UNIVERSITÀ DEGLI STUDI DEL PIEMONTE ORIENTALE
"AMEDEO AVOGADRO"
DIPARTIMENTO DI SCIENZE DEL FARMACO

Corso di Laurea Magistrale in Chimica e Tecnologia Farmaceutiche

TESI DI LAUREA

Sviluppo di un'interfaccia grafica per l'elaborazione dei dati ottenuti mediante simulazioni di dinamica molecolare

Relatore
Prof. Alberto Massarotti

Candidato
Bianchi Francesco

Anno Accademico 2024-25
Sessione straordinaria

Lista delle abbreviazioni

CA – Carbon alpha

CSV – Comma-Separated Values

GUI – Graphical User Interface

H-bond – Hydrogen bond

L-Torsion – Ligand torsion

MD – Molecular Dynamics

NMR – Nuclear Magnetic Resonance

NPT – Constant Number of particles, Pressure and Temperature

PL – Protein-Ligand

RMSD – Root Mean Square Deviation

RMSF – Root Mean Square Fluctuation

SSE – Secondary Structure Elements

Å – Ångström

α-helix – Alpha helix

β-sheet – Beta sheet

π-π – Pi-Pi interaction

Indice

1. Introduzione

1.1 Interazioni proteina-ligando e limiti delle descrizioni statiche

La comprensione delle interazioni tra una proteina ed un ligando rappresenta uno degli aspetti centrali della chimica farmaceutica moderna. L'efficacia di un composto bioattivo dipende infatti dalla sua capacità di riconoscere selettivamente un bersaglio biologico, di inserirsi correttamente nel sito di legame e di mantenere nel tempo un'interazione sufficientemente stabile da esercitare il proprio effetto farmacologico.[1]

Tradizionalmente, lo studio di queste interazioni si è basato su tecniche sperimentali come la cristallografia a raggi X o la risonanza magnetica nucleare (NMR), che forniscono informazioni strutturali di grande valore. Tuttavia, tali tecniche restituiscono una rappresentazione essenzialmente statica del sistema, ovvero una "istantanea" media di una struttura che, in realtà, in ambiente fisiologico è soggetta a continui movimenti e fluttuazioni.[2]

In condizioni biologiche le proteine non sono entità rigide: oscillano, respirano, si adattano alla presenza del ligando e del solvente. Allo stesso modo, il ligando può modificare la propria conformazione, esplorando diverse geometrie all'interno del sito attivo. Questi fenomeni dinamici possono influenzare in modo significativo la stabilità del complesso, la persistenza delle interazioni e, in ultima analisi, l'attività biologica del composto.[2]

Diventa quindi evidente come una descrizione puramente statica, sebbene necessaria, non sia sempre sufficiente a cogliere la complessità reale del legame proteina-ligando.[2]

1.2 L'esigenza di un approccio dinamico nello studio del legame

Per superare i limiti delle descrizioni statiche, negli ultimi decenni si è affermata la necessità di affiancare alle tecniche strutturali tradizionali strumenti in grado di descrivere l'evoluzione temporale dei sistemi biomolecolari.[2]

Dal punto di vista farmacologico, non è sufficiente sapere dove un ligando si lega, ma è altrettanto importante comprendere come si comporta nel tempo:

- Rimane stabilmente ancorato al sito di legame o mostra tendenza a spostarsi?
- Induce adattamenti conformazionali nella proteina?
- Quali interazioni sono persistenti e quali transitorie?

Rispondere a queste domande è fondamentale per interpretare correttamente le differenze di affinità tra molecole strutturalmente simili e per razionalizzare i risultati sperimentali ottenuti in vitro o in vivo.

In questo contesto si inserisce la dinamica molecolare, una tecnica di simulazione computazionale che consente di osservare il comportamento atomico di un sistema nel tempo, offrendo una visione complementare e integrativa rispetto ai dati sperimentali.[2]

1.3 La dinamica molecolare: una descrizione concettuale

La dinamica molecolare è una metodologia di simulazione che permette di seguire l'evoluzione temporale di un sistema molecolare, descrivendo il movimento degli atomi in funzione delle forze che agiscono tra di loro.[2]

Invece di osservare una singola struttura, si analizza una sequenza di configurazioni successive che descrivono come proteina, ligando e solvente si muovono e interagiscono nel tempo.

Le simulazioni di dinamica molecolare si basano sull'applicazione delle leggi della meccanica classica agli atomi del sistema. Le interazioni tra atomi sono descritte mediante modelli matematici noti come campi di forza, che permettono di stimare l'energia del sistema e di calcolare le forze responsabili del movimento atomico.[2]

Grazie all'aumento della potenza di calcolo ed al miglioramento degli algoritmi, oggi è possibile simulare sistemi biomolecolari complessi per tempi dell'ordine di centinaia di nanosecondi, ottenendo traiettorie sufficientemente lunghe da osservare fenomeni rilevanti dal punto di vista biologico.[2]

1.4 Ruolo della dinamica molecolare in chimica farmaceutica

Nel contesto della chimica farmaceutica e del drug design, la dinamica molecolare rappresenta uno strumento di grande valore per analizzare in modo approfondito il comportamento dei complessi proteina-ligando.[2]

In particolare, le simulazioni permettono di:

- Valutare la stabilità del legame nel tempo;
- Osservare eventuali fenomeni di adattamento conformazionale
- Identificare le interazioni chiave che contribuiscono alla stabilizzazione del complesso;

- Analizzare la flessibilità conformazionale sia della proteina sia del ligando.[2]

Queste informazioni non sono sempre ricavabili in modo completo da un singolo esperimento strutturale e risultano spesso più chiaramente interpretabili attraverso l'integrazione tra dati sperimentali e simulazioni.[2]

In questo senso, la dinamica molecolare non sostituisce le tecniche sperimentali, ma le integra, fornendo una chiave di lettura dinamica dei dati strutturali e contribuendo ad una visione più realistica del legame molecolare.[2]

1.5 Parametri di analisi e loro significato fisico

Dalle traiettorie generate da una simulazione di dinamica molecolare è possibile estrarre diversi parametri quantitativi, utili a descrivere il comportamento del sistema nel tempo.[2] In questo lavoro sono stati considerati in particolare i seguenti:

Stabilità strutturale (RMSD)

La deviazione media quadratica (RMSD) misura la variazione strutturale del sistema rispetto ad una configurazione di riferimento e viene comunemente utilizzata come indicatore della sua stabilità conformazionale nel tempo.

Nel caso del ligando, l'RMSD rispetto alla proteina permette di valutare se rimane stabilmente nel sito di legame o se mostra tendenze a spostamenti significativi nel tempo.

Flessibilità residuo-specifica (RMSF)

La fluttuazione media quadratica (RMSF) misura quanto ciascun residuo della proteina oscilla attorno alla propria posizione media.

Questo parametro consente di distinguere regioni strutturalmente rigide da zone più flessibili, come loop o terminali, e di valutare se la presenza del ligando influenzi localmente la dinamica della proteina.

Struttura secondaria (SSE)

Il monitoraggio della percentuale di elementi di struttura secondaria (α eliche e β foglietti) permette di verificare la conservazione del folding proteico durante la simulazione.
Il mantenimento complessivo degli elementi di struttura secondaria rappresenta un indicatore utile della conservazione del folding durante la simulazione

Interazioni proteina-ligando

L'analisi delle interazioni non covalenti (legami idrogeno, contatti idrofobici, interazioni aromatiche e ponti d'acqua) consente di identificare quali forze contribuiscono maggiormente alla stabilità del complesso e con quale persistenza temporale.[1]

Flessibilità conformazionale del ligando

L'analisi dei profili torsionali del ligando permette di valutare il grado di libertà rotazionale dei legami interni.
Un ligando meno flessibile tende a mantenere una geometria più definita nel pocket, mentre una maggiore flessibilità può favorire l'adattamento a microvariazioni del sito di legame.

1.6 Approccio comparativo nello studio dei complessi

Quando più complessi proteina-ligando vengono simulati in condizioni identiche, è possibile confrontarne quantitativamente il comportamento dinamico.
In questo lavoro, il confronto tra i sistemi è stato effettuato mediante metriche quantitative di similarità in particolare confronti basati sull'andamento dei profili temporali e sulla similarità coseno dei descrittori estratti dalle simulazioni. Questo approccio consente di valutare in modo oggettivo il grado di somiglianza tra i diversi complessi, considerando soprattutto la forma dei profili dinamici più che i valori assoluti dei singoli parametri.
L'analisi comparativa rappresenta uno strumento utile per evidenziare differenze sottili ma significative tra ligandi strutturalmente affini, fornendo una visione globale delle relazioni dinamiche tra i sistemi studiati.

2. Scopo del lavoro

Lo scopo di questo lavoro è stato analizzare e confrontare in modo quantitativo il comportamento dinamico di quattro complessi ligando-proteina (ST44, ST45, STH2, STH3), simulati mediante dinamica molecolare in condizioni omogenee, al fine di costruire un quadro comparativo utile ad evidenziare analogie e differenze tra i diversi sistemi.

L'analisi è stata impostata secondo un approccio integrato, basato sull'interpretazione congiunta di parametri strutturali, dinamici e di interazione ricavati dagli output delle simulazioni Desmond. In particolare, l'obiettivo è stato:

- Valutare la stabilità geometrica dei complessi e del ligando nel sito di legame mediante analisi RMSD;
- Analizzare la flessibilità locale della proteina e del ligando attraverso i profili RMSF;
- Monitorare la conservazione della struttura secondaria come indicatore di coerenza strutturale durante la simulazione;
- Identificare e confrontare le principali interazioni ligando-proteina persistenti, con particolare attenzione a legami idrogeno, water bridges, contatti idrofobici e interazioni aromatiche;
- Caratterizzare il comportamento conformazionale del ligando mediante l'analisi dei profili torsionali;
- Confrontare in modo quantitativo i diversi sistemi, al fine di individuare similarità e differenze nel loro comportamento dinamico complessivo.

Un ulteriore obiettivo del lavoro è stato sviluppare una procedura di analisi in Python per l'organizzazione, l'elaborazione ed il confronto sistematico dei dati ottenuti dalle simulazioni, così da rendere i diversi descrittori direttamente confrontabili in modo riproducibile.

Questo tipo di approccio assume in particolare rilevanza in chimica farmaceutica, poiché consente di collegare le caratteristiche strutturali dei ligandi al loro comportamento dinamico nel sito di legame, fornendo indicazioni utili per la comprensione dei meccanismi di interazione molecolare e per future strategie di ottimizzazione.[2]

3. Risultati e discussione

3.1 Analisi RMSD del ligando rispetto alla proteina (Lig_wrt_Protein)

Per descrivere la stabilità del posizionamento del ligando nel sito di legame lungo la simulazione di 100 ns, è stato considerato il parametro RMSD (Root Mean Square Deviation) del ligando rispetto alla proteina. In questa metrica il complesso viene prima allineato sul backbone proteico del frame di riferimento, e successivamente si calcola la deviazione media quadratica degli atomi pesanti del ligando: in questo modo l'RMSD riflette soprattutto eventuali spostamenti o ri-orientamenti del ligando nel pocket, riducendo l'influenza dei moti rigidi dell'intero complesso.

I dati (file PL-RMSD.dat) sono stati importati e analizzati tramite linguaggio Python in ambiente Colab. Per confrontare in modo sintetico i quattro sistemi (ST44, ST45, STH2, STH3) è stata calcolata la similarità coseno, che fornisce un indice di somiglianza tra due andamenti temporali: valori prossimi ad 1 indicano profili molto simili, mentre valori più bassi indicano differenze più marcate.

	ST44	ST45	STH2	STH3
ST44	1.00	0.94	0.98	0.98
ST45	0.94	1.00	0.95	0.91
STH2	0.98	0.95	1.00	0.96
STH3	0.98	0.91	0.96	1.00

Tabella 3.1. Matrice di similarità coseno (RMSD Lig_wrt_Protein, 0–100 ns).

Figura 3.1. Heatmap della similarità coseno tra le serie temporali di RMSD Lig_wrt_Protein (0–100 ns).

I valori di similarità risultano globalmente elevati **(0.91-0.98)**, indicando che, nelle condizioni simulate, il ligando mantiene in tutti i sistemi un comportamento complessivamente stabile rispetto alla proteina. In altre parole, non emergono segnali compatibili con una diffusione del ligando fuori dal sito di legame: i profili RMSD sono tra loro molto simili e suggeriscono una permanenza nel pocket lungo l'intera finestra temporale analizzata.

Le coppie **ST44-STH2** e **ST44-STH3** mostrano la similarità più alta **0.98**, suggerendo un andamento temporale praticamente sovrapponibile. La differenza più marcata riguarda **ST45** rispetto a **STH3 (0.91)**: il valore rimane alto, ma indica una dinamica relativamente meno concorde, ad esempio per oscillazioni più ampie o per un diverso assestamento del ligando nelle fasi iniziali.

Un limite del confronto tramite similarità coseno è che riassume la forma generale dei profili e può non evidenziare eventi brevi (picchi locali) o differenze confinate a specifici

intervalli della traiettoria. Per questo l'RMSD viene interpretato insieme a RMSF, SSE ed alle analisi dei contatti proteina-ligando riportate nelle sezioni successive.

3.2 Analisi RMSF (Cα)

3.2.1 Significato del parametro ed impostazione dell'analisi

Dopo aver valutato la stabilità del ligando nel sito di legame tramite l'RMSD è stata analizzata la flessibilità locale della proteina attraverso l'RMSF (Root Mean Square Fluctuation) dei carboni α.[3]

A differenza dell'RMSD, che descrive uno scostamento globale rispetto ad una struttura di riferimento, l'RMSF quantifica quanto ciascun residuo oscilla attorno alla propria posizione media durante la simulazione, mettendo in evidenza regioni rigide (tipicamente elementi di struttura secondaria) e regioni più mobili (loop e terminali).[3]

Per un residuo *i,* l'RMSF può essere espresso come:

$$RMSF(i) = \sqrt{\langle \| \mathbf{r}_i(t) - \langle \mathbf{r}_i \rangle_t \|^2 \rangle_t}$$

Dove $\mathbf{r}_i(t)$ è la posizione del Cα del residuo i al tempo t e $\langle \mathbf{r}_i \rangle_t$ è la sua posizione media lungo la traiettoria. Valori elevati indicano maggiore mobilità locale; valori bassi indicano maggiore rigidità strutturale. [3]

In questo lavoro, l'obiettivo non è stato descrivere nel dettaglio ogni singolo picco del profilo RMSF, ma verificare se i diversi ligandi (ST44, ST45, STH2, STH3), simulati nelle stesse condizioni, inducono differenze apprezzabili nella flessibilità complessiva della proteina.

3.2.2 Elaborazione dei dati e output dei prodotti

I valori di RMSF per residuo sono stati ottenuti dai file P_RMSF.dat generati da Desmond e rielaborati in Python.

Per ciascun sistema è stata estratta la colonna relativa ai Cα (CA), ottenendo un vettore di 301 valori (uno per residuo).

Per confrontare in modo scientifico i quattro profili, è stata calcolata una matrice di similarità coseno tra i vettori RMSF dei sistemi. In questo contesto, valori prossimi ad 1

indicano profili di flessibilità molto simili (stesse regioni mobili/rigide e ampiezza comparabile delle fluttuazioni).

	ST44	ST45	STH2	STH3
ST44	1.000000	0.981129	0.953150	0.963110
ST45	0.981129	1.000000	0.955727	0.961106
STH2	0.953150	0.955727	1.000000	0.960749
STH3	0.963110	0.961106	0.960749	1.000000

Tabella 3.2. Matrice di similarità (coseno) dei profili RMSF (Cα) tra i quattro complessi.

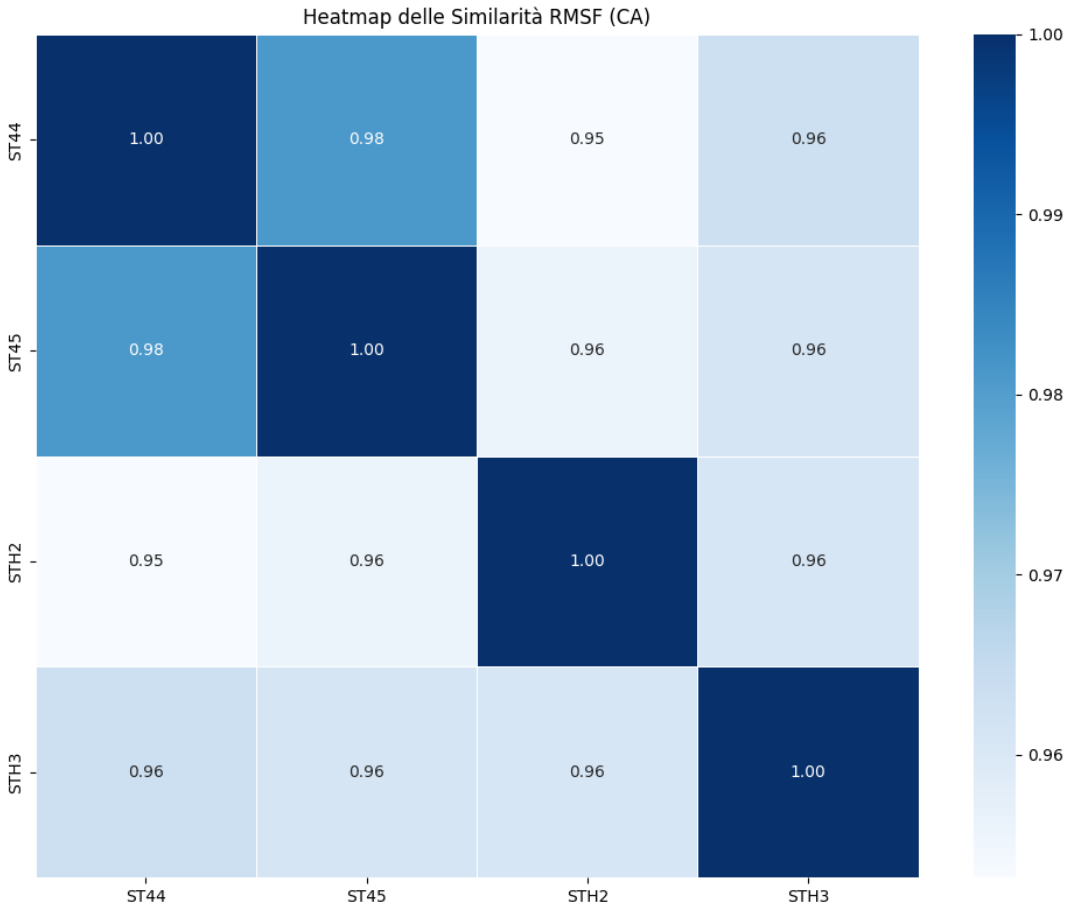

Figura 3.2. Heatmap della similarità RMSF (Cα) tra ST44, ST45, STH2 e STH3 (0–100 ns).

La matrice di similarità mostra valori elevati per tutte le coppie **(0.95-0.98)**, indicando che la flessibilità della proteina è sostanzialmente sovrapponibile nei quattro complessi. In termini pratici, i residui più mobili e quelli più rigidi risultano gli stessi, e l'ampiezza delle oscillazioni è comparabile tra sistemi.

Il dato che mostra la maggiore similarità è la coppia **ST44-ST45 (0.981)**, mentre quella meno simile è ST44-STH2 **(0.953)**. Anche questo scostamento, tuttavia, rimane contenuto e non suggerisce cambiamenti strutturali macroscopici: è più plausibile che derivi da differenze locali nel pocket (ad esempio variazioni nella micro-dinamica di loop prossimi al sito di legame o di residui laterali coinvolti nei contatti con il ligando).

Questi risultati sono coerenti con quanto ci si aspetta quando il target proteico mantiene un folding stabile in tutte le simulazioni e il contributo dei ligandi si manifesta soprattutto a livello di rete di interazioni locali, più che come riorganizzazione della flessibilità globale della proteina. In altre parole i ligandi variano soprattutto il modo in cui si ancorano nel pocket, ma non sconvolgono la dinamica complessiva del backbone.

3.2.3 Considerazioni critiche

La similarità coseno riassume i profili RMSF in un singolo indice e quindi è molto utile per un confronto rapido, ma non sostituisce l'osservazione del profilo residuo per residuo quando si vogliono localizzare precisamente le differenze.

Per evitare interpretazioni eccessive, in questa tesi l'RMSF viene utilizzato principalmente come indicatore di "coerenza globale" tra i quattro complessi. Eventuali differenze funzionalmente rilevanti, se presenti, saranno discusse in modo più diretto nelle sezioni dedicate ai contatti proteina-ligando (H-bond, interazioni idrofobiche, WaterBridge, π-π), che sono più sensibili alle variazioni locali nel pocket.

3.3 Analisi della Struttura Secondaria (SSE)

3.3.1 Significato del parametro e obiettivo dell'analisi

La struttura secondaria descrive la frazione di residui che, lungo la simulazione, mantiene conformazioni ordinate come α-eliche e β-foglietti. In dinamica molecolare, monitorare la SSE è utile anche come controllo di qualità: variazioni marcate potrebbero indicare perdita di folding o instabilità del backbone.[2] In questo lavoro la SSE è stata usata per verificare se la proteina mantenesse un comportamento coerente nei quattro complessi (ST44, ST45, STH2, STH3) simulati per 100 ns, dato che cambia il ligando ma il target proteico è lo stesso.

3.3.2 Procedura di estrazione e confronto

Per ciascun sistema sono state considerate le percentuali medie di α-elica, β-foglietto e SSE totale riportate dai report di Desmond. Per ottenere un confronto quantitativo tra complessi, queste tre variabili sono state trattate come un vettore per sistema e confrontate con la similarità coseno. Valori prossimi a 1 indicano una composizione di struttura secondaria quasi identica.

3.3.3 Risultati

Le percentuali medie di SSE risultano molto simili tra i quattro complessi (Tabella 3.3)

Sistema	% Helix	% Strand	% SSE totale
ST44	28.16	7.64	35.80
ST45	29.05	8.04	37.09
STH2	28.68	8.73	37.41
STH3	28.34	8.38	36.72

Tabella 3.3. Percentuali medie di struttura secondaria (α-eliche, β-foglietti e SSE totale) nei quattro complessi

La SSE totale varia in un intervallo ristretto (35.80-37.41 %), con differenze assolute inferiori a ~1,6 punti percentuali. Anche le componenti Helix e Strand rimangono comparabili tra sistemi.

	ST44	ST45	STH2	STH3
ST44	1.000000	0.999994	0.999781	0.999880
ST45	0.999994	1.000000	0.999849	0.999928
STH2	0.999781	0.999849	1.000000	0.999985
STH3	0.999880	0.999928	0.999985	1.000000

Tabella 3.4. Similarità coseno della SSE tra i complessi (basata su %Helix, %Strand, %SSE totale).

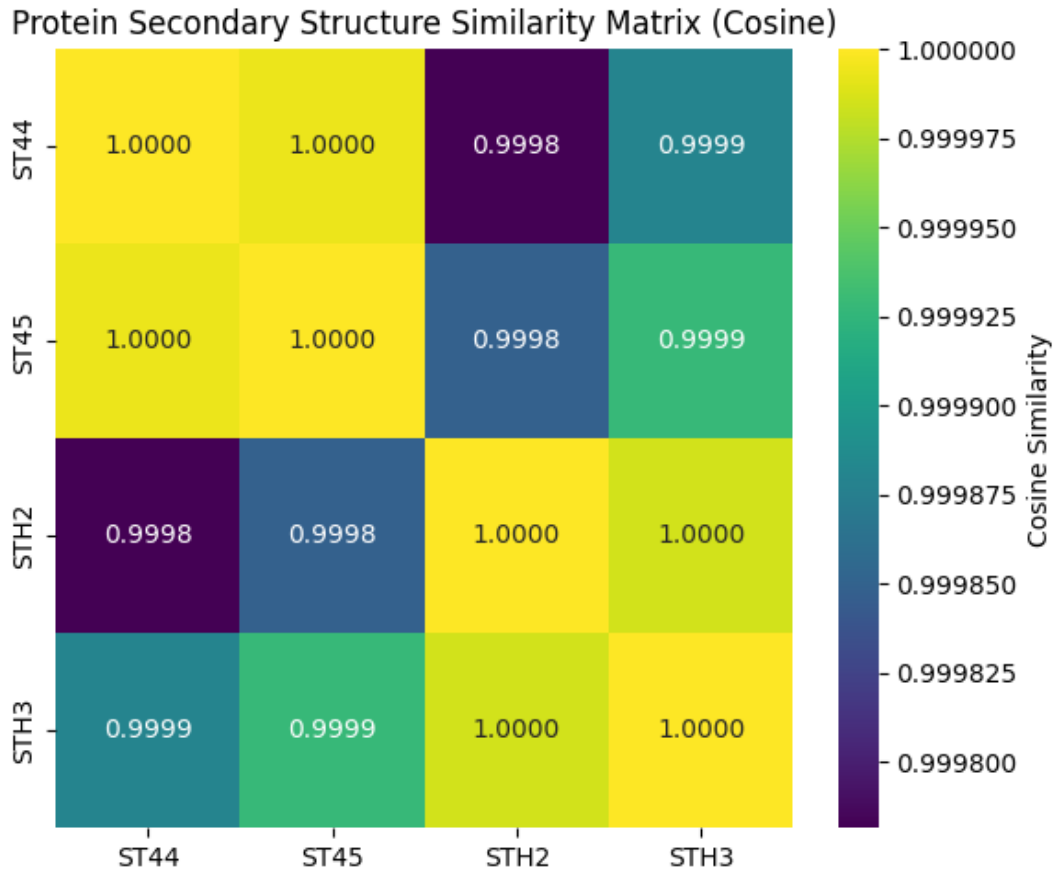

Figura 3.3. Heatmap della similarità coseno della struttura secondaria (SSE) tra ST44, ST45, STH2 e STH3.

La similarità molto prossima ad 1 per tutte le coppie conferma che la proteina mantiene praticamente la stessa composizione di struttura secondaria nei quattro complessi. Questo risultato è coerente con quanto osservato in RMSD/RMSF: non emergono segnali di instabilità globale o di riarrangiamenti del backbone attribuibili al cambio di ligando. In pratica, le eventuali differenze tra complessi tendono a riguardare soprattutto il modo in cui il ligando interagisce nel pocket (contatti specifici), più che la conformazione generale della proteina.

Nota critica: la similarità coseno è calcolata qui su sole tre variabili (%Helix, %Strand, %SSE totale). Con un numero così ridotto di feature e valori molto vicini tra loro, la metrica tende naturalmente a produrre valori estremamente elevati. Per questo motivo la matrice di similarità va interpretata soprattutto come conferma di assenza di cambiamenti macroscopici, non come strumento per discriminare differenze sottili tra i sistemi.

Nel complesso, la SSE risulta stabile lungo 100 ns e sostanzialmente sovrapponibile tra i quattro complessi. La proteina conserva il folding e non mostra segnali di unfolding o

riorganizzazioni estese. Questo supporta l'idea che, nel seguito, le differenze più informative tra i sistemi vadano cercate nelle analisi dei contatti proteina-ligando (H-bond, interazioni idrofobiche, water bridge, interazioni aromatiche) e nella flessibilità conformazionale del ligando.

3.4 Analisi delle interazioni proteina-ligando: legami idrogeno (H-bond)

3.4.1 Significato dell'analisi

Tra le interazioni non covalenti che contribuiscono al riconoscimento molecolare, i legami idrogeno sono particolarmente rilevanti perché combinano una componente elettrostatica con requisiti geometrici stringenti, rendendoli in genere più direzionali rispetto ai contatti idrofobici.[1] In una traiettoria di dinamica molecolare, la loro presenza e soprattutto la loro persistenza nel tempo possono indicare un contributo stabile al posizionamento del ligando nel sito di legame. In questa sezione sono stati confrontati i pattern di H-bond dei quattro complessi (ST44, ST45, STH2, STH3) per verificare quanto i residui coinvolti e le relative frequenze siano condivisi tra i sistemi.

3.4.2 Risultati

	ST44	ST45	STH2	STH3
ST44	1	0	0	0
ST45	0	1	0.919923	0.180273
STH2	0	0.919923	1	0.195965
STH3	0	0.180273	0.195965	1

Tabella 3.5. matrice di similarità coseno calcolata sui profili residuo–frequenza degli H-bond.

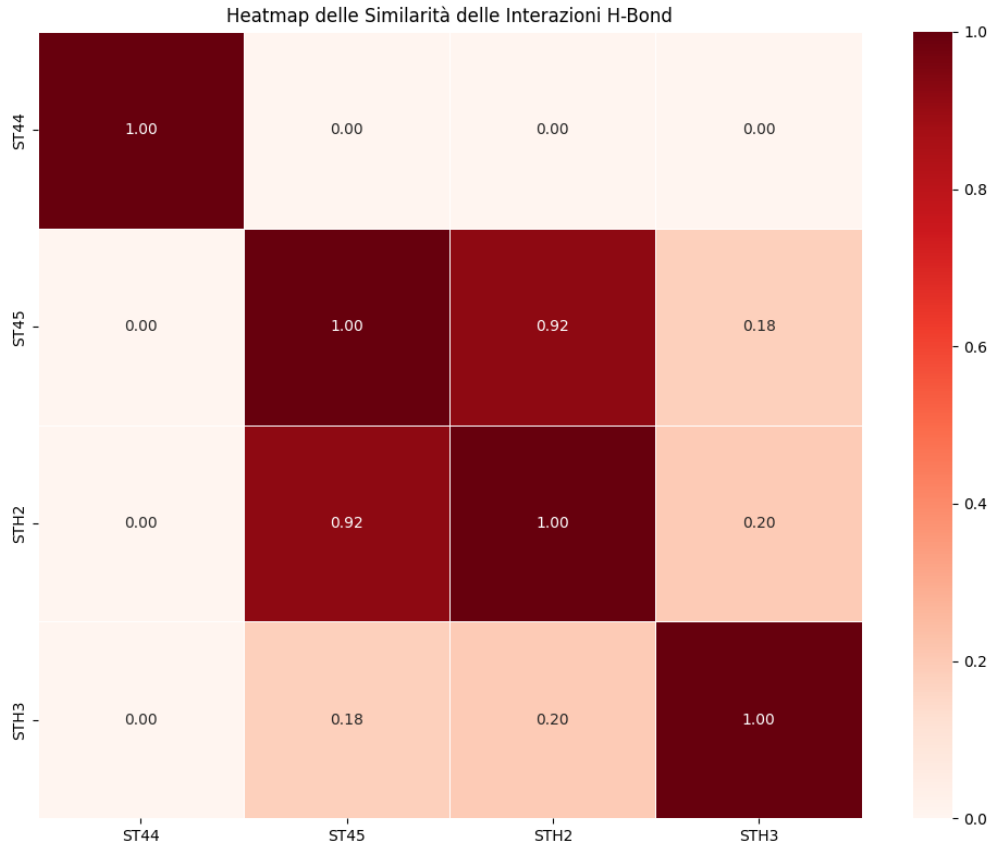

Figura 3.3. Heatmap delle interazioni H-Bond.

Il risultato più evidente è la forte somiglianza tra **ST45 e STH2 (similarità 0.92)**, che indica un pattern di H-bond molto simile: i due ligandi tendono a coinvolgere gli stessi residui e con una persistenza comparabile. Questo dato è coerente con l'idea che i due ligandi presentino gruppi polari posizionati in modo analogo nel pocket, favorendo un set di interazioni dirette sovrapponibile.

Al contrario, ST44 mostra similarità pari a 0 rispetto agli altri sistemi. In termini operativi, questo significa che i residui che formano H-bond in ST44 non coincidono (o coincidono in modo trascurabile) con quelli che compaiono negli altri complessi. Il controllo qualitativo degli eventi nei file suggerisce che in ST44 il contributo sia concentrato su un contatto polare dominante (ad esempio con Arg97), quindi su una modalità di ancoraggio più precisa.

STH3, infine, presenta valori bassi con tutti **(0.18-0.20 verso ST45/STH2)**, indicando che gli H-bond diretti non sono l'elemento più conservato del suo binding. Questo non implica necessariamente instabilità: un ligando può rimanere confinato nel sito anche se

il contributo polare diretto è ridotto, grazie a contatti idrofobici, interazioni aromatiche o interazioni mediate dal solvente.[1] Tuttavia, la scarsa sovrapposizione degli H-bond suggerisce che STH3 utilizzi un set diverso di ancoraggi polari o che tali interazioni siano più intermittenti.

3.4.3 Note critiche

La similarità coseno applicata ai profili residuo-frequenza è utile per confronti globali, ma non distingue se due sistemi condividono gli stessi residui con geometrie diverse o se presentano poche interazioni molto persistenti contro molte interazioni deboli e frammentarie.[1] Per questo motivo, i risultati di questa sezione vanno letti insieme alle altre classi di contatto (idrofobiche, π-π, water bridge) per ricostruire un quadro complessivo del meccanismo di stabilizzazione del complesso.

3.5 Analisi delle interazioni idrofobiche

3.5.1 Significato dell'analisi

Le interazioni idrofobiche sono un contributo centrale alla stabilizzazione dei complessi proteina-ligando, perché favoriscono l'inserzione del ligando nelle regioni apolari del pocket e aiutano a mantenere un orientamento di legame coerente nel tempo.[1] A differenza dei legami idrogeno, non richiedono una geometria direzionale precisa e dipendono in larga misura da effetti di desolvatazione e complementarietà tra superfici apolari di ligando e di pocket.[1] Nel contesto delle simulazioni MD, tali interazioni si manifestano come contatti ripetuti tra porzioni apolari del ligando e residui idrofobici della proteina. Lo scopo di questa sezione è confrontare in modo quantitativo i profili di similarità tra pattern di residui coinvolti e frequenza di contatto.

	ST44	ST45	STH2	STH3
ST44	1	0.609	0.526	0.838
ST45	0.609	1	0.808	0.730
STH2	0.526	0.808	1	0.744
STH3	0.838	0.730	0.744	1

Tabella 3.6. Matrice di similarità coseno delle interazioni idrofobiche (ST44, ST45, STH2, STH3)

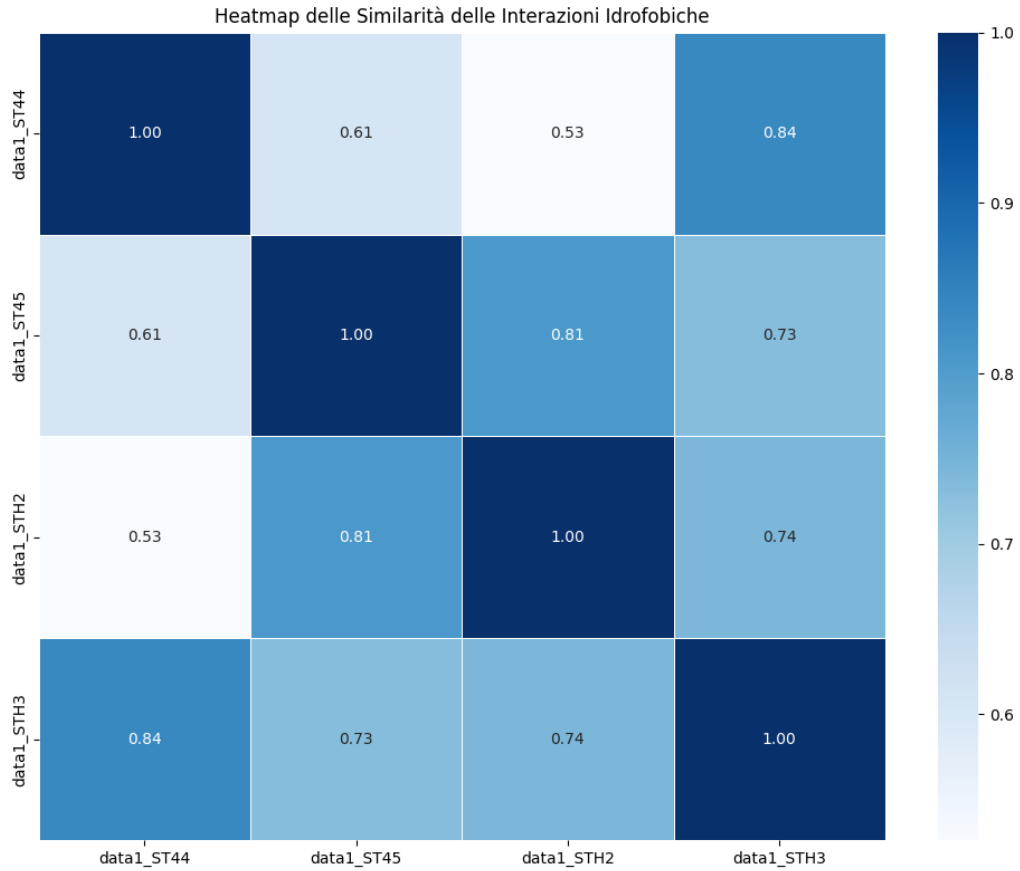

Figura 3.4.Heatmap delle interazioni idrofobiche

I valori di similarità si distribuiscono tra **0.536** e **0.838**, evidenziando una variabilità più marcata rispetto agli H-bond. Questo è coerente con la natura meno precisa delle interazioni idrofobiche, che tendono a coinvolgere più residui e porzioni del ligando in modo diffuso. [1]

Due relazioni emergono con maggiore chiarezza:

- **ST44-STH3 (0.838)**: i due complessi condividono un pattern di contatti apolari molto simile
- **ST45-STH2 (0.808)**: anche questa coppia mostra un'elevata sovrapposizione dei contatti idrofobici.

Al contrario, **ST44** risulta meno simile a **STH2 (0.526)** ed a **ST45 (0.609)**, suggerendo che, pur mantenendo la stabilità globale del complesso (coerentemente con RMSD/RMSF), il ligando in ST44 potrebbe sfruttare una porzione diversa del pocket idrofobico o un orientamento alternativo che coinvolge residui differenti.

Nel complesso, i dati indicano che i quattro sistemi condividono solo parzialmente la stessa rete idrofobica, con due coppie più affini, ST44-STH3 e ST45-STH2. Questo

risultato si integra con quanto osservato nelle sezioni precedenti: la stabilità del legame può essere raggiunta con combinazioni diverse di interazioni polari e apolari. La sezione successiva sui water bridge aiuta a chiarire se e quanto il solvente contribuisca a compensare differenze nei contatti diretti.

3.6 Analisi delle interazioni water bridge

3.6.1 Significato dell'analisi

I ponti d'acqua (water bridge) sono interazioni in cui una o più molecole d'acqua mediano il contatto tra un gruppo polare del ligando ed un residuo della proteina. In pratica, l'acqua crea un collegamento attraverso legami idrogeno (o contatti compatibili) con entrambi i partner, contribuendo alla stabilizzazione del complesso anche quando un'interazione diretta non è geometricamente favorita.[1]

In questo lavoro di tesi i water bridge sono stati confrontati tra i quattro complessi (ST44, ST45, STH2, STH3) per verificare se i water bridge seguono un pattern stabile e condiviso, o se cambiano in funzione del ligando.

3.6.2 Risultati

	ST44	ST45	STH2	STH3
ST44	1	0.324	0.063	0.369
ST45	0.324	1	0.044	0.536
STH2	0.063	0.044	1	0.038
STH3	0.369	0.536	0.038	1

Tabella 3.7. Matrice di similarità coseno delle interazioni water bridge

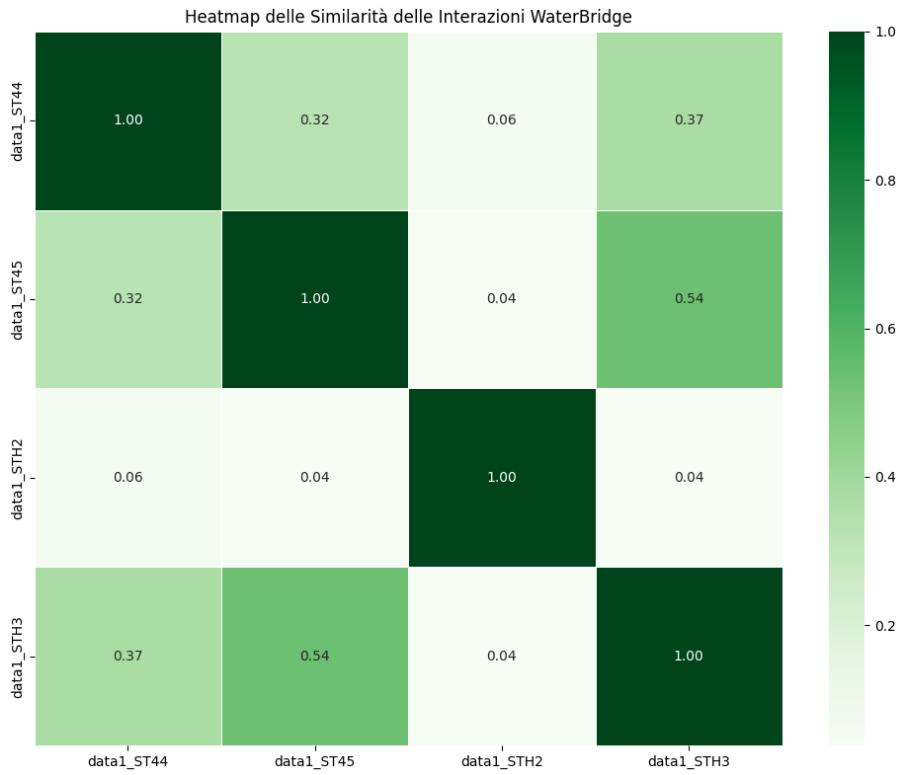

Figura 3.5. Heatmap della similarità coseno tra i profili water bridge nei quattro complessi.

I valori fuori diagonale risultano complessivamente bassi **(0.04-0.54)**, indicando una conservazione limitata del pattern di ponti d'acqua tra i diversi ligandi.

A differenza dei parametri strutturali globali (RMSD/RMSF e SSE), I water bridge mostrano un comportamento più variabile. Questo è coerente con la natura dinamica dell'acqua nel pocket: le molecole possono entrare e uscire rapidamente e la loro presenza dipende dalla disposizione dei gruppi polari del ligando e dalla micro-geometria della cavità.

La coppia più simile è **ST45-STH3 (0.536)**. Questo suggerisce che in questi due complessi una parte della mediazione del solvente avviene su residui comparabili e con frequenze relativamente simili. Anche **ST44** mostra una similarità moderata con **STH3 (0.369)** e con **ST45 (0.324)**, indicando la possibile presenza di alcuni residui polari che possono partecipare a water bridge in più sistemi, ma senza un network pienamente conservato.

STH2 risulta invece quasi indipendente dagli altri **(0.038-0.063)**. Un'interpretazione plausibile è che in questo complesso la stabilizzazione sia sostenuta soprattutto da interazioni dirette ligando-proteina, riducendo la necessità (o la possibilità) di water

bridge persistenti. In termini pratici, il ligando potrebbe occupare in modo più compatto la cavità o posizionare i gruppi polari in modo da favorire contatti diretti.
In sintesi, i water bridge sembrano contribuire in modo differenziale tra ligandi e risultano meno adatti, rispetto ad altre categorie di contatto, a definire un meccanismo comune condiviso da tutti i sistemi. Più che un indicatore di stabilità globale, rappresentano un'informazione aggiuntiva sul ruolo del solvente e sulla specificità del microambiente di legame.

3.7 Analisi delle interazioni π-π

3.7.1 Introduzione e significato dell'analisi

Le interazioni π-π rappresentano un contributo non covalente importante alla stabilità dei complessi ligando-proteina quando nel sito di legame sono presenti anelli aromatici (del ligando e/o di residui proteici come Phe, Tyr, Trp, His).[1]
Durante la simulazione di dinamica molecolare, la presenza e la persistenza di contatti aromatici può contribuire a mantenere l'orientamento del ligando nel pocket e stabilizzarne il posizionamento complessivo, in sinergia con interazioni polari e idrofobiche.[1]
In questa sezione sono state confrontate le interazioni π-π nei quattro complessi (ST44, ST45, STH2, STH3) tramite un'analisi di similarità tra i profili di contatto.

3.7.2 Presentazione dei risultati

	ST44	ST45	STH2	STH3
ST44	1.000	0.802	0.724	0.798
ST45	0.802	1.000	0.952	0.810
STH2	0.724	0.952	1.000	0.783
STH3	0.798	0.810	0.783	1.000

Tabella 3.8. Matrice di similarità coseno dei profili di interazione π–π (residuo–frequenza).

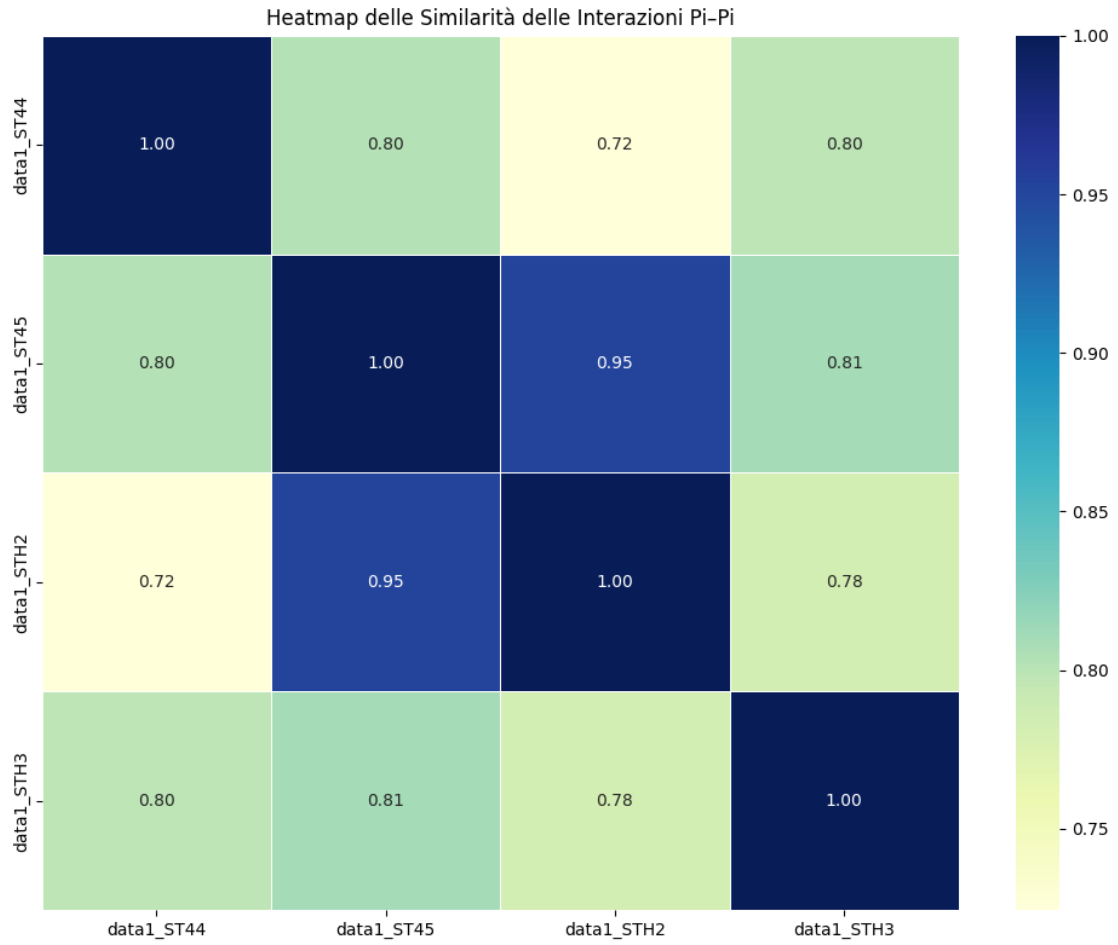

Figura 3.6. Heatmap delle interazioni π–π (cosine similarity).

I valori di similarità risultano complessivamente elevati **(0.724-0.952)** per tutte le coppie, indicando che i pattern di contatto aromatico sono in larga parte conservati tra i sistemi. L'elevata similarità suggerisce che il contributo delle interazioni aromatiche costituisca un elemento comune del sito di legame e che, indipendentemente dalle differenze chimiche tra i ligandi, l'orientamento relativo delle porzioni aromatiche rimanga comparabile lungo la simulazione.

La coppia con la maggiore similarità è **ST45-STH2 (0.9552),** coerente con quanto osservato anche per altre categorie di interazione (ad esempio H-bond).

Le similarità tra **ST44** e gli altri sistemi restano comunque alte **(0.724-0.803)**, indicando un binding mode aromatico in parte condiviso ma non perfettamente sovrapponibile (possibili differenze di orientamento/distanza o di residui coinvolti).

A differenza dei water bridge, che mostrano forte specificità e variabilità tra i sistemi, i contatti π–π risultano più riproducibili: questo è compatibile con l'idea che le interazioni

aromatiche, una volta formate nel pocket, abbiano una componente più strutturata rispetto alle interazioni mediate dal solvente.

3.7.3 Considerazioni e conclusioni

In sintesi, l'analisi π–π evidenzia un quadro di forte conservazione dei contatti aromatici tra i quattro complessi.

Il risultato più marcato è l'alta sovrapposizione tra ST45 e STH2, mentre ST44 e STH3 mostrano comunque pattern simili ma leggermente meno allineati.

Nel complesso, le interazioni π–π sono un contributo stabilizzante del binding in tutti i sistemi, e si integrano con le altre categorie di contatto (polari e idrofobiche) nel mantenere il ligando correttamente posizionato nel pocket durante i 100 ns di simulazione.

3.8 Analisi dei profili di L-Torsion

3.8.1 Introduzione e significato dell'analisi

L'analisi dei profili torsionali del ligando permette di descrivere la flessibilità conformazionale del ligando durante la dinamica molecolare. I legami rotabili possono esplorare angoli di torsione diversi nel tempo: un comportamento stabile suggerisce un ancoraggio conformazionale nel sito di legame, mentre transizioni frequenti tra stati o ampiezze elevate indicano maggiore libertà interna e adattamento dinamico. Nel presente lavoro i profili torsionali sono stati confrontati tra i complessi ST44, ST45, STH2 e STH3 per verificare se i ligandi condividano o meno un comportamento conformazionale simile all'interno del pocket.

3.8.2 Descrizione metodologica

L'analisi è stata eseguita sui file di output di Desmond relativi alle torsioni del ligando (L-Torsion.dat). Per rendere confrontabili i quattro sistemi, i dati sono stati sintetizzati con due strategie complementari e poi comparati mediante similarità coseno.

Approccio A – media per colonna (profilo medio per torsione):

Per ciascun legame rotabile è stata calcolata la media dell'angolo di torsione lungo tutta la simulazione. Ogni ligando è quindi rappresentato da un vettore di valori medi (uno per torsione), utile per confrontare la "configurazione media" complessiva.

Approccio B – media per riga (andamento temporale medio):

Per ogni frame è stata calcolata la media delle torsioni. Si ottiene così una serie temporale che descrive come varia nel tempo la flessibilità globale del ligando, indipendentemente dal dettaglio del singolo legame.

In entrambi i casi la similarità coseno è stata usata per quantificare la somiglianza tra i profili dei sistemi (valori da -1 a 1): valori positivi indicano trend concordi, valori prossimi a zero assenza di relazione e valori negativi trend opposti.

3.8.3 Risultati – media per colonna

	ST44	ST45	STH2	STH3
ST44	1.000	-0.323	-0.300	0.276
ST45	-0.323	1.000	0.363	-0.026
STH2	-0.300	0.363	1.000	-0.393
STH3	0.276	-0.026	-0.393	1.000

Tabella 3.9. Matrice di similarità coseno delle torsioni del ligando media per colonna.

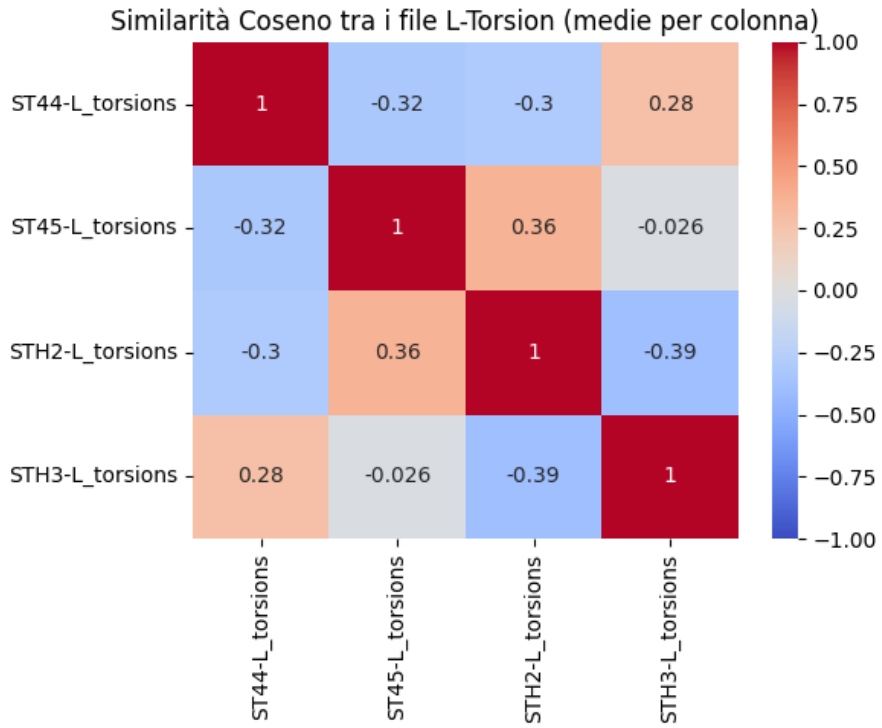

Figura 3.7. Heatmap della similarità coseno dei profili L-Torsion (media per colonna).

La matrice evidenzia una bassa concordanza complessiva tra i profili medi delle torsioni: molti confronti sono vicini allo zero o negativi. L'unica coppia con similarità moderatamente positiva è **ST45 – STH2 (0.363)**, mentre ad esempio **ST44 – ST45 (-0.323)** e **STH2 – STH3 (-0.393)** mostrano andamenti medi opposti. In pratica, i ligandi tendono a stabilizzarsi su conformazioni medie differenti, anche se rimangono nel pocket secondo quanto indicato dalle analisi RMSD.

3.8.4 Risultati – media per riga

	ST44	ST45	STH2	STH3
ST44	1.000	-0.334	-0.175	0.123
ST45	-0.334	1.000	0.196	-0.301
STH2	-0.175	0.196	1.000	-0.003
STH3	0.123	-0.301	-0.003	1.000

Tabella 3.10 Matrice di similarità coseno delle torsioni del ligando media per riga.

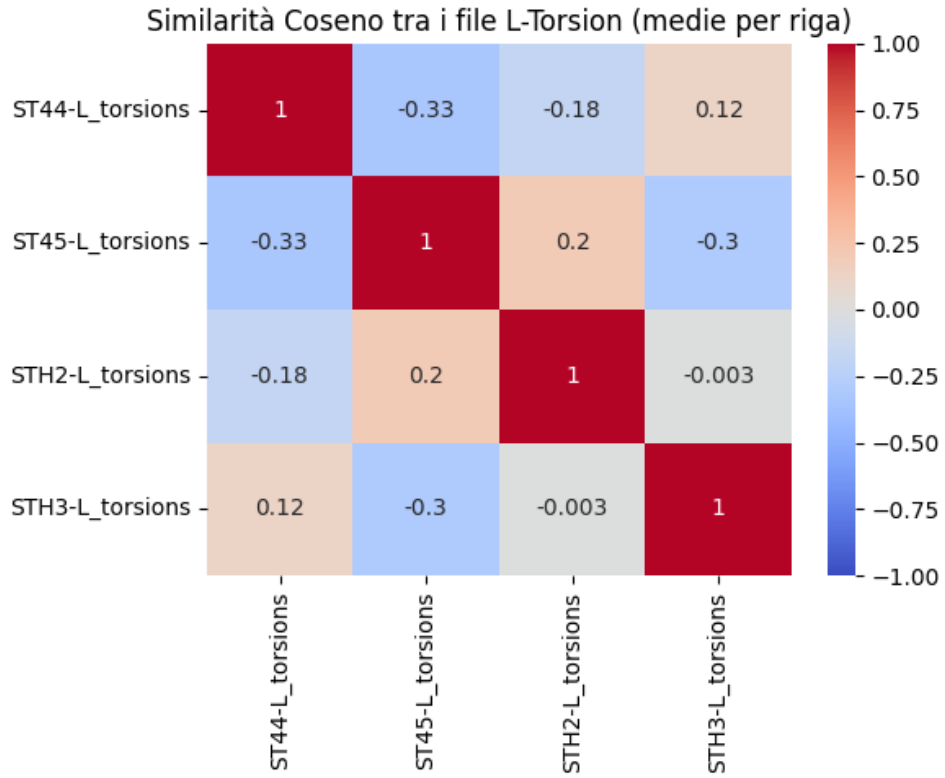

Figura 3.8. Heatmap della similarità coseno delle serie temporali L-Torsion (media per riga).

Anche considerando la tendenza temporale media, le similarità restano basse o negative. La coppia **ST45 – STH2** mantiene una debole concordanza positiva **(0.196),** mentre **ST44 – ST45 (-0.334)** e **ST45 – STH3 (-0.301)** risultano discordanti. Questo suggerisce che la variabilità torsionale nel tempo è fortemente dipendente dal ligando, con transizioni conformazionali e/o ampiezze di fluttuazione non sovrapponibili tra i sistemi.

3.8.5 Interpretazione biochimico – strutturale

Nel complesso, i profili torsionali indicano che i quattro ligandi non condividono un comportamento conformazionale unico: il binding può essere stabile in termini di posizione (RMSD), ma la stabilità "interna" del ligando, ovvero le torsioni possono differire sensibilmente.

- ST45 e STH2: mostrano le concordanze più alte (seppur moderate) in entrambi gli approcci, coerenti con il fatto che nelle altre analisi (H-bond e π-π) questi sistemi risultano

spesso più simili. Ciò è compatibile con un binding mode relativamente riproducibile, che limita alcune rotazioni interne.

- ST44 e STH3: presentano similarità basse o negative rispetto agli altri, indicando che i legami rotabili tendono ad occupare stati differenti. Questo non implica instabilità del complesso, ma suggerisce una maggiore plasticità conformazionale del ligando all'interno del pocket

Un punto importante: la similarità coseno è un indicatore globale e non distingue se la discordanza derivi da poche torsioni chiave o da differenze diffuse su tutto il ligando. Per una discussione più fine, sarebbe necessario isolare le torsioni principali (legate a gruppi funzionali nel pocket) e valutarne separatamente le popolazioni conformazionali.
In questa tesi, l'obiettivo è invece di fornire un confronto sintetico e riproducibile tra i sistemi.

3.8.6 Conclusioni

L'analisi L-Torsion evidenzia che:

1- I profili torsionali sono in generale poco sovrapponibili tra i quattro complessi;

2- ST45 e STH2 costituiscono la coppia più simile, con similarità positiva in entrambe le modalità di sintesi;

3- ST44 e STH3 mostrano comportamenti più indipendenti, con confronti spesso prossimi a zero o negativi;

4- La stabilità del legame non è descritta solo dalla posizione del ligando (RMSD), ma anche dalla sua flessibilità interna, che può rappresentare una componente importante dell'adattamento nel sito di legame.

4. Materiali e Metodi

4.1 Materiali

Il presente lavoro è stato condotto su quattro complessi ligando-proteina, identificati come ST44, ST45, STH2 e STH3, ottenuti da simulazioni di dinamica molecolare. L'attività svolta in prima persona ha riguardato il post-processing degli output di simulazione e lo sviluppo di una pipeline analitica in Python, implementata in ambiente Google Colab, finalizzata alla costruzione di tabelle e rappresentazioni comparative tra diversi sistemi. Tale pipeline ha costituito il laboratorio computazionale del lavoro sperimentale.

4.1.1 Dati di input

I dati analizzati sono costituiti da file di testo in formato .dat, prodotti da Desmond e organizzati in sottocartelle contenente una directory "raw-data". Il notebook è stato progettato per identificare automaticamente i file di interesse all'interno della cartella di lavoro, leggerli in modo strutturato e convertirli in dataset omogenei per i quattro complessi studiati.

4.1.2 File utilizzati e contenuto

I file che sono stati analizzati sono i seguenti:

PL_RMSD.dat:

File tabellare con una riga per frame della traiettoria. Nel notebook è stata selezionata la colonna Lig_wrt_Protein (Å), che descrive l'RMSD del ligando rispetto alla proteina dopo allineamento del complesso sul backbone proteico. Questo parametro è stato impiegato come descrittore della stabilità del ligando nel sito di legame.

P_RMSF.dat:

File contenente i valori di RMSF per residuo della proteina. Ai fini dell'analisi comparativa è stata estratta la colonna CA, corrispondente ai valori di RMSF dei carboni α, utilizzata come firma della mobilità locale del backbone lungo la sequenza proteica.

SSE:

Valori medi di %Helix, %Strand e %SSE totale ricavati dai report Desmond e utilizzati come descrittori sintetici della conservazione della struttura secondaria proteica nel corso della simulazione

PL-Contacts_HBond.dat:

File contenente gli eventi di legame idrogeno osservati lungo la traiettoria. Nel notebook, tali eventi sono stati aggregati per residuo al fine di costruire un profilo residue-based di interazione.

PL-Contacts_Hydrophobic.dat:

File relativo agli eventi di contatto idrofobico tra proteina e ligando. I dati sono stati trasformati in profili residuo-frequenza confrontabili tra sistemi.

PL-Contacts_WaterBridge.dat:

File relativo agli eventi di contatto idrofobico tra proteina e ligando. I dati sono stati trasformati in profili residuo-frequenza confrontabili tra sistemi.

PL-Contacts_Pi-Pi.dat:

File contenente gli eventi di interazione aromatica di tipo π-π, successivamente aggregati nel notebook in una firma interazionale comparabile tra i complessi.

L-Torsion.dat:

File contenente i valori delle coordinate torsionali del ligando lungo la traiettoria. Le righe descrivono i frame di simulazione, mentre le colonne corrispondono alle diverse torsioni monitorate. Tali dati sono stati caricati come matrici numeriche e sintetizzati nel notebook secondo due modalità complementari: medie per colonna e serie temporali delle medie per frame.

4.2 Metodi

4.2.1 Ambiente computazionale e librerie utilizzate

Il post processing degli output di simulazione è stato eseguito in Python all'interno di notebook sviluppati in Google Colab. In questo contesto, il notebook ha rappresentato l'ambiente operativo dell'intero workflow, permettendo la lettura automatica dei file, l'organizzazione dei dati in strutture numeriche confrontabili, il calcolo di metriche quantitative tra sistemi e la produzione di output tabellari e grafici.

Le principali librerie utilizzate:

Pandas è stata la libreria di riferimento per la gestione dei dati tabellari. Utilizzata per il parsing dei file .dat prodotti dall'analisi Desmond, per la costruzione di Data Frame e Series, per l'unione di profili provenienti da sistemi differenti per la gestione dei valori mancanti. Il suo impiego ha consentito di trattare in modo ordinato insiemi di dati

eterogenei, facilitando confronti tra diversi complessi ligandi-proteina e permettendo successive elaborazioni quantitative.

Numpy è stata impiegata per la manipolazione numerica degli array e per l'esecuzione di operazioni vettoriali efficienti. In particolare, si è rivelata utile nell'analisi dei profili torsionali del ligando, nel calcolo di medie per riga o per colonna e nella trasformazione dei dati in formati compatibili con le successive analisi matematiche. L'utilizzo di NumPy ha reso possibile una gestione più rigorosa ed efficiente dei dati numerici ad alta dimensionalità derivanti dalla simulazione.

Scikit-learn è stata utilizzata per il calcolo della similarità coseno tra vettori numerici rappresentativi dei diversi profili conformazionali. Questa misura ha permesso di quantificare il grado di somiglianza tra sistemi differenti, sia considerando valori medi sintetici sia valutando l'andamento temporale dei dati. L'impiego di questa libreria ha quindi fornito un supporto quantitativo al confronto tra i complessi studiati.

Matplotlib ha costituito la libreria di base per la realizzazione dei grafici, consentendo di visualizzare l'andamento dei parametri strutturali lungo la traiettoria di simulazione. È stata utilizzata per generare figure lineari, grafici comparativi e rappresentazioni necessarie a descrivere l'evoluzione temporale dei sistemi analizzati.

Seaborn, costruita su Matplotlib, è stata adottata per migliorare la resa visiva e interpretativa delle rappresentazioni grafiche, in particolare per la costruzione di heatmap, grafici comparativi adatti a mettere in evidenzia pattern, similarità e differenza tra i sistemi. Il suo utilizzo ha reso più immediata la lettura dei risultati e più efficace la comunicazione visiva dei dati.

Ipywidgets è stata infine impiegata per introdurre nel notebook una semplice interfaccia grafica interattiva (GUI), utile per selezionare file, parametri o modalità di visualizzazione senza modificare manualmente il codice ad ogni esecuzione. Questo approccio ha migliorato la fruibilità dell'ambiente di lavoro e ha reso più agevole l'esplorazione dei risultati analitici.

Nel complesso, l'integrazione di queste librerie ha permesso di costruire una pipeline computazionale completa, dalla lettura dei dati grezzi fino alla loro elaborazione quantitativa e rappresentazione grafica, garantendo riproducibilità, chiarezza metodologica e robustezza analitica.

4.2.2. Principi di pre-processing e metrica di confronto

Per rendere i sistemi direttamente confrontabili, ciascun output è stato trasformato in un vettore numerico specifico per complesso. A seconda della natura del dato, i sistemi sono stati rappresentati come serie temporali, nel caso di RMSD e delle torsioni temporali; come profili residue-based, nel caso di RMSF e dei contatti ligando-proteina; oppure come vettori sintetici di parametri strutturali, nel caso della struttura secondaria.

Nei profili residue-based, l'unione degli indici provenienti dai diversi sistemi può generare celle mancanti. In questi casi, i valori assenti sono stati gestiti mediante fillna(0), così da garantire una dimensionalità uniforme dei vettori e consentire il confronto quantitativo tra complessi. Tale scelta va intesa come un passaggio tecnico di armonizzazione del dataset e non come attribuzione autonoma di significato strutturale ai valori mancanti.

La metrica adottata per confrontare i profili è la similarità coseno:

$$\cos(\theta) = (x \cdot y) / (||x|| \cdot ||y||)$$

Questa misura è stata utilizzata per quantificare il grado di sovrapposizione tra i diversi profili numerici ottenuti dal notebook. Valori prossimi a 1 indicano profili fortemente simili, mentre valori prossimi a 0 indicano una ridotta sovrapposizione. Nel caso delle analisi torsionali, la similarità può assumere anche valori negativi, poiché l'angolo tra vettori può superare i 90°.

4.2.3 Workflow analitico implementato nel notebook

Per garantire un'analisi riproducibile, standardizzata e quantitativa degli output prodotti dalle simulazioni molecolari, è stato sviluppato un notebook in Python organizzato come una pipeline analitica modulare. Il workflow è stato progettato per acquisire automaticamente i file generati da Desmond, estrarre i parametri strutturali e dinamici di interesse, convertirli in strutture numeriche confrontabili e applicare procedure di analisi comparativa tra i diversi complessi ligando – proteina.

L'intero processo è stato suddiviso in funzioni dedicate, ciascuna associata ad una specifica categoria di output. In particolare, il notebook comprende moduli per l'analisi RMSD, RMSF, struttura secondaria, legami a idrogeno, interazioni idrofobiche, water

bridge, interazioni π-π e profili torsionali del ligando. Tale organizzazione ha consentito di separare con chiarezza le fasi di acquisizione dei dati, trasformazione numerica, confronto quantitativo e visualizzazione finale, migliorando la leggibilità del codice e la ripetibilità dell'analisi.

In termini generali, il workflow seguito può essere schematizzato nelle seguenti fasi:

- Identificazione automatica dei file di output nelle directory di lavoro;
- Parsing strutturato dei dati in tabelle o array numerici.
- Estrazione dei descrittori rilevanti per ciascun tipo di analisi
- Costruzione di matrici comparative tra i diversi sistemi
- Calcolo della similarità tra profili dinamici o interazionali;
- Esportazione dei risultati in formato tabellare e grafico

4.2.4 Analisi dei profili RMSD

L'analisi dei profili RMSD è stata implementata mediante una funzione dedicata del notebook che ricerca ricorsivamente i file PL_RMSD.dat presenti nella directory di lavoro, li legge in forma tabellare ed assegna esplicitamente le intestazioni corrispondenti ai descrittori riportati da Desmond. Tra le colonne disponibili è stata selezionata Lig_wrt_protein, utilizzata come descrittore dello scostamento del ligando rispetto alla proteina nel corso della simulazione.
Per ciascun sistema, la serie temporale corrispondente è stata memorizzata in una struttura indicizzata per complesso e successivamente convertita in una matrice numerica in cui ogni riga rappresenta un sistema e ogni colonna un frame della traiettoria. Su tale matrice è stata applicata la similarità coseno, ottenendo una matrice quadrata di somiglianza tra i profili RMSD dei quattro complessi. Il workflow si conclude con l'esportazione del risultato in formato .csv e con la generazione di una heatmap riassuntiva.

4.2.5 Analisi dei profili RMSF

Per l'analisi della flessibilità locale della proteina sono stati utilizzati i file P_RMSF.dat, letti dal notebook come tabelle strutturate. Ai fini del confronto tra sistemi è stata selezionata la colonna CA, corrispondente ai valori di RMSF dei carboni α, utilizzata come indicatore della mobilità del backbone per residuo. La scelta di mantenere il profilo

residue-based, anziché ridurlo preliminarmente ad un valore medio ha consentito di preservare l'informazione locale distribuita lungo la sequenza proteica.
I profili dei diversi sistemi sono stati quindi trasformati in un Data Frame comune, nel quale le righe rappresentano i residui e le colonne i complessi. Dopo la gestione dei valori mancanti mediante fillna(0), il Data Frame è stato trasposto per trattare ciascun sistema come vettore indipendente, sul quale è stata calcolata la similarità coseno. Anche in questo caso il notebook produce una matrice di similarità esportata in formato .csv e una heatmap corrispondente.

4.2.6 Analisi della struttura secondaria

L'analisi della struttura secondaria è stata impiegata come controllo sintetico della conservazione del folding proteico durante la simulazione. A differenza delle analisi basate su file frame by frame, in questo caso il notebook utilizza direttamente i valori medi di Helix, Strand e Total_SSE ricavati dai report Desmond per ciascun sistema. Tali parametri sono stati organizzati in un Data Frame in cui ogni riga rappresenta un complesso e ogni colonna una componente della struttura secondaria.
Su questa matrice sintetica è stata applicata la similarità coseno, ottenendo una misura quantitativa della somiglianza tra i profili medi di struttura secondaria dei quattro complessi. Anche questo risultato è stato estratto in formato tabellare e rappresentato graficamente tramite heatmap.

4.2.7 Analisi dei contatti ligando-proteina

Il notebook comprende una sezione specifica dedicata ai contatti ligando-proteina, finalizzata a trasformare gli eventi interazionali osservati lungo la simulazione in profili residue-based confrontabili tra sistemi. Sono state considerate quattro classi principali di interazione: legami a idrogeno, contatti idrofobici, water bridge e interazioni aromatiche di tipo π-π. Questa classificazione è coerente con i report Desmond, che distinguono esplicitamente le principali categorie di contatto monitorate nel tempo.
Dal punto di vista metodologico, il workflow dei contatti segue una logica comune: identificazione del file di input, lettura strutturata degli eventi, aggregazione per residuo, costruzione di una matrice comparativa tra sistemi e applicazione della similarità coseno. Tuttavia, il notebook distingue correttamente due diverse modalità di quantificazione. Nel caso degli H-bond viene costruito un profilo basato sul conteggio delle occorrenze per

residuo; nel caso di interazioni idrofobiche, water bridge e π-π il conteggio viene normalizzato sul numero di frame unici, ottenendo una frequenza percentuale di contatto per residuo. Questa distinzione è stata mantenuta esplicitamente nel protocollo, in quanto riflette fedelmente la logica implementata nel codice.

4.2.8 Analisi dei legami a idrogeno

Per l'analisi degli H-bond il notebook utilizza i file PL-Contacts_HBond.dat, letti come tabelle di eventi contenenti frame, residuo coinvolto, catena, nome del residuo e frammento del ligando. Per ciascun sistema, il codice esegue il conteggio delle occorrenze dei residui coinvolti mediante value_counts(), costruendo così un profilo residue-based degli eventi osservati lungo la traiettoria.

I profili ottenuti vengono successivamente organizzati in un Data Frame comparativo comune, completato con fillna(0) e trasposto prima dell'applicazione della similarità coseno. L'output finale consiste in una matrice di similarità salvata in formato .csv e in una heatmap dedicata. È opportuno precisare che, in questa specifica analisi, il notebook restituisce una misura di abbondanza osservata degli eventi per residuo e non una persistenza percentuale normalizzata sul numero totale di frame.

4.2.9 Analisi delle interazioni idrofobiche, dei water bridge e delle interazioni π-π

Una procedura analoga è stata adottata per le interazioni idrofobiche, i water bridge e le interazioni π-π, con la differenza che in questi casi il notebook calcola una frequenza percentuale di contatto per residuo. Dopo la lettura dei rispettivi file di input, il codice conteggia le occorrenze per residuo, le normalizza sul numero di frame unici e moltiplica il risultato per 100, ottenendo così un profilo residuo-frequenza direttamente confrontabile tra sistemi

Nel caso delle interazioni π-π, il notebook utilizza le colonne relative ai due residui coinvolti nell'evento interazionale, le inserisce in un'unica lista di identificativi residue-based e applica quindi la medesima logica di aggregazione e normalizzazione. I profili risultanti vengono trasformati in matrici comparative, uniformati per dimensionalità e confrontati con la similarità coseno. L'adozione di una logica comune per queste tre

categorie ha consentito di mantenere coerenza interna al protocollo e di costruire firme interazionali confrontabili tra i quattro complessi.

4.2.10 Analisi delle torsioni del ligando

Per approfondire la flessibilità conformazionale dei ligandi nel sito di legame, il notebook include due analisi torsionali complementari. In entrambe le modalità i file torsionali vengono caricati come matrici numeriche, limitando il confronto al numero minimo comune di coordinate torsionali disponibili tra i diversi sistemi, così da evitare distorsioni dovute a dimensionalità non omogenee.

Nella prima modalità, i dati vengono sintetizzati mediante la media dei valori per colonna, ottenendo per ciascun sistema un vettore che rappresenta il profilo torsionale medio del ligando. Nella seconda modalità, per ciascun frame viene calcolata la media dei valori torsionali disponibili, costruendo così una serie temporale globale rappresentativa dell'evoluzione conformazionale del ligando nel tempo. In entrambi i casi, il confronto tra sistemi è stato effettuato tramite similarità coseno e sintetizzato in matrici esportate in formato .csv e in corrispondenti heatmap.

4.2.11 Interfaccia del notebook, output e tracciabilità

Per rendere l'esecuzione della pipeline più ordinata e accessibile, il notebook è stato dotato di un'interfaccia grafica basata su ipywidgets. L'interfaccia comprende un'area di output dedicata e una serie di pulsanti, ciascuno associato all'avvio di una specifica funzione analitica. Alla selezione di una funzione, l'output precedente viene rimosso e vengono mostrati in sequenza i principali risultati prodotti, quali matrici numeriche, file esportati e rappresentazioni grafiche. Questa componente non modifica la logica scientifica del workflow, ma ne migliora la fruibilità e l'ordine di consultazione.

Per ciascuna analisi implementata, la pipeline produce output sia tabellare sia grafici. In particolare, vengono generati file .csv contenenti le matrici di similarità tra i sistemi e rappresentazioni grafiche sotto forma di heatmap, utili per una lettura immediata dei rapporti di somiglianza o divergenza tra i complessi. La produzione sistematica di output salvati su file ha rappresentato un elemento importante ai fini della tracciabilità del protocollo, poiché ha consentito di conservare in modo ordinato ogni passaggio del workflow e di recuperare i risultati indipendentemente dalla riesecuzione completa del notebook.

5. Conclusioni e sviluppi futuri

Nel complesso, i risultati ottenuti evidenziano un comportamento globale della proteina ampiamente conservato nei quattro sistemi analizzati. I profili di struttura secondaria mostrano valori molto simili tra i complessi, così come gli andamenti RMSF calcolati sui carboni α, indicando che la variazione del ligando non determina alterazioni macroscopiche del folding proteico né modificazioni rilevanti della flessibilità globale del backbone.

Le principali differenze emergono a livello del riconoscimento molecolare e dell'adattamento conformazionale del ligando. L'analisi comparativa delle interazioni non covalenti ha evidenziato pattern distinti tra i complessi, suggerendo modalità di stabilizzazione differenti. In particolare, i sistemi ST45 e STH2 mostrano una maggiore coerenza relativa, soprattutto per quanto riguarda la componente polare ed aromatica, mentre ST44 e STH3 presentano comportamenti più autonomi e meno sovrapponibili. Anche l'analisi dei profili torsionali conferma che i ligandi non condividono necessariamente la stessa dinamica conformazionale interna durante la simulazione, suggerendo che la stabilità del complesso dipenda non solo dalla presenza di contatti favorevoli, ma anche dalla capacità del ligando di adattarsi al microambiente del sito di legame.

Nel loro insieme, i risultati delineano un quadro coerente in cui la proteina conserva una stabilità globale, mentre le differenze tra i sistemi si concentrano principalmente nelle modalità di interazione del ligando e nella sua dinamica interna. Ciò suggerisce che complessi differenti possano raggiungere un comportamento dinamico favorevole attraverso combinazioni diverse di contributi polari, apolari, aromatici e di adattamento conformazionale.

Il lavoro presenta tuttavia alcuni limiti che devono essere considerati nell'interpretazione dei risultati. In primo luogo, la finestra temporale di 100 ns potrebbe non essere sufficiente a campionare pienamente eventi conformazionali più lenti o transizioni rare.[2] In secondo luogo, le analisi comparative sono state condotte mediante la similarità coseno, una metrica utile per confrontare la forma dei profili numerici, ma non sostitutiva di una valutazione diretta dell'affinità di legame, che richiederebbe approcci specifici di stima energetica.

Alla luce di tali considerazioni, possibili sviluppi futuri potranno riguardare l'estensione della durata delle simulazioni, l'esecuzione di repliche indipendenti e l'integrazione con metodi di stima energetica, quali MM-GBSA o altri approcci di calcolo della free energy.

Potranno inoltre essere approfonditi gli aspetti temporali delle interazioni mediante analisi di persistenza o clustering conformazionale.

In conclusione, il presente studio ha consentito di definire un protocollo analitico ordinato e riproducibile per il confronto tra complessi ligando-proteina, basato sull'integrazione di descrittori strutturali, interazionali e conformazionali. Oltre a fornire un quadro comparativo dei quattro sistemi analizzati, il lavoro propone un'impostazione metodologica trasferibile ad altri sistemi di interesse farmaceutico, utile per future applicazioni nell'ambito della dinamica molecolare e del drug design.

6. Il codice

```
import os
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics.pairwise import cosine_similarity

def analisi_rmsd_lig_wrt_protein(
    base_path="/content/drive/MyDrive/tesi/",
    output_csv_path="/content/drive/MyDrive/tesi/rmsd_similarity_matrix.csv"
):
    """
    Analisi RMSD Lig_wrt_Protein:
    - cerca tutti i file PL_RMSD.dat sotto base_path
    - estrae la colonna 'Lig_wrt_Protein'
    - costruisce la matrice di similarità (coseno) tra le serie temporali
    - salva la matrice in CSV
    - mostra la matrice e la heatmap
    """

    file_name = "PL_RMSD.dat"
    all_lig_rmsd_data = {}

    # Cerca ricorsivamente tutti i PL_RMSD.dat
    for root, dirs, files in os.walk(base_path):
        for file in files:
            if file == file_name:
                file_path = os.path.join(root, file)
                print(f"Processing data from: {file_path}")

                try:
                    # Legge i dati, saltando la prima riga e usando la prima colonna come indice
                    df = pd.read_csv(
                        file_path,
                        sep=r"\s+",
                        skiprows=[0],
                        index_col=0,
                        names=[
                            "Frame",
                            "Prot_CA",
                            "Prot_Backbone",
                            "Prot_Sidechain",
                            "Prot_All_Heavy",
                            "Lig_wrt_Protein",
```

```
                "Lig_wrt_Ligand",
            ],
        )

        # Estrae un nome "leggibile" per il sistema
        parent_dir = os.path.dirname(root)
        ligand_name = os.path.basename(parent_dir).split("-")[0].replace("data1_", "")

        # Estrai la colonna Lig_wrt_Protein
        if "Lig_wrt_Protein" in df.columns:
            all_lig_rmsd_data[ligand_name] = df["Lig_wrt_Protein"]
            print(f"Extracted 'Lig_wrt_Protein' for: {ligand_name}")
        else:
            print(f"Column 'Lig_wrt_Protein' not found in {file_path}")

    except pd.errors.EmptyDataError:
        print(f"The file {file_path} is empty or does not contain valid data.")
    except FileNotFoundError:
        print(f"File not found: {file_path}")
    except Exception as e:
        print(f"Error processing file {file_path}: {e}")

# Controllo: abbiamo raccolto qualcosa?
if not all_lig_rmsd_data:
    print("\nNessun dato RMSD raccolto (dizionario vuoto). Controlla base_path.")
    return

# Riepilogo dati raccolti
print("\nCollected Lig_wrt_Protein RMSD Data:")
for ligand, rmsd_series in all_lig_rmsd_data.items():
    print(f"{ligand}: Series with {len(rmsd_series)} data points")

# Converte il dizionario in matrice (ogni riga = serie RMSD di un sistema)
rmsd_data_list = [rmsd_series.values for rmsd_series in all_lig_rmsd_data.values()]
rmsd_data_matrix = np.vstack(rmsd_data_list)

# Calcola la matrice di similarità coseno tra le serie temporali RMSD
similarity_matrix = cosine_similarity(rmsd_data_matrix)

ligand_names = list(all_lig_rmsd_data.keys())
rmsd_similarity_matrix = pd.DataFrame(
    similarity_matrix, index=ligand_names, columns=ligand_names
)
```

```
    # Mostra la matrice
    print("\nMatrice di similarità RMSD (Lig_wrt_Protein):")
    display(rmsd_similarity_matrix)

    # Salva su CSV
    rmsd_similarity_matrix.to_csv(output_csv_path, index=True)
    print(f"\nMatrice di similarità RMSD salvata in: {output_csv_path}")

    # Heatmap
    fig, ax = plt.subplots(figsize=(10, 8))
    sns.heatmap(rmsd_similarity_matrix, cmap="viridis", annot=True, fmt=".2f", linewidths=0.5, ax=ax)
    plt.title("Heatmap delle Similarità RMSD Lig_wrt_Protein")
    plt.tight_layout()
    plt.show()
```

```
import os
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics.pairwise import cosine_similarity

def analisi_rmsf_protein_ca(
    base_path="/content/drive/MyDrive/tesi/",
    output_csv_path="/content/drive/MyDrive/tesi/rmsf_similarity_matrix.csv"
):
    """
    Analisi RMSF proteina (CA per residuo):
    - cerca tutti i file P_RMSF.dat sotto base_path
    - estrae la colonna 'CA' (RMSF per residuo)
    - costruisce la matrice di similarità (coseno) tra i profili per residuo
    - salva la matrice in CSV
    - mostra la matrice e la heatmap
    """

    file_name = "P_RMSF.dat"
    all_prot_rmsf_data = {}

    # Cerca ricorsivamente tutti i P_RMSF.dat
    for root, dirs, files in os.walk(base_path):
        for file in files:
            if file == file_name:
                file_path = os.path.join(root, file)
                print(f"Processing data from: {file_path}")
```

```
try:
    df = pd.read_csv(
        file_path,
        sep=r"\s+",
        skiprows=[0],
        index_col=0,
        names=[
            "Residue",
            "Chain",
            "ResName",
            "LigandContact",
            "CA",
            "Backbone",
            "Sidechain",
            "All_Heavy",
            "B-factor",
        ],
    )

    # Nome “leggibile” del sistema, come nel RMSD
    parent_dir = os.path.dirname(root)
    ligand_name = (
        os.path.basename(parent_dir)
        .split("-")[0]
        .replace("data1_", "")
    )

    if "CA" in df.columns:
        # Manteniamo i valori di RMSF CA per ciascun residuo (nessuna media)
        residue_rmsf_ca = df["CA"]
        # Dizionario: {ligand_name: {residuo: valore RMSF}}
        all_prot_rmsf_data[ligand_name] = residue_rmsf_ca.to_dict()
        print(f"Extracted 'CA' RMSF per residue for: {ligand_name}")
    else:
        print(f"Column 'CA' not found in {file_path}")

except pd.errors.EmptyDataError:
    print(f"The file {file_path} is empty or does not contain valid data.")
except FileNotFoundError:
    print(f"File not found: {file_path}")
except Exception as e:
    print(f"Error processing file {file_path}: {e}")
```

```
    # Controllo: abbiamo raccolto qualcosa?
    if not all_prot_rmsf_data:
        print("\nNessun dato RMSF raccolto (dizionario vuoto). Controlla base_path.")
        return

    print("\nCollected Protein RMSF (CA) Data per Residue:")
    for ligand, rmsf_data in all_prot_rmsf_data.items():
        print(f"{ligand}: Dictionary with {len(rmsf_data)} residues")

    # Dizionario -> DataFrame: colonne = ligandi, righe = residui
    rmsf_df = pd.DataFrame(all_prot_rmsf_data).fillna(0)

    # Similarità coseno tra file (ligandi): usiamo le colonne come vettori → trasposta
    similarity_matrix = cosine_similarity(rmsf_df.T)

    file_names = list(all_prot_rmsf_data.keys())
    rmsf_similarity_matrix = pd.DataFrame(
        similarity_matrix, index=file_names, columns=file_names
    )

    print("\nMatrice di similarità RMSF (CA):")
    display(rmsf_similarity_matrix)

    # Salva CSV
    rmsf_similarity_matrix.to_csv(output_csv_path, index=True)
    print(f"\nMatrice di similarità RMSF salvata in: {output_csv_path}")

    # Heatmap
    fig, ax = plt.subplots(figsize=(10, 8))
    sns.heatmap(
        rmsf_similarity_matrix,
        cmap="Blues",
        annot=True,
        fmt=".2f",
        linewidths=0.5,
        ax=ax,
    )
    plt.title("Heatmap delle Similarità RMSF (CA)")
    plt.tight_layout()
    plt.show()
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

```
from sklearn.metrics.pairwise import cosine_similarity

def analisi_sse_cosine(
    output_csv_path="/content/drive/MyDrive/tesi/sse_similarity_matrix.csv"
):
    """
    Analisi SSE (Secondary Structure Elements) tramite similarità coseno:
    - utilizza i valori medi di Helix, Strand, Total_SSE
    - costruisce la matrice di similarità
    - salva CSV
    - mostra matrice e heatmap
    """

    # Dati estratti dai report Desmond
    sse_data = {
        'ST44': {'Helix': 28.16, 'Strand': 7.64, 'Total_SSE': 35.80},
        'ST45': {'Helix': 29.05, 'Strand': 8.04, 'Total_SSE': 37.09},
        'STH2': {'Helix': 28.68, 'Strand': 8.73, 'Total_SSE': 37.41},
        'STH3': {'Helix': 28.34, 'Strand': 8.38, 'Total_SSE': 36.72}
    }

    # DataFrame
    sse_df = pd.DataFrame(sse_data).T
    print("Dati SSE (Helix / Strand / Total_SSE):")
    display(sse_df)

    # Similarità coseno
    similarity_matrix = cosine_similarity(sse_df)
    similarity_df = pd.DataFrame(similarity_matrix, index=sse_df.index, columns=sse_df.index)

    print("\nMatrice di similarità SSE (cosine similarity):")
    display(similarity_df)

    # Salvataggio CSV
    similarity_df.to_csv(output_csv_path, index=True)
    print(f"\nMatrice SSE salvata in: {output_csv_path}")

    # Heatmap
    plt.figure(figsize=(6,5))
    sns.heatmap(
        similarity_df,
        annot=True,
        cmap="viridis",
        fmt=".4f",
```

```
        cbar_kws={'label': 'Cosine Similarity'}
    )
    plt.title("Protein Secondary Structure Similarity Matrix (Cosine)")
    plt.tight_layout()
    plt.show()
```

```
import os
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics.pairwise import cosine_similarity

def analisi_hbond_similarity(
    base_path="/content/drive/MyDrive/tesi/",
    output_csv_path="/content/drive/MyDrive/tesi/hbond_similarity_matrix.csv"
):
    """
    Analisi H-Bond (Protein–Ligand):
    - cerca i file PL-Contacts_HBond.dat nelle cartelle data1_*
    - per ogni sistema, calcola la frequenza delle interazioni H-Bond per residuo
    - costruisce la matrice di similarità (coseno) tra i profili residuo–frequenza
    - salva la matrice in CSV
    - mostra matrice e heatmap
    """

    file_name = "PL-Contacts_HBond.dat"
    all_hbond_data = {}

    # Scorri tutte le sottocartelle di base_path
    for folder in os.listdir(base_path):
        if not folder.startswith("data1_"):
            continue  # ignoriamo cartelle che non sono dei sistemi

        specific_folder = folder  # es. data1_ST44-20231006T151539Z-001
        system_root = os.path.join(base_path, specific_folder)

        # es. "data1_ST44"
        main_subfolder = specific_folder.split("-")[0]

        # Percorso atteso del file H-Bond:
        # /tesi/data1_ST44-.../data1_ST44/raw-data/PL-Contacts_HBond.dat
        file_path = os.path.join(system_root, main_subfolder, "raw-data", file_name)
```

```
    print(f"\nVerifica del file: {file_path}")

    if os.path.exists(file_path):
        try:
            df = pd.read_csv(
                file_path,
                sep=r"\s+",
                skiprows=[0],  # salta riga header di Desmond
                index_col=None,
                names=["Frame", "Residue", "Chain", "ResName", "LigandFragment"]
            )

            print("File letto correttamente. Prime 5 righe:")
            display(df.head())

            if not df.empty:
                # conteggio contatti per residuo
                residue_counts = df["Residue"].value_counts()

                # nome del sistema (ST44, ST45, STH2, STH3, ...)
                ligand_name = main_subfolder.replace("data1_", "")

                all_hbond_data[ligand_name] = residue_counts.to_dict()

                print("\nResidui unici trovati nelle interazioni H-Bond:")
                print(df["Residue"].unique())

                print("\nConteggio delle interazioni per ogni residuo:")
                display(residue_counts)

                print("\nNomi di residui unici trovati nelle interazioni H-Bond:")
                print(df["ResName"].unique())
            else:
                print(f"Il file {file_path} è vuoto o non contiene dati validi.")

        except pd.errors.EmptyDataError:
            print(f"Il file {file_path} è vuoto o non contiene dati validi.")
        except Exception as e:
            print(f"Errore durante la lettura o l'elaborazione del file {file_path}: {e}")
    else:
        print(f"File non trovato: {file_path}")

# Controllo: abbiamo raccolto qualcosa?
if not all_hbond_data:
```

```
    print("\nNessun dato H-Bond raccolto (dizionario vuoto). Controlla la struttura delle cartelle.")
    return

print("\nCollected H-Bond contact data (frequenze per residuo):")
for ligand, hbond_dict in all_hbond_data.items():
    print(f"{ligand}: {len(hbond_dict)} residui con almeno un contatto H-Bond")

# Dizionario -> DataFrame: righe = residui, colonne = sistemi
hbond_df = pd.DataFrame(all_hbond_data).fillna(0)

# Similarità coseno tra i sistemi: colonne = vettori -> trasposta
similarity_matrix = cosine_similarity(hbond_df.T)

ligand_names = list(all_hbond_data.keys())
hbond_similarity_matrix = pd.DataFrame(
    similarity_matrix, index=ligand_names, columns=ligand_names
)

print("\nMatrice di similarità H-Bond (cosine similarity sui profili residuo–frequenza):")
display(hbond_similarity_matrix)

# Salva CSV
hbond_similarity_matrix.to_csv(output_csv_path, index=True)
print(f"\nMatrice di similarità H-Bond salvata in: {output_csv_path}")

# Heatmap
fig, ax = plt.subplots(figsize=(10, 8))
sns.heatmap(
    hbond_similarity_matrix,
    cmap="Reds",
    annot=True,
    fmt=".2f",
    linewidths=0.5,
    ax=ax
)
plt.title("Heatmap delle Similarità delle Interazioni H-Bond")
plt.tight_layout()
plt.show()
```

```
import os
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

```
from sklearn.metrics.pairwise import cosine_similarity

def analisi_hydrophobic_similarity(
    base_path="/content/drive/MyDrive/tesi/",
    output_csv_path="/content/drive/MyDrive/tesi/hydrophobic_similarity_matrix.csv"
):
    """
    Analisi delle interazioni idrofobiche Protein–Ligand:
    - cerca i file PL-Contacts_Hydrophobic.dat nelle sottocartelle di base_path
    - per ogni sistema, calcola la frequenza (%) di interazione per residuo
    - costruisce la matrice di similarità (coseno) tra i profili residuo–frequenza
    - salva la matrice in CSV
    - mostra matrice e heatmap
    """

    file_name = "PL-Contacts_Hydrophobic.dat"

    # Identifica tutte le sottocartelle "raw-data"
    folders = []
    for f in os.listdir(base_path):
        folder_path = os.path.join(base_path, f)
        if os.path.isdir(folder_path):
            subfolders = os.listdir(folder_path)
            for subfolder in subfolders:
                subfolder_path = os.path.join(folder_path, subfolder, "raw-data")
                if os.path.isdir(subfolder_path):
                    folders.append(subfolder_path)

    all_hydrophobic_interactions = {}  # {nome_sistema: {Residue: freq%}}

    # Carica i dati da ciascun file e calcola la frequenza di interazione
    for folder in folders:
        file_path = os.path.join(folder, file_name)
        if os.path.exists(file_path):
            print(f"\nCaricamento dati da: {file_path}")
            try:
                df = pd.read_csv(
                    file_path,
                    sep=r"\s+",
                    skiprows=[0],  # salta riga header di Desmond
                    index_col=None,
                    names=["Frame", "Residue", "Chain", "ResName", "LigandFragment"],
                )
```

```
            folder_name = os.path.basename(os.path.dirname(folder))

            if not df.empty:
                # Conteggio interazioni per residuo
                residue_counts = df["Residue"].value_counts()

                # Numero totale di frame
                total_frames = df["Frame"].nunique()
                if total_frames > 0:
                    # Frequenza (%) di contatto per residuo
                    residue_frequency = (residue_counts / total_frames) * 100.0
                    all_hydrophobic_interactions[folder_name] = residue_frequency.to_dict()

                    print(f"Residui con contatti idrofobici per {folder_name}:")
                    display(residue_frequency)
                else:
                    print(f"Nessun frame trovato in {file_path}")
            else:
                print(f"File {file_path} vuoto o senza dati validi.")

        except pd.errors.EmptyDataError:
            print(f"Il file {file_path} è vuoto o non contiene dati validi.")
        except Exception as e:
            print(f"Errore durante l'elaborazione del file {file_path}: {e}")
    else:
        print(f"File non trovato: {file_path}")

# Controllo: abbiamo raccolto qualcosa?
if not all_hydrophobic_interactions:
    print("\nNessun dato idrofobico raccolto (dizionario vuoto). Controlla la struttura delle cartelle.")
    return

print("\nCollected Hydrophobic Interaction Data (frequenze % per residuo):")
for system, freq_dict in all_hydrophobic_interactions.items():
    print(f"{system}: {len(freq_dict)} residui con almeno un contatto idrofobico")

# Dizionario -> DataFrame: righe = residui, colonne = sistemi
interaction_df = pd.DataFrame(all_hydrophobic_interactions).fillna(0)

# Vogliamo similarità tra sistemi (colonne = vettori) -> trasposta
interaction_df = interaction_df.T

# Matrice di similarità coseno
similarity_matrix = cosine_similarity(interaction_df)
```

```
    file_names = list(all_hydrophobic_interactions.keys())
    hydrophobic_similarity_matrix = pd.DataFrame(
        similarity_matrix, index=file_names, columns=file_names
    )

    print("\nMatrice di similarità delle interazioni idrofobiche:")
    display(hydrophobic_similarity_matrix)
     # Salva CSV
    hydrophobic_similarity_matrix.to_csv(output_csv_path, index=True)
    print(f"\nMatrice di similarità Idrofobica salvata in: {output_csv_path}")

    # Heatmap
    fig, ax = plt.subplots(figsize=(10, 8))
    sns.heatmap(
        hydrophobic_similarity_matrix,
        cmap="Blues",
        annot=True,
        fmt=".2f",
        linewidths=0.5,
        ax=ax,
    )
    plt.title("Heatmap delle Similarità delle Interazioni Idrofobiche")
    plt.tight_layout()
    plt.show()
```

```
import os
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics.pairwise import cosine_similarity

def analisi_waterbridge_similarity(
    base_path="/content/drive/MyDrive/tesi/",
    output_csv_path="/content/drive/MyDrive/tesi/waterbridge_similarity_matrix.csv"
):
    """
    Analisi delle interazioni WaterBridge Protein–Ligand:
    - cerca i file PL-Contacts_WaterBridge.dat nelle sottocartelle di base_path
    - per ogni sistema, calcola la frequenza (%) di interazione per residuo
```

```
- costruisce la matrice di similarità (coseno) tra i profili residuo–frequenza
- salva la matrice in CSV
- mostra matrice e heatmap
"""

file_name = "PL-Contacts_WaterBridge.dat"

# Identifica tutte le sottocartelle che contengono "raw-data"
folders = []
for f in os.listdir(base_path):
    folder_path = os.path.join(base_path, f)
    if os.path.isdir(folder_path):
        subfolders = os.listdir(folder_path)
        for subfolder in subfolders:
            subfolder_path = os.path.join(folder_path, subfolder, "raw-data")
            if os.path.isdir(subfolder_path):
                folders.append(subfolder_path)

all_waterbridge_interactions = {}  # {nome_sistema: {Residue: freq%}}

# Carica i dati da ciascun file e calcola la frequenza di interazione
for folder in folders:
    file_path = os.path.join(folder, file_name)
    if os.path.exists(file_path):
        print(f"\nCaricamento dati da: {file_path}")
        try:
            df = pd.read_csv(
                file_path,
                sep=r"\s+",
                skiprows=[0],  # salta riga header di Desmond
                index_col=None,
                names=[
                    "Frame",
                    "Residue",
                    "Chain",
                    "ResName",
                    "AtomName",
                    "LigandFragment",
                    "LigandAtom",
                ],
            )

            folder_name = os.path.basename(os.path.dirname(folder))
```

```
            if not df.empty:
                # Conteggio interazioni per residuo
                residue_counts = df["Residue"].value_counts()

                # Numero totale di frame
                total_frames = df["Frame"].nunique()
                if total_frames > 0:
                    # Frequenza (%) di contatto per residuo
                    residue_frequency = (residue_counts / total_frames) * 100.0
                    all_waterbridge_interactions[folder_name] = residue_frequency.to_dict()

                    print(f"Residui con contatti WaterBridge per {folder_name}:")
                    display(residue_frequency)
                else:
                    print(f"Nessun frame trovato in {file_path}")
            else:
                print(f"File {file_path} vuoto o senza dati validi.")

        except pd.errors.EmptyDataError:
            print(f"Il file {file_path} è vuoto o non contiene dati validi.")
        except Exception as e:
            print(f"Errore durante l'elaborazione del file {file_path}: {e}")
    else:
        print(f"File non trovato: {file_path}")

# Controllo: abbiamo raccolto qualcosa?
if not all_waterbridge_interactions:
    print("\nNessun dato WaterBridge raccolto (dizionario vuoto). Controlla la struttura delle cartelle.")
    return

print("\nCollected WaterBridge Interaction Data (frequenze % per residuo):")
for system, freq_dict in all_waterbridge_interactions.items():
    print(f"{system}: {len(freq_dict)} residui con almeno un contatto WaterBridge")

# Dizionario -> DataFrame: righe = residui, colonne = sistemi
interaction_df = pd.DataFrame(all_waterbridge_interactions).fillna(0)

# Vogliamo similarità tra sistemi (colonne = vettori) -> trasposta
interaction_df = interaction_df.T

# Matrice di similarità coseno
similarity_matrix = cosine_similarity(interaction_df)

file_names = list(all_waterbridge_interactions.keys())
```

```
    waterbridge_similarity_matrix = pd.DataFrame(
        similarity_matrix, index=file_names, columns=file_names
    )

    print("\nMatrice di similarità delle interazioni WaterBridge:")
    display(waterbridge_similarity_matrix)

    # Salva CSV
    waterbridge_similarity_matrix.to_csv(output_csv_path, index=True)
    print(f"\nMatrice di similarità WaterBridge salvata in: {output_csv_path}")

    # Heatmap
    fig, ax = plt.subplots(figsize=(10, 8))
    sns.heatmap(
        waterbridge_similarity_matrix,
        cmap="Greens",
        annot=True,
        fmt=".2f",
        linewidths=0.5,
        ax=ax,
    )
    plt.title("Heatmap delle Similarità delle Interazioni WaterBridge")
    plt.tight_layout()
    plt.show()
```

```
import os
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics.pairwise import cosine_similarity

def analisi_pipi_similarity(
    base_path="/content/drive/MyDrive/tesi/",
    output_csv_path="/content/drive/MyDrive/tesi/pi_pi_similarity_matrix.csv"
):
    """
    Analisi delle interazioni π–π Protein–Ligand:
    - cerca i file PL-Contacts_Pi-Pi.dat nelle sottocartelle di base_path
    - per ogni sistema, calcola la frequenza (%) di interazione per residuo
      (residui coinvolti in Residue1 o Residue2)
    - costruisce la matrice di similarità (coseno) tra i profili residuo–frequenza
    - salva la matrice in CSV
```

```
- mostra matrice e heatmap
"""

file_name = "PL-Contacts_Pi-Pi.dat"

# Identifica tutte le sottocartelle che contengono "raw-data"
folders = []
for f in os.listdir(base_path):
    folder_path = os.path.join(base_path, f)
    if os.path.isdir(folder_path):
        subfolders = os.listdir(folder_path)
        for subfolder in subfolders:
            subfolder_path = os.path.join(folder_path, subfolder, "raw-data")
            if os.path.isdir(subfolder_path):
                folders.append(subfolder_path)

all_pi_pi_interactions = {}  # {nome_sistema: {Residue: freq%}}

# Carica i dati da ciascun file e calcola la frequenza di interazione
for folder in folders:
    file_path = os.path.join(folder, file_name)
    if os.path.exists(file_path):
        print(f"\nCaricamento dati da: {file_path}")
        try:
            df = pd.read_csv(
                file_path,
                sep=r"\s+",
                skiprows=[0],
                index_col=None,
                names=[
                    "Frame",
                    "Residue1",
                    "Chain1",
                    "ResName1",
                    "Residue2",
                    "Chain2",
                    "ResName2",
                    "Pi-Pi_Type",
                    "Distance",
                    "Angle",
                ],
            )

            folder_name = os.path.basename(os.path.dirname(folder))
```

```
            if not df.empty:
                # Unisci Residue1 e Residue2
                all_residues = pd.concat([df["Residue1"], df["Residue2"]])

                # Tieni solo identificatori numerici (escludi L-FRAG ecc.)
                all_residues_filtered = pd.to_numeric(all_residues, errors="coerce").dropna()

                residue_counts = all_residues_filtered.value_counts()

                total_frames = df["Frame"].nunique()
                if total_frames > 0:
                    residue_frequency = (residue_counts / total_frames) * 100.0
                    all_pi_pi_interactions[folder_name] = residue_frequency.to_dict()

                    print(f"Residui con contatti π–π per {folder_name}:")
                    display(residue_frequency)
                else:
                    print(f"Nessun frame trovato in {file_path}")
            else:
                print(f"File {file_path} vuoto o senza dati validi.")
        except pd.errors.EmptyDataError:
            print(f"Il file {file_path} è vuoto o non contiene dati validi.")
        except Exception as e:
            print(f"Errore durante l'elaborazione del file {file_path}: {e}")
    else:
        print(f"File non trovato: {file_path}")

# Controllo: abbiamo raccolto qualcosa?
if not all_pi_pi_interactions:
    print("\nNessun dato π–π raccolto (dizionario vuoto). Controlla la struttura delle cartelle.")
    return

print("\nCollected Pi–Pi Interaction Data (frequenze % per residuo):")
for system, freq_dict in all_pi_pi_interactions.items():
    print(f"{system}: {len(freq_dict)} residui con almeno un contatto π–π")

# Dizionario -> DataFrame
interaction_df = pd.DataFrame(all_pi_pi_interactions).fillna(0)

# Similarità tra sistemi (colonne = vettori) -> trasposta
interaction_df = interaction_df.T

# Matrice di similarità coseno
```

```
    similarity_matrix = cosine_similarity(interaction_df)

    file_names = list(all_pi_pi_interactions.keys())
    pi_pi_similarity_matrix = pd.DataFrame(
        similarity_matrix, index=file_names, columns=file_names
    )

    print("\nMatrice di similarità delle interazioni π–π:")
    display(pi_pi_similarity_matrix)

    # Salva CSV
    pi_pi_similarity_matrix.to_csv(output_csv_path, index=True)
    print(f"\nMatrice di similarità Pi–Pi salvata in: {output_csv_path}")

    # Heatmap
    fig, ax = plt.subplots(figsize=(10, 8))
    sns.heatmap(
        pi_pi_similarity_matrix,
        cmap="YlGnBu",
        annot=True,
        fmt=".2f",
        linewidths=0.5,
        ax=ax,
    )
    plt.title("Heatmap delle Similarità delle Interazioni Pi–Pi")
    plt.tight_layout()
    plt.show()
```

```
import os
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics.pairwise import cosine_similarity

def analisi_l_torsion_medie_colonna(
    cartella="/content/drive/MyDrive/tesi/test",
    output_csv_path="/content/drive/MyDrive/tesi/l_torsion_similarity_matrix.csv"
):
    """
    L-Torsion: similarità coseno tra i vettori delle MEDIE di ciascuna colonna
    dei file L-Torsion (.dat) presenti nella cartella indicata.
```

```
Ogni file è rappresentato dal vettore delle medie per colonna
(Prot_CA, Backbone, Sidechain, ecc.), ignorando l'andamento nel tempo.
"""

# Trova tutti i file .dat nella cartella
file_paths = [f for f in os.listdir(cartella) if f.endswith(".dat")]
file_paths.sort()

if not file_paths:
    print("Nessun file .dat trovato in:", cartella)
    return

data_dict = {}

# Carica i dati dai file .dat
for file in file_paths:
    path_completo = os.path.join(cartella, file)
    data = np.loadtxt(path_completo, skiprows=1)[:, 1:]  # salta colonna "Frame"
    data_dict[file] = data

# Nomi senza estensione
labels = [os.path.splitext(f)[0] for f in file_paths]
n = len(data_dict)

# Dimensione minima tra i file (numero colonne)
min_dim = min(data.shape[1] for data in data_dict.values())

# Uniforma alla stessa dimensione
file_list = [data[:, :min_dim] for data in data_dict.values()]

# Matrice di similarità
similarity_matrix = np.zeros((n, n))

for i in range(n):
    for j in range(n):
        v_i = file_list[i].mean(axis=0).reshape(1, -1)
        v_j = file_list[j].mean(axis=0).reshape(1, -1)
        similarity_matrix[i, j] = cosine_similarity(v_i, v_j)[0, 0]

# DataFrame
df = pd.DataFrame(similarity_matrix, index=labels, columns=labels)
print("Matrice di Similarità Coseno – L-Torsion (medie per colonna):")
display(df)
```

```
    # Salvataggio CSV
    df.to_csv(output_csv_path, index=True)
    print(f"\nMatrice di similarità L-Torsion salvata in: {output_csv_path}")

    # Heatmap
    plt.figure(figsize=(6, 5))
    sns.heatmap(df, annot=True, cmap="coolwarm", vmin=-1, vmax=1)
    plt.title("Similarità Coseno tra i file L-Torsion (medie per colonna)")
    plt.tight_layout()
    plt.show()
import os
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics.pairwise import cosine_similarity

def analisi_l_torsion_temporale(
    cartella="/content/drive/MyDrive/tesi/test",
    output_csv_path="/content/drive/MyDrive/tesi/l_torsion_timeseries_similarity_matrix.csv"
):
    """
    L-Torsion: similarità coseno tra le SERIE TEMPORALI delle medie per frame.
    Ogni file L-Torsion.dat è rappresentato da una curva temporale:
    per ogni frame si calcola la media delle colonne (Prot_CA, Backbone, ecc.).
    La similarità è calcolata tra queste curve (evoluzione temporale).
    """

    # Trova tutti i file .dat nella cartella
    file_paths = [f for f in os.listdir(cartella) if f.endswith(".dat")]
    file_paths.sort()

    if not file_paths:
        print("Nessun file .dat trovato in:", cartella)
        return

    data_dict = {}

    # Carica i dati dai file .dat (escludendo la colonna Frame)
    for file in file_paths:
        path_completo = os.path.join(cartella, file)
        data = np.loadtxt(path_completo, skiprows=1)[:, 1:]
        data_dict[file] = data
```

```
labels = [os.path.splitext(f)[0] for f in file_paths]
n = len(data_dict)

# Allinea il numero di colonne (alcuni file hanno 7 colonne, altri 8)
min_dim = min(data.shape[1] for data in data_dict.values())
file_list = [data[:, :min_dim] for data in data_dict.values()]

# Per ogni file: vettore tempo = media per riga (frame)
time_series_list = [data.mean(axis=1) for data in file_list]

# Controllo rapido (assumiamo stesso numero di frame in tutti i file)
lengths = {len(ts) for ts in time_series_list}
if len(lengths) != 1:
    print("Attenzione: i file L-Torsion non hanno lo stesso numero di frame.")
    print("Lunghezze trovate:", lengths)

# Matrice di similarità
similarity_matrix = np.zeros((n, n))
for i in range(n):
    for j in range(n):
        similarity_matrix[i, j] = cosine_similarity(
            time_series_list[i].reshape(1, -1),
            time_series_list[j].reshape(1, -1)
        )[0, 0]

df = pd.DataFrame(similarity_matrix, index=labels, columns=labels)
print("Matrice di Similarità Coseno – L-Torsion (serie temporali):")
display(df)

# Salvataggio CSV
df.to_csv(output_csv_path, index=True)
print(f"\nMatrice di similarità L-Torsion (serie temporali) salvata in: {output_csv_path}")

# Heatmap
plt.figure(figsize=(6, 5))
sns.heatmap(df, annot=True, cmap="coolwarm", vmin=-1, vmax=1)
plt.title("Similarità Coseno tra i file L-Torsion (serie temporali)")
plt.tight_layout()
plt.show()
```

```
import ipywidgets as widgets
from IPython.display import display, clear_output
```

```
# Riquadro dove compariranno i risultati (tabelle, grafici, testo)
output = widgets.Output()

# ========= CALLBACK: COSA FA OGNI BOTTONE =========

def run_rmsd(b):
    with output:
        clear_output()
        print("1 - RMSD Lig_wrt_Protein: analisi di similarità\n")
        analisi_rmsd_lig_wrt_protein()

def run_rmsf(b):
    with output:
        clear_output()
        print("2 - RMSF proteina (CA): analisi di similarità per residuo\n")
        analisi_rmsf_protein_ca()

def run_sse(b):
    with output:
        clear_output()
        print("3 - SSE (Helix / Strand / Total_SSE): similarità (coseno)\n")
        analisi_sse_cosine()

def run_hbond(b):
    with output:
        clear_output()
        print("4 - H-Bond protein–ligand: similarità sulle frequenze per residuo\n")
        analisi_hbond_similarity()

def run_hydrophobic(b):
    with output:
        clear_output()
        print("5 - Interazioni idrofobiche: similarità sulle frequenze per residuo\n")
        analisi_hydrophobic_similarity()

def run_waterbridge(b):
    with output:
        clear_output()
        print("6 - Interazioni WaterBridge: similarità sulle frequenze per residuo\n")
        analisi_waterbridge_similarity()

def run_pipi(b):
    with output:
```

```
        clear_output()
        print("7 - Interazioni π–π: similarità sulle frequenze per residuo\n")
        analisi_pipi_similarity()

def run_l_torsion_colonne(b):
    with output:
        clear_output()
        print("8 - L-Torsion: similarità (medie per colonna)\n")
        analisi_l_torsion_medie_colonna()

def run_l_torsion_temporale_btn(b):
    with output:
        clear_output()
        print("9 - L-Torsion: similarità (serie temporali delle medie per frame)\n")
        analisi_l_torsion_temporale()

# ========== CREAZIONE PULSANTI ==========

btn1 = widgets.Button(
    description="1 · RMSD Lig_wrt_Protein",
    layout=widgets.Layout(width='11%')
)

btn2 = widgets.Button(
    description="2 · RMSF proteina (CA)",
    layout=widgets.Layout(width='11%')
)

btn3 = widgets.Button(
    description="3 · SSE (Helix / Strand / Total_SSE)",
    layout=widgets.Layout(width='11%')
)

btn4 = widgets.Button(
    description="4 · H-Bond (Protein–Ligand)",
    layout=widgets.Layout(width='11%')
)

btn5 = widgets.Button(
    description="5 · Interazioni idrofobiche",
    layout=widgets.Layout(width='11%')
)

btn6 = widgets.Button(
```

```
    description="6 · Interazioni WaterBridge",
    layout=widgets.Layout(width='11%')
)

btn7 = widgets.Button(
    description="7 · Interazioni π–π",
    layout=widgets.Layout(width='11%')
)

btn8 = widgets.Button(
    description="8 · L-Torsion (medie per colonna)",
    layout=widgets.Layout(width='11%')
)

btn9 = widgets.Button(
    description="9 · L-Torsion (serie temporali)",
    layout=widgets.Layout(width='11%')
)

# Collega bottoni alle funzioni di callback
btn1.on_click(run_rmsd)
btn2.on_click(run_rmsf)
btn3.on_click(run_sse)
btn4.on_click(run_hbond)
btn5.on_click(run_hydrophobic)
btn6.on_click(run_waterbridge)
btn7.on_click(run_pipi)
btn8.on_click(run_l_torsion_colonne)
btn9.on_click(run_l_torsion_temporale_btn)

# ========= LAYOUT COMPLETO DELLA DASHBOARD =========

dashboard = widgets.VBox([
    widgets.HTML("<h2>Analisi di similarità – Tesi</h2><p>Seleziona l'analisi:</p>"),

    widgets.HBox([btn1, btn2, btn3, btn4, btn5, btn6, btn7, btn8, btn9]),

    widgets.HTML("<hr>"),
    output
])

display(dashboard)
```

7. Bibliografia

1. Bissantz, C., Kuhn, B. & Stahl, M. A Medicinal Chemist's Guide to Molecular Interactions. *J. Med. Chem.* **53**, 5061–5084 (2010).
2. Hollingsworth, S. A. & Dror, R. O. Molecular Dynamics Simulation for All. *Neuron* **99**, 1129–1143 (2018).
3. Martínez, L. Automatic Identification of Mobile and Rigid Substructures in Molecular Dynamics Simulations and Fractional Structural Fluctuation Analysis. *PLOS ONE* **10**, e0119264 (2015).