{k,k'}, \tag{11}$$

dove $K_{k,k'}$ e $v_{k,k'}$ sono rispettivamente il valore d'aspettazione dell'energia cinetica e dell'energia potenziale tra vettori della base k e k'. Poiché le onde piane diagonalizzano $\hat{p}^2$, si avrà:

$$K_{k,k'} = k^2 \delta_{k,k'}, \tag{12}$$

e

$$v_{k,k'} = \frac{1}{L} \int_0^L dx \; v(x) e^{i(k-k')x}, \tag{13}$$

il quale può essere risolto utilizzando un algoritmo FFT, particolarmente efficiente da un punto di vista computazionale. Arriviamo allora alla formulazione finale del problema:

$$\mathbf{H} \cdot \mathbf{c} = 2E\mathbf{c}, \tag{14}$$

dove $\mathbf{c} = (\dots c_k \dots)^T$ e

$$\mathbf{H} = \begin{pmatrix} \dots & \dots & v_{k,k'} \\ \dots & k^2 + v_{k,k} & \dots \\ v_{k',k} & \dots & \dots \end{pmatrix}. \tag{15}$$

Il problema si riduce dunque al calcolo di elementi di matrice con algoritmo FFT (in particolare sarà usata la libreria fortran `FFTW3`) e alla diagonalizzazione di $\mathbf{H}$ attraverso la libreria `LAPACK`. Sarà anche qui variato il potenziale di Morse cambiando il parametro α di (8) nell'intervallo [0.5 ,2] e analizzata la differenza tra gli autovalori e autovettori ottenuti. Le stesse considerazioni fatte nel TASK A per quanto riguarda la scelta di L, x_0, N e sulla scelta dell'intervallo continuano a valere. Anche in questo caso si utilizza il criterio di arresto assoluto presentato nella precedente sezione.

IL CODICE FORTRAN

Anche in questo caso il codice è strutturato in 4 files: due moduli fortran `fourier_mod.f90` e `eigen_mod.f90`, il programma principale `main.f90` (che prende come input anche in questo caso `input.txt`) e la parte python che gestisce la grafica con Matplotlib `graphics.py` . Ignoreremo questo ultimo file nell'analisi in quanto non interessante per la discussione degli algoritmi utilizzati.

A. Descrizione file di input

```
64 1024 32 6  !Nstart, Nstop, Nstep, M (numero autovettori e autovalori da calcolare)
40 0.5 2 0.5 5 !L, astart, astop, astep, x0
0.00001 'eval.txt' 'evec.txt' 'err.txt' !Tolleranza, nomi file di output
```

Listing 5 File di input commentato.

B. Modulo `fourier_mod.f90`

Il modulo contiene 3 `subroutine` e una `function`.

La subroutine `grids` si occupa di costruire, dati in input il numero di punti N e la lunghezza dell'intervallo L, le griglie di punti nello spazio reale e in quello degli impulsi. La griglia in k viene costruita tenendo conto di come opera l'algoritmo per il calcolo della trasformata: la prima metà del vettore contiene i k positivi multipli di $2\pi/L$, la seconda metà invece contiene i k negativi immagazzinati in ordine decrescente (in valore assoluto).

La function `v` prende in input i parametri del potenziale di Morse e lo calcola secondo la griglia costruita dalla routine precedente.

La subroutine `fourier` calcola la FFT del potenziale in input (e la normalizza) utilizzando a sua volta la subroutine `dfftw_execute_dft` della libreria `FFTW3`: come accennato, le componenti di Fourier corrispondenti ai k positivi sono immagazzinate nella prima metà del vettore in output `trf`, mentre i k negativi nella seconda metà in ordine decrescente in modulo.

La subroutine `ham` crea l'hamiltoniana a partire dalla griglia in k e dalla trasformata del potenziale calcolata da `fourier`, secondo l'espressione (13) e (15):

```
DO i = 0,N-1
    DO j = 0,N-1
        IF (i .ne. j) THEN
            IF (i-j > 0) THEN
                H(i,j) = trf(i-j)
            ELSE
                H(i,j) = trf(N-(j-i))
            ENDIF
        ELSE
            H(i,j) = k(i)**2 + trf(0)
        END IF
    END DO
END DO
```

Listing 6 Procedura per la creazione dell'hamiltoniana

C. Modulo `eigen_mod.f90`

Il modulo contiene 2 `subroutine`: `resolve` si occupa di calcolare gli M autovettori e autovalori dell'hamiltoniana fornita in input. Per fare questo utilizza la procedura `zheevr` di `LAPACK`, in quanto la matrice ottenuta è in generale complessa ed hermitiana. E' bene notare che, a differenza del task A, nel task B calcoliamo anche gli autovettori ad ogni iterazione, poiché la routine utilizzata li calcola contemporaneamente agli autovalori. La procedura prende in input una tolleranza per l'algoritmo interno, che ragionevolmente impostiamo uguale alla tolleranza del criterio di arresto. La funzione degli altri parametri presenti è lasciata alla documentazione di `LAPACK`[3].

La subroutine `eigenvec`, infine, si occupa di prendere i coefficienti c_k e di calcolare la combinazione lineare di onde piane con i suddetti coefficienti, ottenendo gli effettivi autostati (che vengono normalizzati con la formula trapezoidale) dell'equazione di Schrödiger.

D. Programma principale `main.f90`

Il programma principale si occupa di preparare l'ambiente di calcolo leggendo i file di input, assegnando unità ai file di output e svolgendo i cicli di iterazione. Si occupa anche di allocare i vettori (che ovviamente hanno lunghezza variabile a ogni ciclo). L'interazione si ferma se viene raggiunto il valore in input N_{stop}, a cui il programma stampa a terminale un warning, oppure se viene soddisfatto il criterio di arresto. Anche in questo caso viene implementato un contatore di steps per monitorare in quanti passi viene raggiunta (se viene raggiunta) la convergenza.

E. Descrizione dei file di output

Oltre ai due file presentati di seguito, il programma ha anche in questo caso come output un file `err.txt` che tiene traccia della differenza tra gli autovalori calcolati ai vari passi. Il suo output verrà commentato nella sezione di esperimenti numerici.

```
1

   -0.28125   -0.03125    0.00593    0.02278    0.04919    0.08428 !Primi M autovalori
3        20        704        0.5   !Numero di step, N, a
```

Listing 7 Descrizione di `eval.txt`

Il file `evec.txt` è da leggersi insieme a `eval.txt` per associare i corrispondenti valori di α ed N. Per ogni valore di α, vengono stampate $M+1$ righe: la prima contiene la griglia x_k, le altre i primi M autovettori.

```
1   0.00000    0.00527    0.01053    0.01580    0.02107    0.02634    0.03160    0.03687 [...]
    39.97893   39.98420   39.98947   39.99473   40.00000 ! griglia

3   0.00000    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000 [...]
    0.00000    0.00000    0.00000    0.00000    0.00000   !primo autovettore
```

Listing 8 Descrizione di `evec.txt`

[3] `https://www.netlib.org/lapack/explore-html/df/d9a/group__complex16_h_eeigen_ga60dd605c63d7183a4c289a4ab3df6df6.html#ga60dd605c63d7183a4c289a4ab3df6df6`

ESPERIMENTI NUMERICI

L'esperimento numerico è stato svolto con il file di input in Listing 5. Come nel precedente task, anche qui il criterio di arresto viene giustificato dai plot degli errori in funzione di N, di cui di seguito un esempio:

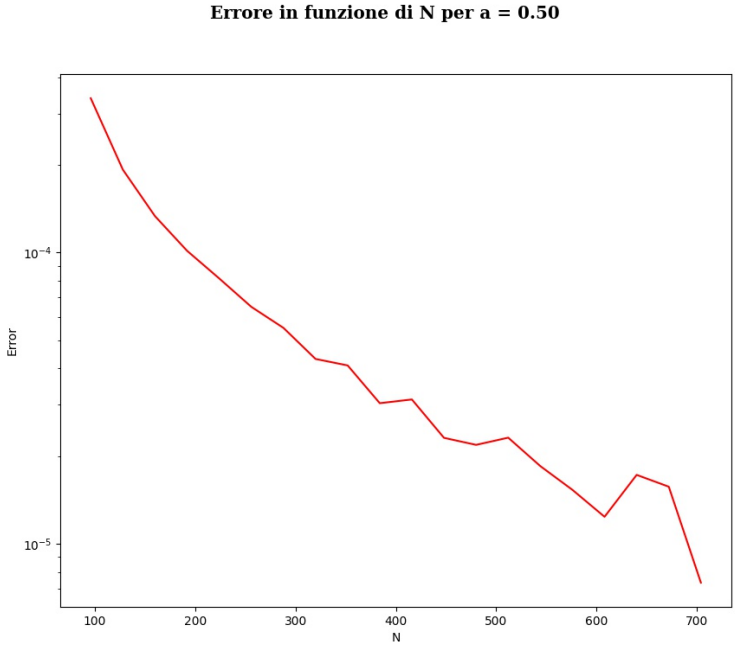
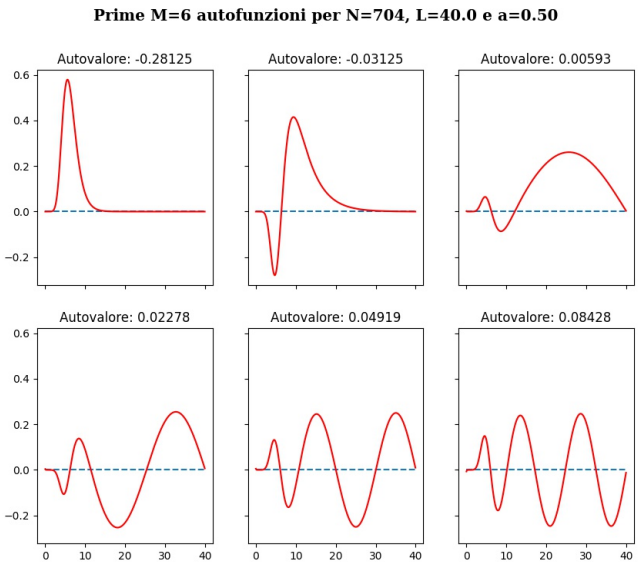
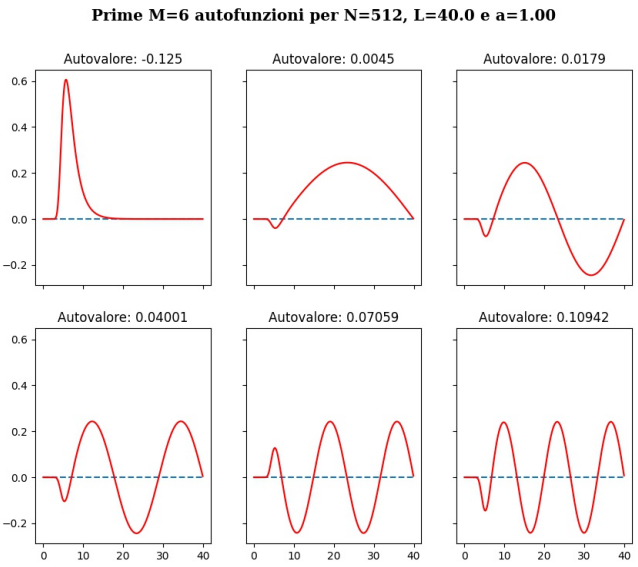


Figura 3 Grafico della differenza degli autovalori calcolati al passo N e al passo $N - N_{step}$, in funzione di N. La scala logaritmica mette in evidenza l'andamento più che esponenziale.

Il problema è quindi ben condizionato e converge (entro la tolleranza) più che esponenzialmente all'aumentare di N. Le fluttuazioni sono amplificate dalla scala logaritmica, ma sono comunque dell'ordine della tolleranza adottata sia dal criterio di arresto che dall'algoritmo di diagonalizzazione. Riportiamo anche in questo caso, prima della conclusione, i grafici delle prime M = 6 autofunzioni e relativi autovalori per quattro valori di α:





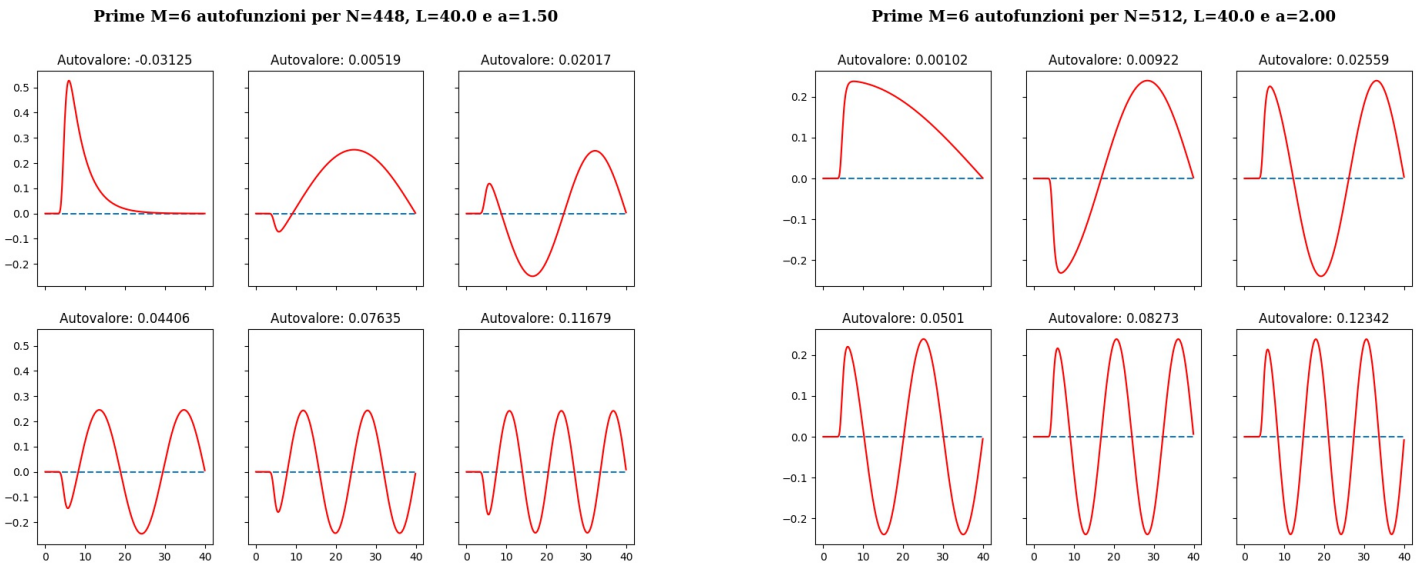


Figura 4 Autovalori e autovettori per il potenziale di Morse al variare di α, rappresentando la (1) su una base di onde piane.

Anche per questi risultati valgono le stesse considerazioni fatte nel task A.

DISCUSSIONE DEI RISULTATI E CONCLUSIONI

I risultati ottenuti per i due task sono in accordo tra loro sia per quanto riguarda gli autovalori (entro la tolleranza scelta per i due algoritmi) che gli autovettori. Alcune autofunzioni risultano differire di una fase di $e^{i\pi}$: ciò è dovuto alle differenze tra le routine utilizzate e non ha alcun effetto sulla fisica del sistema, in quanto ricordiamo che in meccanica quantistica le autofunzioni sono sempre definite a meno di una fase.

Notiamo inoltre che man mano che α aumenta, il numero di stati legati diminuisce, come già accennato. Questo è dovuto al fatto che il potenziale di Morse si stringe aumentando α e la differenza energetica tra gli stati legati aumenta, come succede in una buca di potenziale a pareti finite al diminuire della sua larghezza. Il passaggio da almeno uno stato legato a nessuno stato legato, in seguito ad altri esperimenti numerici, avviene tra $\alpha \in [1.9, 2]$. Fisicamente, questo accade quando il potenziale (in particolare la sua parte attrattiva) diventa talemente stretto da non essere più in grado di legare stati.

Per quanto riguarda le proprietà numeriche degli algoritmi utilizzati: il task B utilizza una unica routine di diagonalizzazione per il calcolo simultaneo di autovettori e autovalori (a differenza del task A). Questo, unito al fatto che nella seconda parte dell'esperimento sono maneggiate quantità generalmente complesse mentre nella prima parte solamente reali, rende i tempi di esecuzione molto maggiori. Nonostante ciò, come si nota da Figura 1 e Figura 3, la convergenza del task B avviene (a parità di tolleranza) sistematicamente per N minori, efficienza dovuta all'algoritmo FFT utilizzato e alla routine LAPACK. Inoltre, la matrice da diagonalizzare nel secondo task è una matrice in cui gli elementi diagonali dominano rispetto agli altri, fattore che contribuisce ulteriormente alla convergenza.

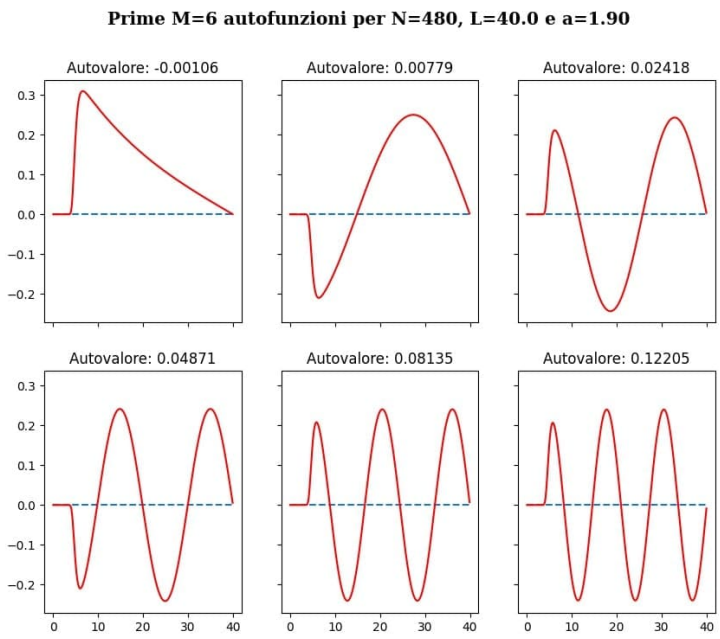


Figura 5 Autovalori ed autofunzioni per $\alpha = 1.90$, calcolati con l'algoritmo del task B. Si nota che è presente un solo stato legato, mentre per $\alpha = 2$ non ce ne sono.

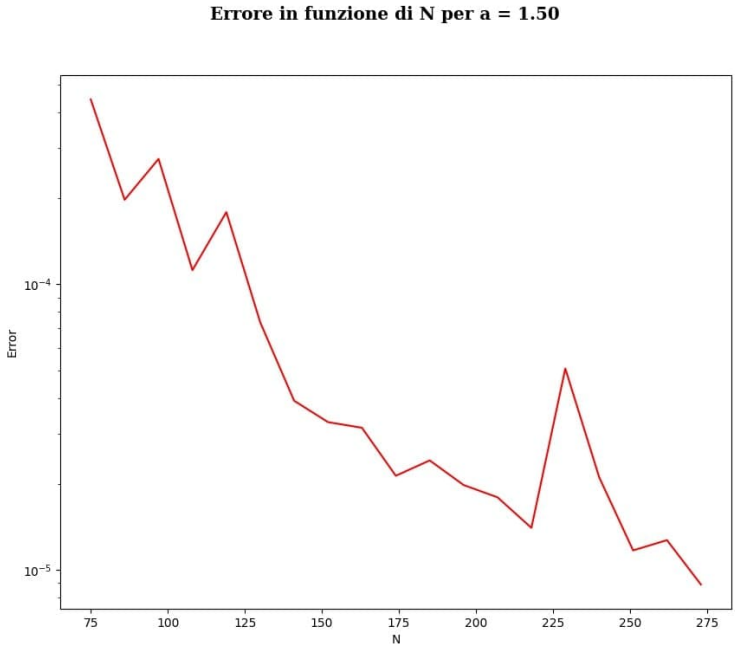


Figura 6 Risposta del task B all'introduzione di N dispari nel ciclo di iterazione.

Accenniamo inoltre alla risposta degli algoritmi all'uso di N dispari. Gli esperimenti numerici presentati sono svolti per valori di N generalemente pari, in cui sono incluse potenze del 2. Per N dispari, il task A non è particolarmente perturbato, ma la convergenza del task B, come è possibile evincere dalla Figura 6 viene significativamente affetto.

Infine, all'aumentare di L l'algoritmo di entrambi i task continua a fornire gli stessi risulati, indice che il valore di L scelto rappresenta bene la fisica del sistema, mentre al suo diminuire si va a imporre sul problema una barriera di potenziale a pareti infinite che non era inizialmente presente e di conseguenza gli autovalori cambiano.