Sveučilište u Zagrebu

Prirodoslovno-matematički fakultet

Kemijski odsjek

Matej Kožić

In silico istraživanje utjecaja glikana i mutacija na rekombinantni enzim peroksidaza hrena

Zagreb, 2022.

Ovaj rad izrađen je na Zavodu za fizikalnu kemiju Kemijskog odsjeka Prirodoslovno-matematičkog fakulteta Sveučilišta u Zagrebu pod mentorstvom prof. dr. sc. Branimira Bertoše i neposrednim voditeljstvom dr. sc. Antuna Barišića, te je predan na natječaj za dodjelu Rektorove nagrade u akademskoj godini 2021./2022. Dio rezultata predstavljenih u ovom radu dobiveni su istraživanjem u sklopu Horizon 2020 projekta *Marilia* (šifra projekta: 952110).

Popis korištenih kratica:

HRP- divlja peroksidaza hrena

mHRP – rekombinantna preoksidaza hrena

dHRP – neglikozilirana divlja preoksidaza hrena

dmHRP – neglikozilirana rekombinantna preoksidaza hrena

PCA – engl. *Protein-fragment Complementation Assay*, analitička metoda

PPI – protein-protein interakcije

PDB – engl. *Protein Data Bank,* baza podataka proteinskih struktura

MD – molekularna dinamika, računalna metoda

CD – ciruklarni dokroizam, eksperimentalna metoda

QM/MM – engl. *Quantum Mechanics / Molecular Mechanics,* računalna metoda

MM – molekularna mehanika, računalna metoda

DFT- engl. *Density Functional Theory*, računalna metoda

PBC – engl. *Periodic Boundary Conditions*, tip uvjeta simulacijske kutije

RMSD- engl. *Root Mean Square Deviation*, statistička metoda analize

RMSF – engl. *Root Mean Square Fluctuation*, statistička metoda analize

FS – engl. *First Step*

PS – engl. *Previous Step*

HLH – engl. *Helix-Loop-Helix,* strukturni element proteina

TrajMap – mapa trajektorija, engl. *Trajectory Map*

Sadržaj

1. Uvod

Detaljno istraženi enzim peroksidaza hrena (engl. *Horseradish Peroxidase*, HRP) zahvaljujući svojim karakteristikama koje ga čine atraktivnim za biotehnološku primjenu poput: svojstava reakcije koju katalizira, šest lizina dostupnih za konjugiranje, terminalnim krajevima polipetidnih lanaca lako dostupnim za modifikacije i sl. nalazi široku primjenu u raznim granama biotehnologije. Taj enzim sastoji se od 308 aminokiselina i s hem kofaktorom koji je ključan za odvijanje reakcije. Noviju primjenu nalazi u sklopu metode komplementacije proteinskih fragmenata (engl. *Protein-fragment Complementation Assay*, PCA) koja se koristi za detekciju protein-protein interakcija (PPI).[1] Metoda se zasniva na sposobnosti enzima detektora, u ovom slučaju HRP, da se umjetno pocijepa na dvije jedinice koje se konjugiraju na proteine čija se interakcija želi istražiti. Ako dođe do PPI, dolazi i do rekonstitucije enzima detektora i njegova se katalitička aktivnost može izmjeriti[2]. Iako u načelu svaki enzim može biti korišten kao detektor, problem predstavlja stabilnost enzima nakon što se podjeli na dvije nezavisne jedinice kao i njegova sposobnost rekonstruiranja u jednu podjedinicu. U slučaju HRP, upravo u svrhu svladavanja navedenih problema, Martell i suradnici su 2016. godine metodom usmjerene evolucije razvili šesterostruki mutant koji se iskazao poboljšanom aktivnosti i stabilnosti u opisanom sklopu[1]. Time je izrađen rekombinantni HRP (mHRP), koji se pokazao bolje prilagođen opisanoj svrsi od divljeg tipa HRP-a. Shodno tome, mHRP je uspješno iskorišten za analizu intracelularnih protein-protein interakcija.[1]

U prirodi je HRP posttranslacijski modificiran s devet glikana[4], te je poznato kako je jedna od uloga glikozilacije stabilizacija enzima u vodenim otopinama.[3] Kada bi se enzim proizvodio u većim količinama u prokariotima, s obzirom da oni nemaju potrebnu staničnu mašineriju za izvršiti glikozilaciju putem posttranslacijske modifikacije, takav bi HRP bio neglikoziliran. Pošto se jednom od šest Martellovih mutacija gubi jedno mjesto glikozilacije (Asn255Asp), glikozilirani mHRP glikoziliran je na osam mjesta umjesto devet[4] kao kod HRP. Kako bi se dobila kompletna slika punog opsega utjecaja mutacija, sagledan je i utjecaj glikozilacije na obje varijante enzima, mutirani i divlji tip.

Termički raspad HRP enzima odvija se u dva koraka, preko intermedijernog stanja.[5,6] Njega karakterizira gubitak tercijarne strukture oko šupljine aktivnog mjesta, uz potpunu retenciju sekundarne strukture[5,6]. U sklopu ovog istraživanja opisani proces je istražen na molekularnoj razini računalnim metodama, te je također istražen utjecaj mutacija na njega.

Cilj istraživanja predstavljenog u ovom radu jest istražiti utjecaj mutacija i glikolizacije na stabilnost enzima HRP. Računalnim metodama istražen je divlji tip i mutirani oblik enzima HRP u glikoziliranom i neglikoziliranom obliku. U prvom djelu istraživanja fokus je na ulozi glikozilacije na stabilnost i aktivnost enzima, dok je u drugom djelu istražena uloga Martellovih mutacija [1] na stabilnost enzima HRP, s fokusom na termičku stabilnost. Računalni rezultati su vrednovani usporedbom s literaturno dostupnim eksperimentalnim podacima.

U prvom djelu rada predstavljeni su postojeći literaturni podatci o HRP enzimu što uključuje njegovu strukturu, literaturni podatci o glikozilaciji, literaturni podatci o utjecaju mutacija te literaturni podatci o termičkom raspadu enzima. Također je predstavljena teorijska osnovica korištene metode simulacije molekulske dinamike, te korištenih metode analiza. U eksperimentalnom djelu opisana je *in silico* izgradnja sustava, način na koji su provedene simulacije molekulske dinamike te je opisana posebno izrađena skripta za programski jezik Python korištena za analizu simuliranih sustava. Rezultati i diskusija prezentirani su u zajedničkom poglavlju te je na temelju njih donesen zaključak po pitanju utjecaja glikolizacije i Martellovih mutacija na stabilnost mHRP i HRP enzima i procesa termičkog raspada.

2. Literaturni pregled

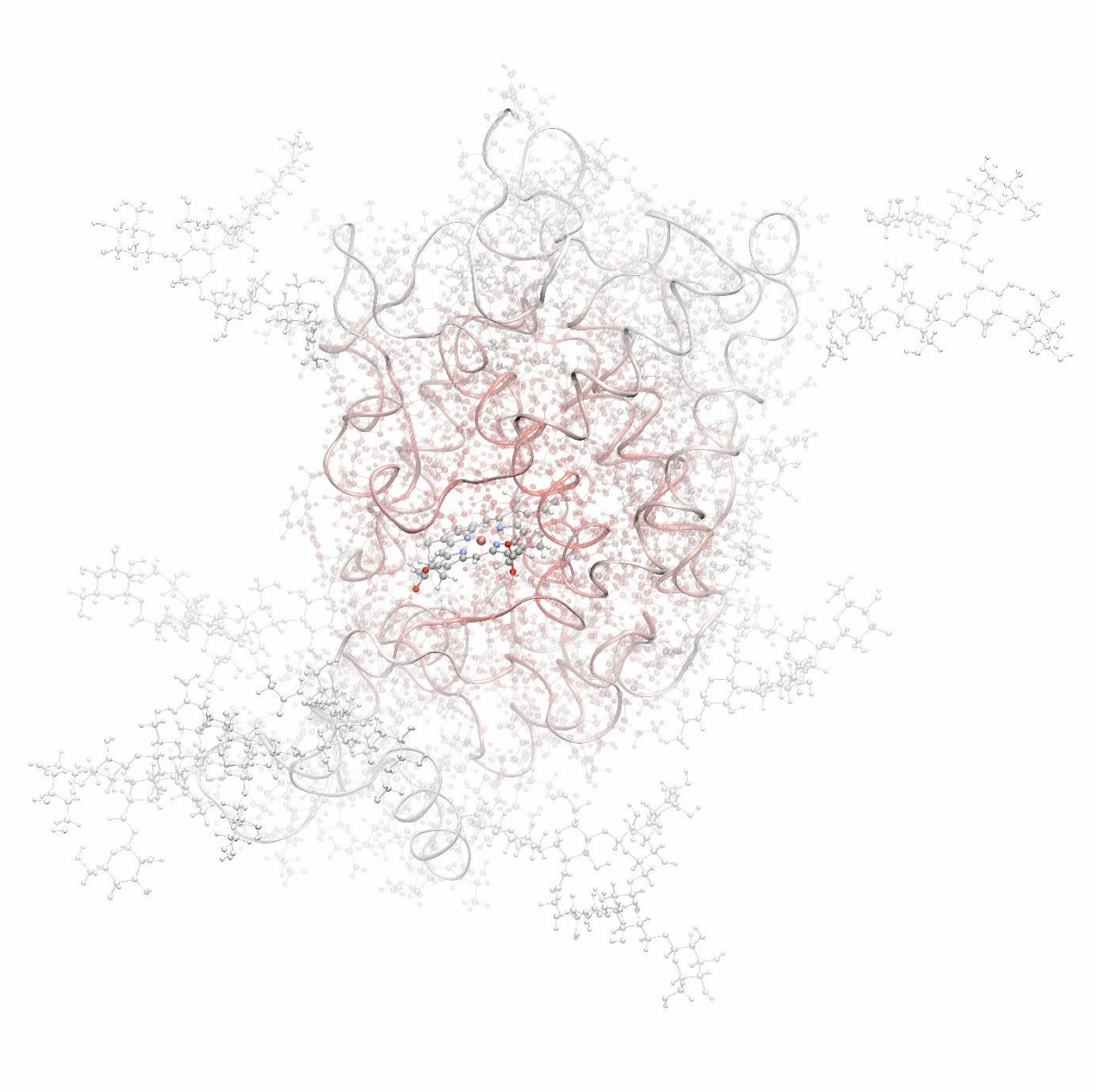
2.1. Peroksidaza hrena

Peroksidaza iz hrena (HRP, engl. *Horseradish Peroxidase*) je detaljno istraženi enzim koji svoju primjenu nalazi u mnogim granama biotehnologije. HRP protein često se koristi kao biosenzor u svrhu čišćenja otpadnih voda, organskoj sintezi, dijagnostici, terapeutici i slično.[4]

Općenito, peroksidaze su obitelj enzima koji se dijele u više kategorija i klasa, pa tako postoje tri superobitelji (engl. *superfamily*): biljne peroksidaze, životinjske peroksidaze i katalaze. Životinjske katalaze dijele se na dvije klase, a biljne peroksidaze dijele se na tri klase: peroksidaze prokariotskog podrijetla, peroksidaze fungalnog podrijetla te peroksidaze biljnog podrijetla. Enzim HRP je peroksidaza biljnog podrijetla, te se dodatno nalazi skupu izoenzima: A, B, C, D, E. Nadalje, pojedine obitelji izoenzima dolaze u svojim varijacijama pa tako postoje A1, A2 i A3, B1, B2 i B3, C1 i C2 te E1, E2, E3, E4, E5 i E6. Svi ti izoenzim dijele slična kemijska, fizikalna i kinetička svojstva, a razlikuju se najviše po ponašanju tijekom razdiobe na izoelektričnom gelu za fokusiranje[4]. Peroksidaza hrena istraživana u sklopu ovog rada je peroksidaza hrena izoenzim C1, koji će se dalje u radu referencirati kao HRP.

2.1.1. Struktura HRP

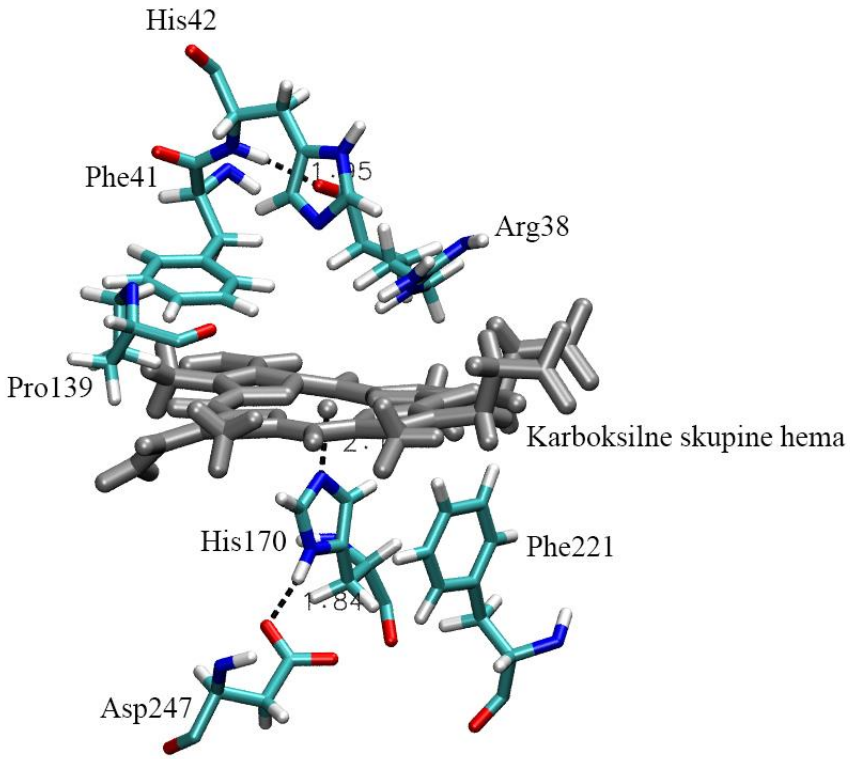
Enzim HRP, prikazan slikom 1, je monomerni protein sastavljen od 308 aminokiselina. Za njegovu aktivnost odgovorna je prostetička grupa hem (feriprotoporfirin IX) preko koje se događa reakcija koju HRP katalizira. U nativnom stanju HRP sadrži devet mjesta N glikozilacije preko asparaginskih bočnih ogranaka, četiri disulfidna mosta, te dva iona kalcija koji imaju ulogu strukturnih kofaktora.[4]



Slika 1. Enzim HRP, obojan radijalno na način da je boja intenzivnija bliže centra mase proteina. Modelom CPK prikazani su svi atomi proteina, hem kofaktora i glikani, a dodatno je modelom vrpce prikazana okosnica enzima. U sredini enzima vidljiv je otvor aktivnog mjesta s prostetičkom skupinom hem.

Enzim HRP se može podijeliti na dvije domene koje su odvojene aktivnim mjestom u kojem se nalazi prostetička skupina hem. S proksimalne strane skupine hem, na željezov ion iz hema kovalentno je vezan imidazolni prsten s His170 preko epsilon atoma dušika (N_{ε}). Time je definirana proksimalna, manja, domena, te distanlna, veća, domena. S distalne strane hema nalazi se aktivno mjesto u kojem se događa reakcija, te ga tvore aminokiseline: Arg38, Phe41, Pro139 te His42.[4,7] Navedene aminokiseline ponašaju se kao donori i akceptori protona u

kemijskim reakcijama, te su esencijalni za katalitičku aktivnost. Građa aktivnog mjesta prikazana je na slici 2. Asp247 ostvaruje vodikovu vezu s His170, a on s Phe221 ostvaruje π-π preklapajuće (engl. *stacking*) interakcije. Ion željeza(III) u hemu koordiniran je s četiri atoma dušika iz porfirinskog prstena, te s proksimalnim histidinom His170, što ostavlja jedno koordinacijsko mjesto slobodno s distalne strane na kojoj se događa kemijska reakcija. Koordinativna veza između imidazolnog prstena His170 i iona željeza kovalentnog je karaktera i može biti pocijepana u kiselim uvjetima što može dovesti do disocijacije hema. Uz koordinativnu vezu, hem također ostvaruje vodikove veze i van der Waalsove interakcije s okolnim bočnim ograncima koje ga dodatno stabiliziraju u šupljini aktivnog mjesta. Ulaz supstrata i vode u šupljinu aktivnog mjesta odvija se preko otvora koji se nalazi kod karboksilnih skupina hema i Arg38 (slika 1).[4]

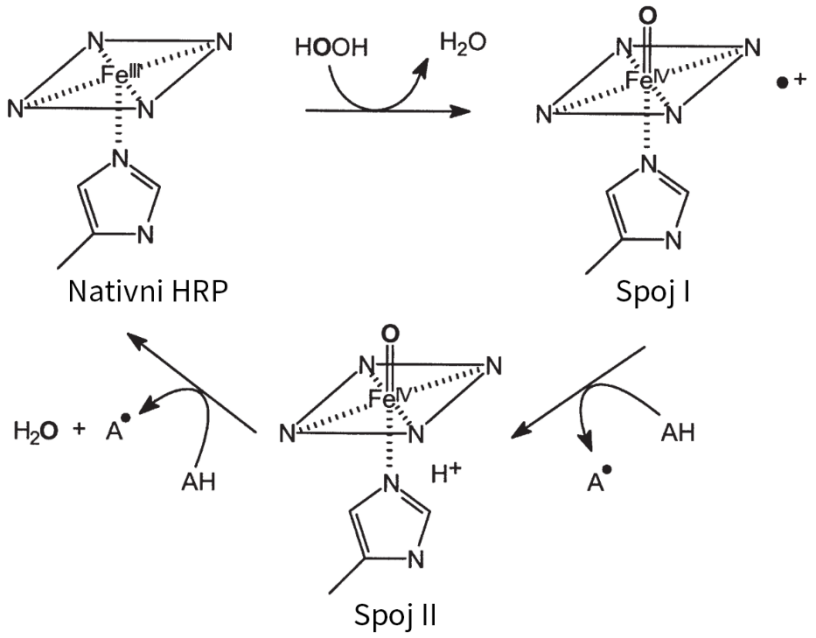


Slika 2. Građa aktivnog mjesta HRP enzima. Hem prostetička skupina obojan je sivom bojom.

2.1.2. Katalitička svojstva HRP

Enzim HRP vrši oksidaciju elektron donora s vodikova peroksida (H_20_2) te je ta reakcija cikličkog oblika s tri koraka: hem biva oksidiran s H_2O_2, stvara se tzv. spoj 1 (engl. *compound*

1), nakon čega se stvara se tzv. spoj 2 (engl. *compound 2*) koji se na kraju vraća u nativni hem (slika 3).[4]



Slika 3. Reakcijski mehanizam HRP. Slika je preuzeta i prilagođena iz izvora[4].

U prvom koraku jedan kisikov atom iz molekule peroksida tvori spoj 1 s oksiferil (Fe^{IV} =O) skupinom i porfirinskim radikalom, uz izdvajanje molekule vode. U drugom koraku spoj 1 oksidira supstrat s protonom, oblika AH, tako da dolazi do predaje radikala od porfirinskog prstena na spoja AH, te predaje protona od jedinke AH na porfirinski prsten. Ovime nastaje spoj 2, koji se zatim reducira predajom jednog elektrona na drugi supstrat AH pri čemu nastaje jedinka Fe(III), molekula vode od kisika oksiferila, te dva protona od dva AH supstrata. Molekula vode se otpušta s hema i time se on vraća u izvorno stanje. Sumarna reakcija je prevođenje molekule peroksida u dvije molekule vode, uz redukciju dva AH supstrata.[4]

2.1.3. Glikozilacija

Glikozilacija, kao najučestalija posttranslacijska modifikacija proteina, biološki je važan proces i njen utjecaj na strukturu i stabilnost brojnih glikoproteina je opširno proučavan[4,8,9]. Lee i suradnici[10] istraživali su utjecaj N-glikana na strukturna i dinamična svojstva glikoproteina temeljem analize dostupnih struktura glikoproteina unutar PDB baze podataka proteina (PDB, engl. *Protein Data Bank*) te simulacija molekularne dinamike (MD) glikoziliranih i neglikoziliranih proteina. Zaključili su da glikozilacija smanjuje dinamiku

proteina, ali ne inducira značajne promjene u konformaciji proteina. Također je pokazano da glikozilacija može stabilizirati proteine i čak utjecati na njihovo aktivno mjesto.[3,4]

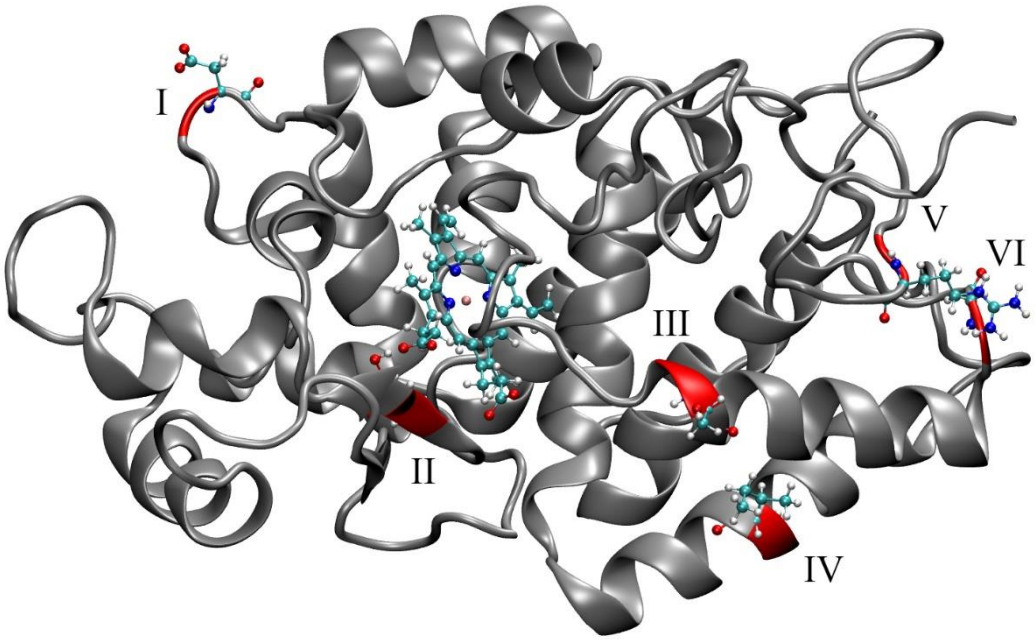
Glikani su naspram HRP volumno velike i teške skupine koje umanjuju fluktuacije prisutne u enzimu te pozitivno utječu na termodinamičku stabilizaciju enzima.[2] Prema tome, nije iznenađujuće da je eksperimentalno pokazano kako je glikozilirani oblik HRP dužeg životnog vijeka i veće katalitičke aktivnosti od neglikoziliranog oblika HRP.[8] Pored termodinamičke stabilizacije, prisutna je i kinetička stabilizacija tijekom smatanja enzima.[2] S obzirom da glikozilacija nije neophodna za pravilno smatanje i aktivnost enzima[2,8-12], smisleno je istraživati kako njen gubitak utječe na enzim.

Divlji tip enzima HRP glikoziliran je na devet mjesta preko Asn aminokiselina. Glikozilacija enzima slijedi Asn-X-Thr/Ser uzorak, gdje X predstavlja bilo koju aminokiselinu izuzev prolina ili aspartata. Na istraživanom HRP glikozilacije se nalazi na aminokiselinama: Asn13, Asn57, Asn158, Asn186, Asn198, Asn214, Asn268 i Asn286.[4,11] Sama veličina i oblik glikana ovisi o vrsti stanica u kojima se HRP sintetizira pri čemu ne moraju sva glikozilacijska mjesta biti glikozilirana niti na svakom mjestu treba biti glikan jednake veličine ili oblika. Uobičajeno je da se HRP izolira iz biljnih stanica koje sadrže 75-80 % N-glikana $Man_3GlcNAc_2$ s različitim grananjem na svakom N-glikozilacijskom mjestu.[13] Izolacija enzima HRP iz kvasca daje puno homogeniju raspodjelu glikana s osnovnim $Man_8GlcNAc_2$ glikanom.[14]

2.1.4. Mutacije

Enzimsko inženjerstvo prirodnih enzima se često koristi s ciljem povećanja aktivnosti ili stabilnosti enzima. Primjer enzimskog inženjerstva je rekombinantni HRP s četiri mutacije glikozilacijskih mjesta (Asn13Asp, Asn57Ser, Asn255Asp i Asn268Asp) koje dovodi do značajno veće termodinamičke stabilnosti i pojačane katalitičke aktivnosti.[15]

Kako bi se dobio enzim bolje prilagođen korištenju za detekciju protein-protein interakcija metodom komplementacije proteinskih fragmenata, Martell i suradnici su metodom usmjerene evolucije u kvascu priredili šesterostruki mutant (Leu299Arg, Asn255Asp, Asn175Ser, Arg93Gly, Pro78Ser i Thr21Ile) koji se iskazao boljom stabilnošću od divljeg tipa.[1] Posebno je zanimljiva mutacija Asn255Asp pošto ona dovodi do gubitka jednog od glikozilacijskih mjesta (Asn255) pa time mutirani oblik HRP posjeduje osam glikozilacijskih mjesta. Sve mutacije prikazane su crvenom bojom na slici 4.



Slika 4. Mjesta Martellovih mutacija[1] rekombinantnog HRP enzima (crvena boja): I – Asn255Asp, II – Asn175Ser, III – Pro78Ser, IV – Thr21Ile, V – Leu299Arg i VI – Arg93Gly.

2.1.5. Termički raspad HRP

Praćenjem spektra cirkularnog dikroizma (CD) ustvrđeno je kako se termički raspad enzima HRP odvija u dva koraka, preko intermedijernog stanja[5,6]. Prvi prijelaz nativne strukture N u intermedijarno stanje U' karakterizirano je promjenom u konformaciji oko hema, odnosno šupljine aktivnog mjesta. Navedeni prvi prijelaz N u U' opažen je u temperaturnom rasponu od 40 °C do 50 °C te u njemu sekundarne strukture ostaju sasvim očuvane i netaknute, no dolazi do promjene u tercijarnoj strukturi oko hema[5,6]. Intermedijarno stanje U' literaturno je nazvano engl. *pre-moletn globule*.[5] Drugi prijelaz iz U' u U'' karakteriziran je kompletnim gubitkom hema i drugim drastičnim promjenama koje dovode do gubitka aktivnosti. Taj prijelaz opažen je pri 74 °C.[5]

2.2. Računalne metode

2.2.1. Molekularno modeliranje

Molekularno modeliranje spoj je teorijskih modela i računalnih tehnika napravljenih u svrhu modeliranja molekula i imitiranja njihovog ponašanja u realnim sustavima. Potvrda dobivenih računalnih rezultata dobiva se povezivanjem makroskopskih svojstava simuliranih sustava s poznatim eksperimentalnim podatcima. Stoga, rezultati dobiveni računalnim metodama komplementiraju rezultate dobivene eksperimentalnim metodama na način da se koriste kao nadopuna eksperimentima u vidu potvrđivanja, opovrgavanja i objašnjavanja eksperimentalnih opažanja. Cilj ovog načina korištenja molekularnog modeliranja je smanjivanje troškova u razvoju novih lijekova i materijala, koji se često baziraju na principu pokušaja i pogreške. Molekularnim modeliranjem se mogu predvidjeti svojstva kao što su: struktura, pozicioniranje atoma unutar molekule, reaktivna mjesta, energije interakcije, raspodjela elektronske gustoće naboja, dipolni i multipolni momenti, vibracijske frekvencije, reaktivnost, kao i ostale spektroskopske veličine. U današnje doba, zahvaljujući velikim napredcima u korištenoj tehnologiji, računalne metode mogu se koristiti i za predviđanje eksperimentalnih rezultata. Primarna podjela računalnih metoda odnosi se na način na koji je modeliran sustav od interesa, pa se tako klasična molekulska mehanika temelji na Newtonovoj mehanici, a kvantno-mehaničke *ab initio* (lat. "od početka") metode temelje se na kvantnoj mehanici.[16]

Kako je *ab initio* metoda temeljena na kvantno mehaničkim opisima atoma i molekula, ne koriste se empirijski određeni parametri koji mogu uvoditi greške. Ona rješavanjem vremenski ovisne Schrödingerove jednadžbe danog sustava daje opis ponašanja sustava u vremenu. Prednost metode je što konvergira u egzaktno rješenje ako su aproksimacije prihvatljivog utjecaja, te može opisati interakcije elektronskih oblaka poput nastajanja i nestajanja kemijskih veza. Schrödingerova jednadžba je općenito rješiva samo za određene iznose energije te posjeduje analitičko rješenje samo za jednočestične sustave. Da bi se mogla riješiti Schrödingerova jednadžba za više-elektronske sustave, potrebno je uvesti Born-Oppenheimerovu aproksimaciju koja pretpostavlja da se gibanje jezgara atoma i gibanje elektrona koji ih okružuju može odvojiti. Uvođenje ove aproksimacije je moguće jer je tromost (masa) jezgara u usporedbi s masama elektrona znatno veća. Born-Oppenheimerova aproksimacija omogućava da ukupnu Schrödingerovu jednadžbu, kojom se opisuje sustav, separiramo na dvije neovisne funkcije (elektronsku valnu funkciju i valnu funkciju jezgara).

Nedostatak *ab initio* metoda je računalna zahtjevnost kada se usporede s manje preciznim metodama. Kompleksnost i trajanje računa eksponencijalno raste po svakoj iteraciji s brojem atoma u simuliranom sustavu.[16]

Pored *ab initio* metoda postoje i semiempirijske metode koje uvode dodatne aproksimacije u obliku eksperimentalno dobivenih parametara kako bi se smanjila kompleksnost računa, a samim time i vrijeme trajanja izračuna. Vrlo popularna metoda je teorija funkcionala gustoće, engl. *Density Functional Theory* (DFT). Ta metoda elektronske oblake opisuje funkcionalom gustoće, čime se izbjegavaju računalno najzahtjevnije kalkulacije prisutne u *ab initio* metodama. Također, kod sustava koji su preveliki da bi se u potpunosti tretirali kvantno-mehanički, poput enzima i drugih bioloških makromolekula, koristi se spoj molekulske mehanike i kvantne mehanike (QM/MM) gdje se kvantnom mehanikom simulira mali dio sustava u kojem se zbiva reakcija od interesa (npr. radijus od 0,5 nm oko aktivnog mjesta u kojemu dolazi do reakcije), a ostatak sustava se simulira molekulskom mehanikom. [16]

Simulacije klasične molekulske mehanike (MM) bazirane su na promatranju sustava kao sklopa kuglica (koji predstavljaju atome) povezanih s oprugama (koji predstavljaju kovalentne veze). Pristup klasičnom molekulskom mehanikom idealan je za opis sustava gdje nisu od interesa specifične i ezoterične interakcije elektronskih oblaka (poput nastajanja i pucanja veza i sl.), nego su od interesa konformacije koje enzim poprima, utjecaj određenih skupina na dostupnost aktivnog mjesta, dostupnost vode određenim skupinama i slična fizikalna svojstva. U sklopu ovog istraživanja promatrane su konformacije koje enzim poprima, utjecaj glikozilacije na dostupnost određenih skupina, dostupnost aktivnog mjesta, te izloženost određenih skupina vodi. Za to je korištena tehnika simuliranja molekulskom dinamikom (MD) zbog svoje pouzdanosti i učestalosti korištenja za istraživanja ovog tipa.[16]

2.2.2. Molekulska dinamika

Klasična molekulska mehanika počiva na tri temelja: (I) sustav se opisuje u prostoru kroz tri prostorne koordinate za svaki atom, (II) svaki od atoma može se opisati kroz empirijske parametre koji odgovaraju njegovim svojstvima (radijus, naboj i sl.), te (III) interakcije između atoma mogu se opisati kroz izraze temeljene na klasičnoj mehanici poput harmonijskog potencijala za istezanje kovalentne veze ili savijanje valentnog kuta te Lennard-Jonnesovim ili Bückinghamovim potencijalom za opis van der Waalsovih interakcija, Coulombov izraza za elektrostaske interakcije i sl.[16]

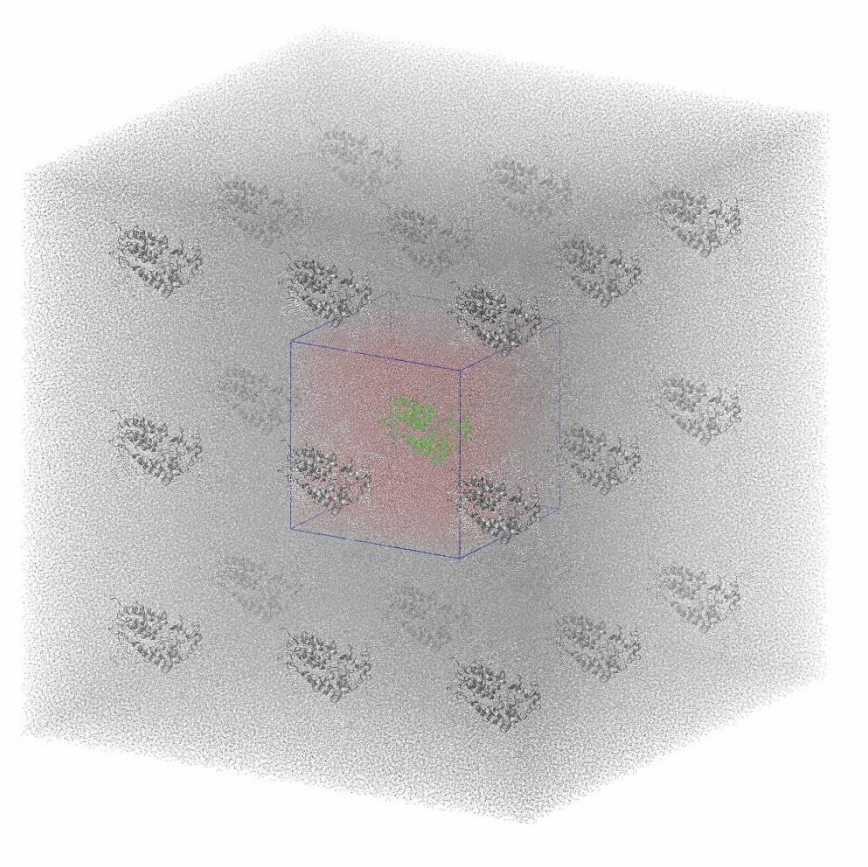
Klasična molekulska mehanika obuhvaća više metoda i primjena, a jedna od njih je molekulska dinamika (MD). Temelj molekulske dinamike je temperatura sustava preko koje se uvodi kinetička energija zbog koje svaka čestica dobiva određenu brzinu s obzirom na svoju masu. Pomoću polja sila izračunavaju se sile koje djeluju na svaku česticu sustava, pa se zatim preko dobivenog gradijenta izračunava nova koordinata svake čestice (kao da je svaka čestica pomaknuta u sumarnom smjeru sila koje osjeća). Odabir polja sila je krucijalan za pravilno izvođenje MD simulacija, jer se njime opišu sile svih mogućih interakcije unutar sustava. Različita polja sile prilagođena su za različite tipove sustava, pa su tako često korištena polja sila AMBER, GROMOS, OPLS, CHARMM i sl. Polja sila OPLS i GROMOS optimizirana su za male organske molekule, a AMBER i CHARMM prilagođeni su za biološke makromolekule, konkretno za proteine, RNA i DNA.[16]

Kako bi MD simulacije bile odraz realnih procesa, potrebno ih je uskladiti sa zakonima statističke mehanike i termodinamike. One definiraju odnos makroskopskih svojstava sustava poput tlaka (p), temperature (T), volumena (V) i množine(N) s mikroskopskim stanjima sustava poput definiranih pozicija atoma i njihovim količinama gibanja. Ansambl se može definirati kao višedimenzijski prostor svih mikrostanja dostupnih sustavu pri definiranim opservablama N, p, T i V. Različita makrostanja sustava definiraju se na način da se fiksira dio termodinamičkih parametara. Održavanjem konstantnog tlaka i temperature dobiva se izobarno-izotermni *NPT* ansambl, a održavanjem konstantnog volumena i temperature dobiva se kanonski *NVT* ansambl. Ta dva ansambla najčešće su korištena jer odgovaraju fizikalnim uvjetima gdje su konstantni temperatura i/ili tlak, no pored njih mogući su i drugi ansambli poput velekanonskog μVT gdje je kemijski potencijal održavan konstantnim i sl. Fiksiranje tih opservabli vrši se algoritmičkim termostatom i/ili barostatom koji ugađa brzine atoma kako bi mikroskopska slika sustava odgovarala makroskopskim opservablama.[16]

Jednom kada je sustav uravnotežen (ekvilibriran), moguće je započeti s produkcijskom fazom simulacije molekulske dinamike. Atomima se pripisuju nasumične vrijednosti brzina, najčešće prema Boltzmanovoj raspodjeli, te im se u prvom koraku predviđa pomak, izračunavaju nove vrijednosti pozicije, brzine i ubrzanja. To „kretanje“ atoma se simulira matematičkim algoritmima, od kojih su najpopularniji Verletov algoritam, „*leap-frog*“ algoritam i sl. Vremenski korak tog pomaka definiran je tako da mora biti manji nego najbrži pomak u sustavu. Za to se uzima perioda vibracije C-H veze koja iznosi oko 10 fs, te je stoga vremenski korak tijekom simulacije 1 fs. Uz korištenje SHAKE, LINCS[17] ili sličnih algoritama vibracija C-H se može fiksirati pa se mogu koristiti i veći vremenski koraci,

uglavnom oko 2 fs. Što je manji vremenski korak, potrebno je više komputacijskog vremena za postizanje simulacije određene duljine, pa je poželjno imati čim veći mogući korak. Jednom kada su izračunati novi položaji te nove brzine i ubrzanja atoma u sustavu, iz toga se izračunavaju sile te se na temelju njih pomiču svi atomi. Zatim slijedi primjenjivanje periodičkih uvjeta te korigiranje temperature i tlaka od strane termostata ili barostata, po potrebi. Na kraju se izračunavaju opservable, te se zapisuju zajedno s trajektorijama svih atoma za taj korak. To sačinjava jedan korak koji se iterira tijekom simulacije. Obično duljine simulacija traju od par nanosekundi do jedne mikrosekunde, ovisno o tipu i veličini sustava te računalnim resursima koji su na raspolaganju. Preko opservabli zapisanim u trajektorijama ustvrdi se je li sustav postigao ravnotežu, te se po potrebi produži ili ponovi simulacija.[16]

Sustav koji se simulira MD metodom najčešće uključuje makromolekulu od interesa okruženu molekulama otapala u kutiji periodičnih uvjeta (PBC, engl. *Periodic Boundary Conditions*) (slika 5). Njene preddefinirane granice su takvog svojstva da ako čestica izađe iz jedne stranice, npr. *x*, identična čestica ulazi u kutiju kroz suprotnu stranicu -*x*. To ima efekt kao da se simulira beskonačno veliki periodični sustav gdje se sve čestice slobodno gibaju, bez uvođenja nerealnih granica, kao što je slučaj u realnim otopinama. Prije početka same simulacije energija sustava se minimizira kako bi se postigla optimalna geometrija početne konformacije solvatirane makromolekule, te kako bi se uklonili eventualni artefakti solvatacije i steričke nepravilnosti. Nakon toga slijedi ekvilibracija temperature preko *NVT* ansambla, za kojim slijedi ekvilibracija tlaka preko *NPT* ansambl. Nad minimiziranim i ekvilibriranim sustavom pokreće se simulacije molekulske dinamike u broju definiranih koraka, te na kraju slijedi analiza rezultata iz dobivenih podataka. Ona uključuje promatranje trajektorija (vizualizaciju) i kvantitativne (statističke) analize.



Slika 5. Periodični uvjeti, PBC kutija proteina solvatiranog u vodi s prikazanim virtualnim periodičnim kutijama. Crvenom bojom označena je simulacijska kutija.

2.2.3. Analize simulacija

U sklopu analize simulacija važan korak predstavlja vizualizacija koja uključuje promatranje ponašanja trodimenzionalnog sustava u vremenu prema dobivenim trajektorijama simulacije, te donošenje zaključaka i njihovu interpretaciju na temelju kemijske intuicije i relevantnih znanja. Tako dobivene rezultate potrebno je kvantificirati kako bi se potvrdili i dokazali, a to se postiže statističkom analizom trajektorija. Metode analize korištene u sklopu ovog rada, pored vizualizacije, uključuju: RMSD (engl. *Root Mean Square Deviation*) analizu, RMSF (engl. *Root Mean Square Fluctuation*) analizu, analizu radijusa giracije te analizu mapa trajektorija za koje su u sklopu ovog rada napisane skripte u programskom jeziku Python, te su sastavni dio rada (dodatak).

RMSD analizom moguće je dobiti informacije o stabilnosti sustava u vremenu (t), odnosno tijekom simulacije. Izraz kojim se računa korijen kvadrata srednje vrijednosti pomaka (RMSD), u slučaju proteina, najčešće Cα ugljikovih atoma proteinske okosnice, u Euklidskom

prostoru prikazan je jednadžbom (1). U izrazu (1), *N* predstavlja broj Cα ugljikovih atoma po kojima se sumira, a $r_i(t)$ predstavlja koordinate atoma *i* u vremenu *t*.[18]

$$\rho^{RMSD}(t) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(r_i(t) - r_{i,ref})^2} \quad (1)$$

Dok se RMSD koristi za istraživanje stabilnosti cijelog enzima kroz vrijeme, RMSF daje informaciju o količini fluktuacija pojedinih aminokiselinskih ostataka (ili pojedinih atoma, ovisno kako se definira). Računa prema jednadžbi (2), gdje r_i predstavlja trodimenzionalne koordinate atoma. RMSF je koristan kao mjera fleksibilnosti aminokiselinskih ostataka, te uspoređivanjem više RMSF grafova mogu se dobiti informacije o dijelovima enzima u kojima je došlo do povećanja fluktuacija tijekom simulacije.[18]

$$\rho_i^{RMSF} = \sqrt{\langle (r_i - \langle r_i \rangle)^2 \rangle} \quad (2)$$

Radijus giracije je mjera kompaktnosti proteina, te također sadrži informaciju o stabilnosti enzima i eventualnim konformacijskim promjenama koje bi uzrokovale mijenjanje kompaktnosti, oblika, ili volumena. Računa se kao prikazano jednadžbom 3, gdje *M* predstavlja sveukupnu masu proteina, sastavljenog od *N* atoma, s koordinatama centra mase *R*.[18]

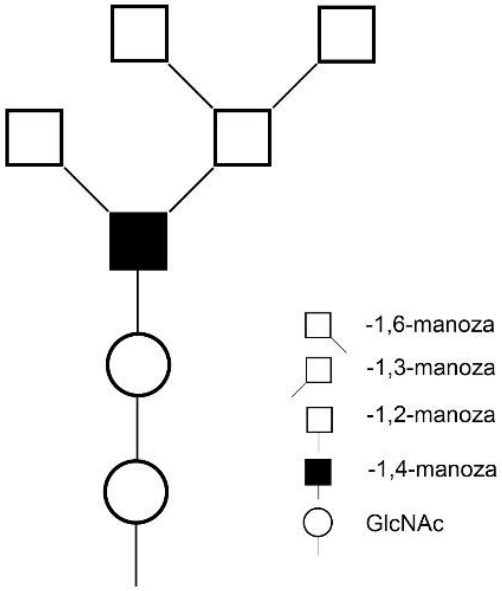
$$R_g = \sqrt{\frac{1}{M}\sum_{i=1}^{N} m_i(r_i - R)^2} \quad (3)$$

3. Eksperimentalne metode

3.1. *In silico* izgradnja sustava

Za modeliranje proteina korištena je kristalna struktura HRP enzima C1A izolirana iz *Armoracia rusticana* (pdb kod: 1h5a[19]). Četiri forme enzima izgrađene su *in silico* na temelju dostupne pdb strukture: (I) glikozilirani divlji tip (HRP), (II) glikozilirani rekombinantni enzim s Martellovim mutacijama (mHRP), (III) neglikozilirani divlji tip (dHRP) te (IV) neglikozilirani mutant (dmHRP). Dalje u tekstu navedene forme enzima će biti nazvane prema pripadnim kraticama.

U ovom istraživanju za glikozilaciju je korišten Man_5GlNAc_2 glikan[20] (slika 6) koji se ravnomjerno veže na sva glikozilacijska mjesta HRP enzima prilikom izolacije iz kvasca. Martellove mutacije su: Leu299Arg, Asn255Asp, Asn175Ser, Arg93Gly, Pro78Ser, te Thr21Ile.[1] Glikozilirane varijante HRP proteina glikozilirane su s navedenim glikanima na devet mjesta u slučaju HRP i osam mjesta u slučaju mHRP na način koji je opisan u literaturnom pregledu (poglavlje 2.1.3). Kako bi se istražila postojanost konformacije enzima, utjecaj temperature na konformaciju enzima, te proces termičkog razmatanja enzima, odrađeno je sveukupno osam simulacija molekulske dinamike. U tom djelu simulirane su samo glikozilirane HRP i mHRP pri četiri različitih temperature: 300 K (26,85 °C), 318 K (44,85 °C), 333 K (59,85 °C), te 353 K (79,85 °C). Temperature su odabrane sukladno literaturno dostupnom eksperimentalno istraženom dvofaznom odmatanju enzima koje se odvija preko intermedijernog stanja U' (poglavlje 2.1.5). Do prvog prijelaza dolazi pri temperaturama od 40 °C do 50 °C. Drugi prijelaz iz U' u U'' odvija se pri 74 °C, i kod njega dolazi do gubitka hema te posljedičnog gubitka aktivnost. Zbog korištenja empirijske računalne metode (MD simulacije), hem je kovalentno vezan s His170, te se zbog toga opisani efekt pri 74 °C ne može promatrati jer u MD simulacijama ne može doći do pucanja kovalentne veze. Utjecaj glikozilacije na strukturu i aktivnost HRP enzima istražen je pri 300 K usporedbom simuliranih struktura s njihovim neglikoziliranim varijantama (dHRP i dmHRP).



Slika 6. Shematski prikaz glikana Man_5GlNAc_2.

Priprema modeliranih sustava za računalne simulacije napravljena je korištenjem CHARMM-GUI mrežnog poslužitelja.[21-23] U skladu sa strukturom, uvedena su četiri cisteinska mosta na mjestima: Cys11-Cys91, Cys44-Cys49, Cys177-Cys209 te Cys97-Cys301. Nedostajuće aminokiseline C-terminalnog kraja, 307 i 308, također su modelirane u sklopu izgradnje sustava. U sklopu protonacije, svi arginini i lizini su protonirani i pozitivno nabijeni, dok su hidstidini protonirani na delta dušikovom atomu. Cisteini su neutralni (nenabijeni), a glutamati i aspartati deprotonirani i negativnog naboja. Završno, definirana je kovalentna veza između His170 i Fe^{2+} iona hema. Protonacija, solvatacija, neutralizacija i izgradnja PBC kutije također je napravljena korištenjem CHARMM-GUI mrežnog poslužitelja. Udaljenost ruba solvatacijske kutije od proteina iznosila je 2 nm, te je korišten eksplicitni model molekula vode TIP3P, a sustav je po potrebi neutraliziran dodatkom Cl^- iona. Konkretno, kod pripreme simulacija mHRP i dmHRP nije dodan niti jedan ion, dok je kod HRP i dHRP dodan jedan ion klora. Mutacije i glikani su uvedeni u potrebne sustave korištenjem CHRAMM-GUI poslužitelja u sklopu izgradnje sustava. Konačna veličina stranice kvadratne simulacijske kutije ovisila je o temperaturi i vrsti simuliranog sustava (tablica 1). Za simulacije je korišteno polje sila CHARMM36m[24].

Tablica 1. Dimenzije stranica kvadratnih simulacijskih kutija svake odrađene simulacije.

	HRP	**mHRP**	**dHRP**	**dmHRP**
300 K	11,37 nm	11,95 nm	10,80 nm	10,60 nm
318 K	11,43 nm	12,02 nm		
333 K	11,49 nm	12,08 nm		
353 K	11,56 nm	12,16 nm		

3.2. Simulacija molekulske dinamike

Prije produkcijske faze MD simulacije, odrađeno je tri koraka pred-produkcije, odnosno uravnoteženja sustava: minimizacija, *NVT* ekvilibracija, te *NPT* ekvilibracija. Minimizacijom je optimizirana geometrija sustava, te su otklonjene možebitne nepravilnosti i sterički konflikti do kojih je moglo doći tijekom izgradnje (i solvatacije) sustava. Korištena je metoda najstrmijeg spusta s konvergencijskim kriterijem energije od 1000 kJ mol^{-1} nm^{-1}.[18]

U sklopu ekvilibracije prvo je napravljena ekvilibracija u *NVT* ansamblu. Koristeći „*leap-frog*“ algoritam integratora odrađeno je sveukupno 0,6 ns ekvilibracije s vremenskim korakom od 1 fs. Prvih 0,5 ns, temperatura se podizala kontinuirano od 10 K do zadane temperature pripadne simulacije, dok je zadnju 0,1 ns odrađeno na konstantnoj pripadnoj temperaturi kako bi se sustav stabilizirao na željenoj temperaturi. Za temperaturnu regulaciju tijekom ekvilibracije korišten je Berendsen termostat[25]. Početne brzine specija generirane su prema Boltzmannovoj distribuciji pri temperaturi od 10 K, te je algoritamski otklonjena brzina translacije centra mase cijelog sustava.

NPT ekvilibracija nastavljena je na prethodno odrađenu *NVT* ekvilibraciju, te je analogno njoj korišten „*leap-frog*“ integrator. Ekvilibracija je trajala 0,5 ns s vremenskim korakom od 0,1 fs. Također je korišteno temperaturno sprezanje pri zadanoj temperaturi s Berendsen termostatom[25], te je dodatno tome korišteno i tlačno sprezanje s Berendsen izotropskim barostatom[25] pri tlaku od 1 bar.

Produkcijska faza MD simulacija provedena je u NPT ansamblu. Simulacije su trajale 500 ns s vremenskim korakom od 2 fs. Temperaturno sprezanje sustava postignuto je Nose-Hoover algoritmom termostata[26], pri prethodno definiranoj temperaturi. Tlačno sprezanje postignuto je Parinello-Rahman algoritmom barostata[27], izotropski, pri tlaku od 1 bar. Trajektorije su zapisivane svakih 10 ps, a analizirane su u koracima od 1 ns.

U svim simulacijama korišten je LINCS[17] algoritam za ograničavanje vibracija veza s atomom vodika. Za računanje dugodometnih elektrostatskih interakcija korištena je PME metoda[28], engl. *Particle-mesh Ewald*, s duljinom prekida (engl. *Cut-off*) Coulombovih interakcija od 1,2 nm, Fourier razmakom od 0,16 nm i Verlet shemom prekida. Van der Waalsove interakcije opisane su korištenjem Lennard-Jonesovog potencijala. Sve simulacije provedene su korištenjem GROMACS 2020.6 programskog paketa.[18, 29]

3.3. Analize trajektorija

RMSD (engl. *Root Mean Square Deviations*), RMSF (engl. *Root mean Square Fluctuations*), te analiza radijusa giracije napravljene su u GROMACS 2020.6 programskom paketu[18, 29]. Vizualizacija trajektorija izvršena je u programu VMD[30]. Informacija o količini molekula vode oko mjesta od interesa dobivena je s vlastitom skriptom napisanom za program VMD koja broji molekule vode na zadanoj udaljenosti od definirane specije tijekom simulacije (dodatak 1). Mjerenje torzijskog kuta odrađeno je vlastitom skriptom za VMD (dodatak 2), a mjerenje pomaka regija odrađeno je vlastitom skriptom za programski jezik Python (dodatci 8-10

Kako bi se dobile vizualiuacije trajektorija nazvane mape trajektorija (engl. *Trajectory Map*, kratično TrajMap) napisane su skripte u programskom jeziku Python (dodatci 9-11). Za napomenuti je kako su u literaturi slične vizualizacije referencirane kao engl. *Per-residue RMSD, RMSD heatmap* i dr.[31-33] Ulazna datoteka za skriptu su trajektorije jedne simulacije modificirane u VMD programu na način da je definirana samo okosnica proteina te poravnata i centrirana s funkcijom „*Align*“, te eksportirana u .pdb formatu. Shodno tome, ulazna datoteka je .pdb trajektorija svih kadrove poravnate okosnice proteina. U prvom koraku skripta pročita i obradi trajektoriju u oblik pogodan za rukovanje. U drugom koraku iz trajektorija se izračunava pomak centra mase okosnice u vremenu t u odnosu na referentno vrijeme t_{ref}, na način da se računa Euklidska udaljenost dvaju točaka u trodimenzionalnom koordinatnom sustavu, (x_{cm_t}, y_{cm_t}, z_{cm_t}) od (x_{cm_ref}, y_{cm_ref}, z_{cm_ref}). Izračunati pomaci se zapisuju kao lista trodimenzionalnih koordinata oblika (aminokiselina, vrijeme, pomak). U trećem koraku koordinate se sortiraju u matricu s x osi koja odgovara vremenu od 0 do 499 ns i y osi koja predstavlja aminokiseline od 0 do 307 (budući da Python indeksiranje počinje od 0 prva aminokiselina je rednog broja 0 a prvi kadar je rednog broja 0). U zadnjem koraku iz matrice se kreira toplinska karta na način da se vrijednosti pomaka prikažu na brojevnoj skali u boji.

Korištena su dva tipa mapa trajektorija, ovisno o tome koji kadar se uzima kao referentni kadar. U jednom slučaju referentni kadar je prvi kadar trajektorija (engl. *Frist step*, „FS“), pa je referentno stanje $s_{ref} = s_0$, dok je u drugom tipu referenti kadar prethodni kadar (engl. *Previous step*, „PS“), $s_{ref} = s_{t-1}$. Formula za računanje pomaka p centra mase aminokiseline a s pripadnim koordinatama centra mase (x,y,z), u vremenu t u odnosu na vrijeme t_{ref} prikazana je jednadžbom (4). U svim daljnjim vizualizacijama u radu korištena je isključivo FS mapa trajektorija gdje je referentni kadar početni kadar simulacije.

$$p(a,t,t_{ref}) = \sqrt{\left(x_{a,t} - x_{a,t_{ref}}\right)^2 + \left(y_{a,t} - y_{a,t_{ref}}\right)^2 + \left(z_{a,t} - z_{a,t_{ref}}\right)^2} \quad (4)$$

Razlog računanja pomaka centra mase, umjesto pomaka npr. Cα ugljikova atoma proteinske okosnice, je što na taj način rotacije i vibracije jedne aminokiseline daju smanjen signal na mapi trajektorija, jer tijekom njih ne dolazi do značajnog pomaka centra mase okosnice. To znači da je rezolucija po *z* osi, koja odgovara promjenama u enzimu, povećana, a eventualni šum od pomaka koji ne odgovaraju promjenama u konformaciji je umanjen.

FS mape trajektorija, gdje je referentan kadar prvi kadar u simulaciji, korištene su za vizualizaciju cijelog tijeka simulacije te prikaz evolucije konformacije kroz vrijeme (dodatak 14 slike D1 a). Iz takve mape moguće je odrediti redosljede pomaka regija u proteinu (dodatak 14 slika D2). Ako se na FS mapu nacrta RMSD graf (dobiven GROMACS programskim paketom), moguće je pratiti kakav doprinos ima individualna promjena u konformaciji na RMSD vrijednost enzima (dodatak 14 slika D14 b).

PS mape prikazuju magnitudu individualnog pomaka iz vremena t_{i-1}, u vrijeme t_i. Takvim pristupom moguće je vidjeti vrijeme i lokaciju svih individualnih pomaka (dodatak 14 slika D1 c). Mogu se uprosječiti svi pomaci svih aminokiselina u jednom vremenskom koraku (analogno gledanju mape kroz *y* os). Time dobivena krivulja sadrži informaciju o apsolutnoj količini pomaka koji su se dogodili iz kadra *i-1* u kadar *i*, što se aproksimativno može interpretirati kao „apsolutna stabilnost enzima u relativnom vremenu“. Ta krivulja nacrtana na FS mapu trajektorija može dati informaciju o tome kako koja individualna promjena utječe na sveukupnu stabilnost enzima (dodatak 14 slika D1 d). Drugim riječima, takvim prikazom vidljiv je doprinos individualnih konformacijskih promjena sveukupnom fluktuiranju centra mase okosnice (engl. „*backbone*“) enzima.

Kako su toplinske karte samo matrice prikazane skalom u boji, s njima je moguće računati. Mogu se uzimati prosjeci matrica, te se matrice mogu oduzimati kako bi se konkluzivno

pokazale sve razlike pomicanja okosnice proteina korištenih simulacija. Tako dobivena razlikovna mapa trajektorija prikaže se s divergentnom paletom boja pa su vidljive sve razlike u oduzetim simulacijama s pripadnim lokacijama u enzimu, vremenima i magnitudama pomaka (dodatak 14 slike D3 i D4). Regije koje su blizu nula odgovaraju regijama gdje su pomaci bili isti ili slični. Pozitivne regije odgovaraju mjestima i vremenima gdje su pomaci bili veći u simulaciji od koje se oduzima, a negativne regije odgovaraju dijelovima gdje su pomaci bili veći u matrici kojom se oduzima. Ova metoda iskazala se iznimno praktičnom za isticanje razlika u simulacijama jer pored samih razlika vidljivo je kada one nastupaju, kako se ponašaju u vremenu, te jesu li one prvenstveno povećane fluktuacije ili konformacijske promjene.

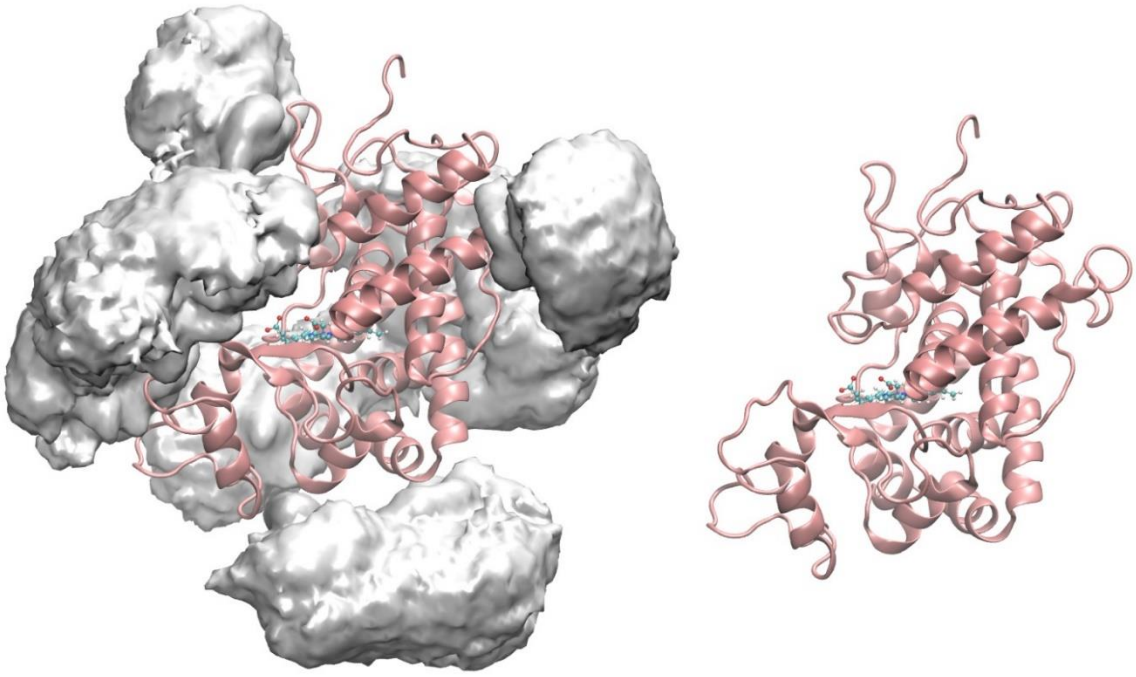
Završno, napisana je skripta koja iz matrice pomaka iz koje se dobiva toplinska karta izračunava ovisnost pomaka prosjeka definirane regije o vremenu (dodatak 10). Tom metodom grafički su prikazane magnitude pomaka određenih regija enzima tijekom simulacije (dodatak 14 slika D5).

Svi grafovi i mape trajektorija napravljene su korištenjem Matplotlib 3.4.3[34] biblioteke za programski jezik Python, u Spyder 5 integriranom znanstvenom razvojnom okružju za Python 3.9.[35,36] Korištene skripte za programski jezik Python priložene su dodatcima 3-10. Dodatno, priložena je modularizirana verzija skripti za izradu mapa trajektorija koja omogučava jednostavnu izradu mapa trajektorija kroz lokalno importiranje tog priloženog koda (dodatak 11). Dodatkom 12 dan je predložak za korištenje navedene biblioteke, te opisi funkcija iste.

4. Rezultati i diskusija

4.1. Utjecaj glikozilacije na stabilnost HRP enzima

Za istraživanje utjecaja glikozilacije napravljene su četiri simulacije koje su se međusobno razlikovale s obzirom na dvije varijable: divlji tip / mutirani te glikozilirani / neglikozilirani oblik enzima. Usporedba te četiri simulacije omogućila je istraživanje utjecaja glikozilacije u kontekstu i HRP i mHRP, a daje uvid u njihov međusoban odnos. Prikaz volumena koji zauzimaju glikani njihovim fluktuiranjem prikazan je na slici 7. Glikani sterički parcijalno zakrivaju otvor aktivnog mjesta, no pošto se radi o fleksibilnim skupinama koje konstantno fluktuiraju tijekom simulacija, ulaz vode i supstrata i dalje je moguć.

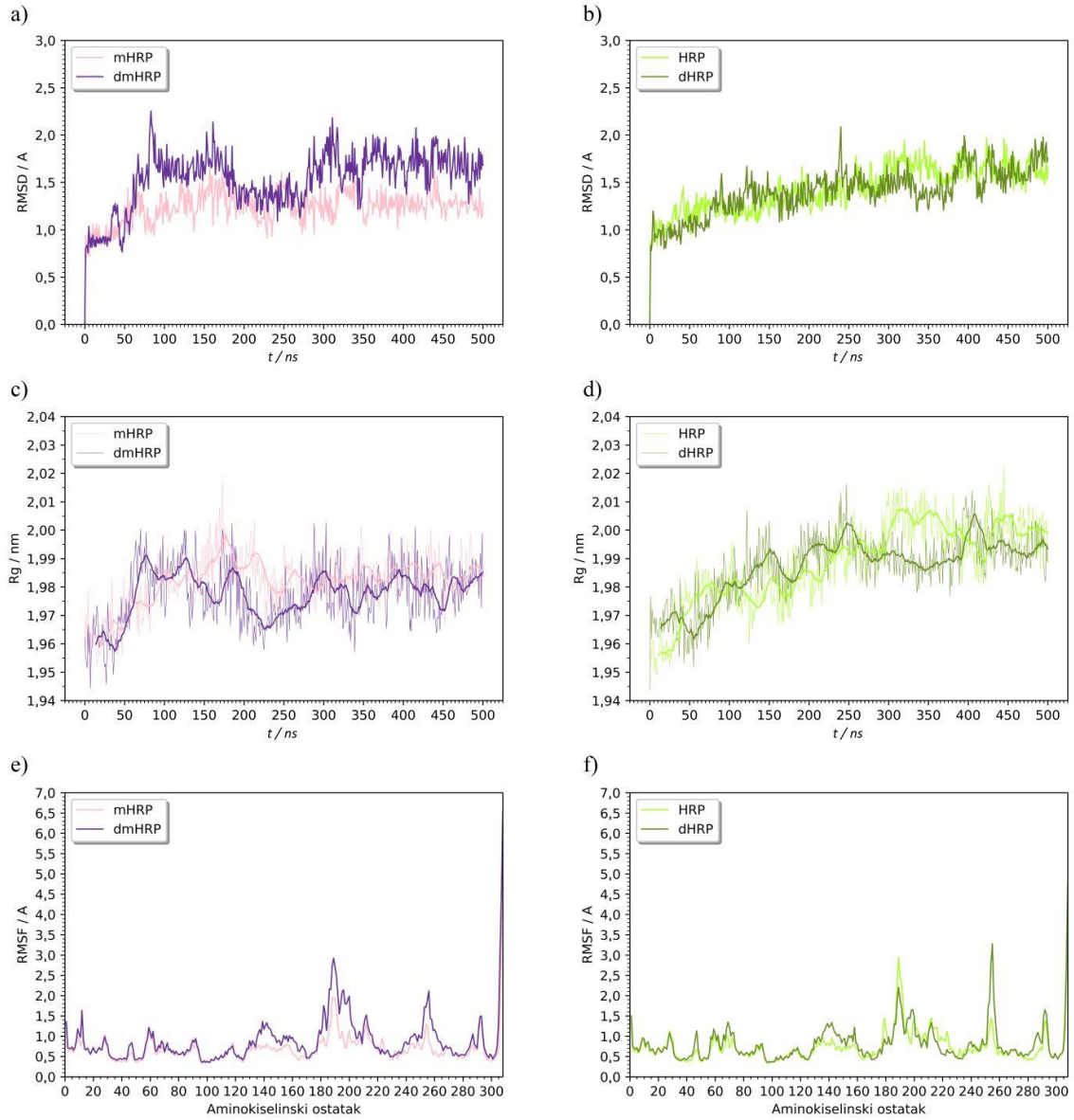


Slika 7. Prikaz enzima HRP s i bez glikozilacije. Prosječni volumen koji zauzimaju $Man_5GlcNAc_2$ glikani tijekom simulacije prikazan je bijelim volumenima, HRP je prikazan sekundarnim strukturama, dok je hem kofaktor prikazan CPK modelima. Ulaz u aktivno mjesto odvija se preko karboksilnih skupina hema označenih na slici crvenom bojom kisikovih atoma.

Rezultati RMSD analiza prikazani su na slici 8 (a i b). Krivulja RMSD dmHRP enzima nestabilnija je u odnosu na krivulju mHRP, dok je promjena RMSD tijekom simulacija HRP i dHRP slična. Rezultat upućuje da glikozilacija stabilizira određene regije mHRP pa kada je ona maknuta dolazi do manjih promjena u konformaciji. RMSD govori o sličnosti enzima s referentnim prvim kadrom, a kao još jedna mjera stabilnosti koristi se graf radijusa giracije

(prikazi c i d na slici 8). Zanimljivo je kako su se mutirani enzimi iskazali većom sveukupnom stabilnosti nego divlji, što je vidljivo iz spomenutih grafova radijusa giracije i nagiba RMSD grafova. Uklanjanje glikozilacije nije imalo značajnog utjecaja na volumen i kompaktnost enzima, što je dobar znak koji svjedoči u prilog stabilnosti neglikoziliranih formi.

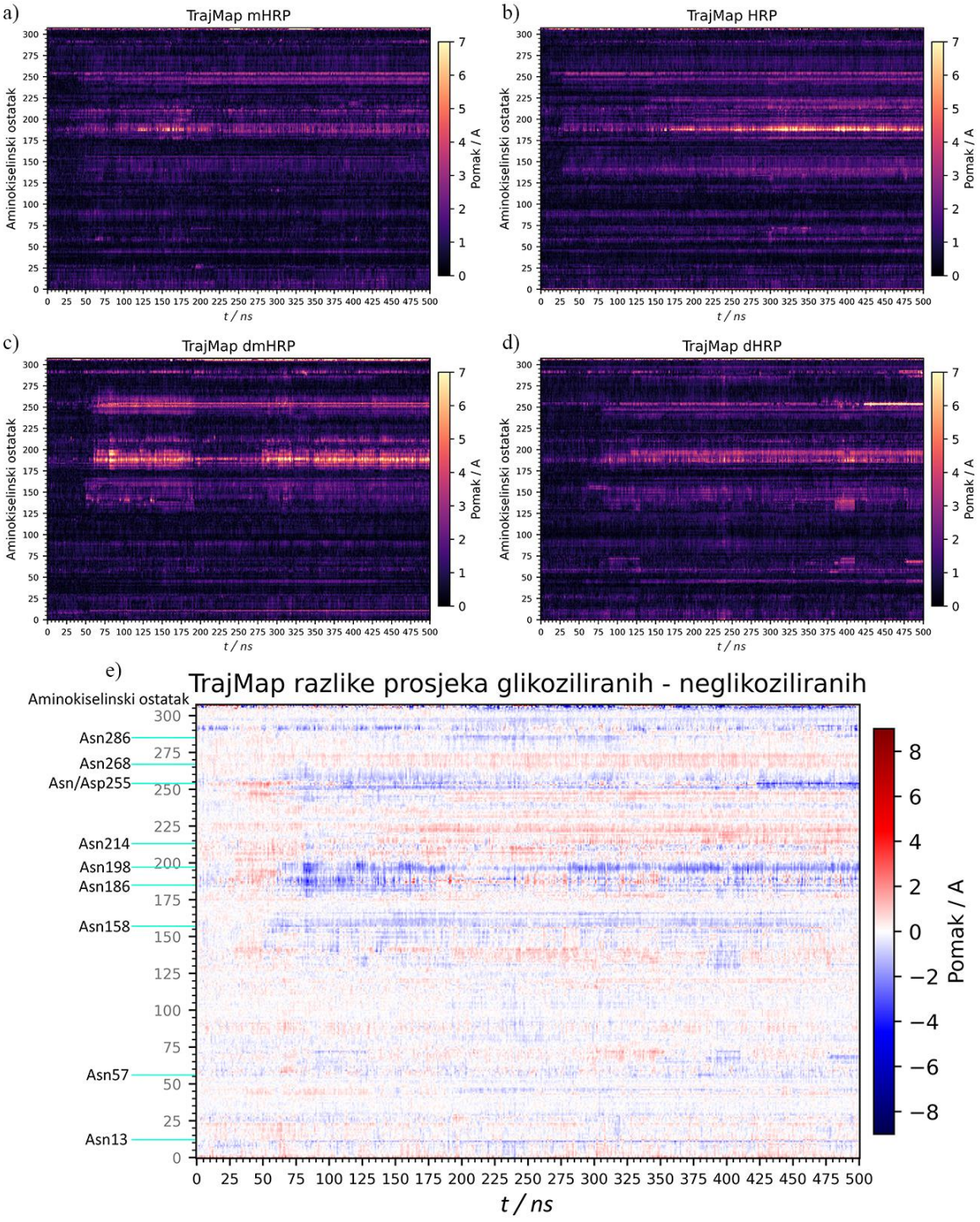
Regije u kojima je došlo do konformacijskih promjena i/ili povećanja fluktuacija vidljive su na RMSF prikazima pod c) i d). Potvrđeno je kako su fluktuacije povećane u neglikoziliranim varijantama što je u skladu s literaturnim podatcima[11] iznesenim u poglavlju 2.1.3. Kod HRP primijećene su regije gdje su fluktuacije pojačane kada je glikozilacija prisutna (aminokiseline od 180. do 220.). Iako fluktuacije nisu jake te se ne nalaze blizu mjesta od interesa za katalitičku aktivnost, zanimljivo je kako je taj fenomen obrnut u slučaju mHRP i dmHRP. Tamo su te regije stabilnije kada ima glikana, a nestabilnije kada nema, što upućuje da su mutacije stabilizirale taj dio enzima. Konkretna mutacija koja je u pitanju jest Asn175Ser koja stabilizira cjelokupnu okolnu regiju. Osim toga, druge regije s povećanim fluktuacijama odgovaraju raznim petljama i drugim predjelima koja nisu od značaja za katalitičku aktivnost. Mutacije i neglikozilacija omogućile su C-terminalnom kraju enzima da slobodnije fluktuira što može pozitivno utjecati na sprezanja s markerima poput heksaHis-oznaka. Na N-terminalnom kraju nije opažen utjecaj mutacija ili neglikolizacije.



Slika 8. Grafovi analiza rekombinantnog enzima označenog ljubičastom bojom i divljeg tipa označenog zelenom bojom. Pod a) i b) su prikazane analize RMSD, pod c) i d) analize radijusa giracije (Rg) sa pokretnim prosjekom (engl. *Rolling average*) od 15 ns) te pod e) i f) RMSF analize.

Na temelju simulacija izrađene su mape trajektorija (slika 9 prikazi od a) do d)). Kako bi se dodatno istaknule razlike između glikoliziranog i ne glikoliziranog HRP, napravljena je razlikovna mapa trajektorija (prikaz e) na slici 9). Razlikovna mapa trajektorija napravljena je uzimanjem prosjeka promjena mHRP i HRP simulacija, te oduzimanjem od toga prosjeka promjena dHRP i dmHRP simulacija istaknute su razlike između simulacija glikoziliranih i neglikoziliranih sustava. Plave regije odgovaraju mjestima gdje je pomak veći kada nema

glikozilacije, bijela mjestima gdje su pomaci slični, te crvena regije gdje su pomaci veći kada ima glikozilacije.



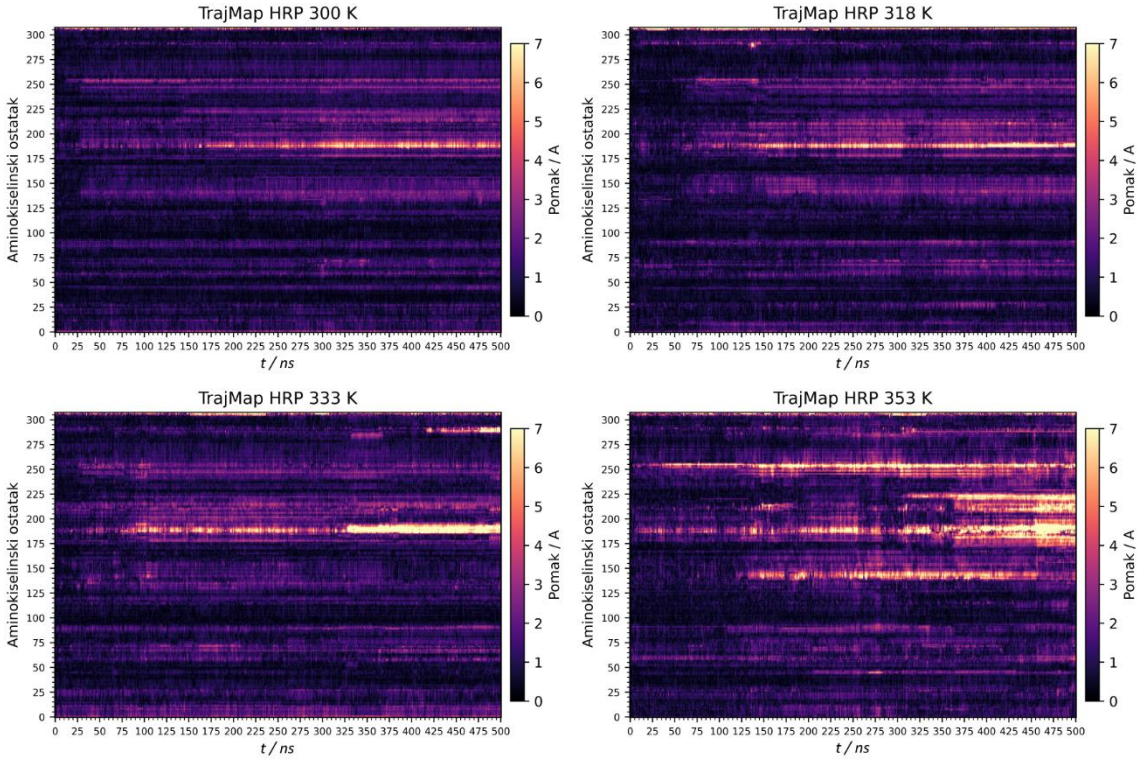
Slika 9. Mape trajektorija (TrajMap) individualnih simulacija prikazane na: a) – d). Na e) je prikazana razlikovna mapa trajektorija dobivena oduzimanjem prosjeka neglikoziliranih varijanti (c i d) od prosjeka glikoziliranih varijanti (a i b).

Na mapi su označena mjesta glikozilacije. Plave pruge kod Asn286, Asn13, Asn186, Asn198, Asn158 te Asn255 označavaju mjesta čije promjene su izraženije u slučajevima bez glikozilacije. Mapa je pretežno bijela s regijama gdje se izmjenjuje blago crveno i blago plavo, što još jednom potvrđuje kako prisutnost ili neprisutnost glikozilacije nije krucijalna za stabilnost enzima. Plave pruge kod Asn286, Asn13, Asn186, te Asn198 pripadaju petljama i drugim regijama koje nisu od značaja za aktivnost enzima. Zanimljive su pruge kod Asn158 te Asn/Asp255 što odgovara mjestu mutacije. Kod mjesta 255 vidljiva je izuzetno izražena promjena nakon 400. nanosekunde, koja je primijećena kod dHRP (prikaz d). O njoj će biti podrobno diskutirano u poglavlju 4.2.3. jer je povezana s konformacijskom promjenom podizanja zavojnice E. Sumarno, potvrđeni su literaturni navodi da glikozilacija stabilizira enzim, no gubitak iste nema negativnih posljedica u vidu gubitka strukture ili katalitičke aktivnosti.

4.2. Utjecaj temperature na stabilnost enzima HRP

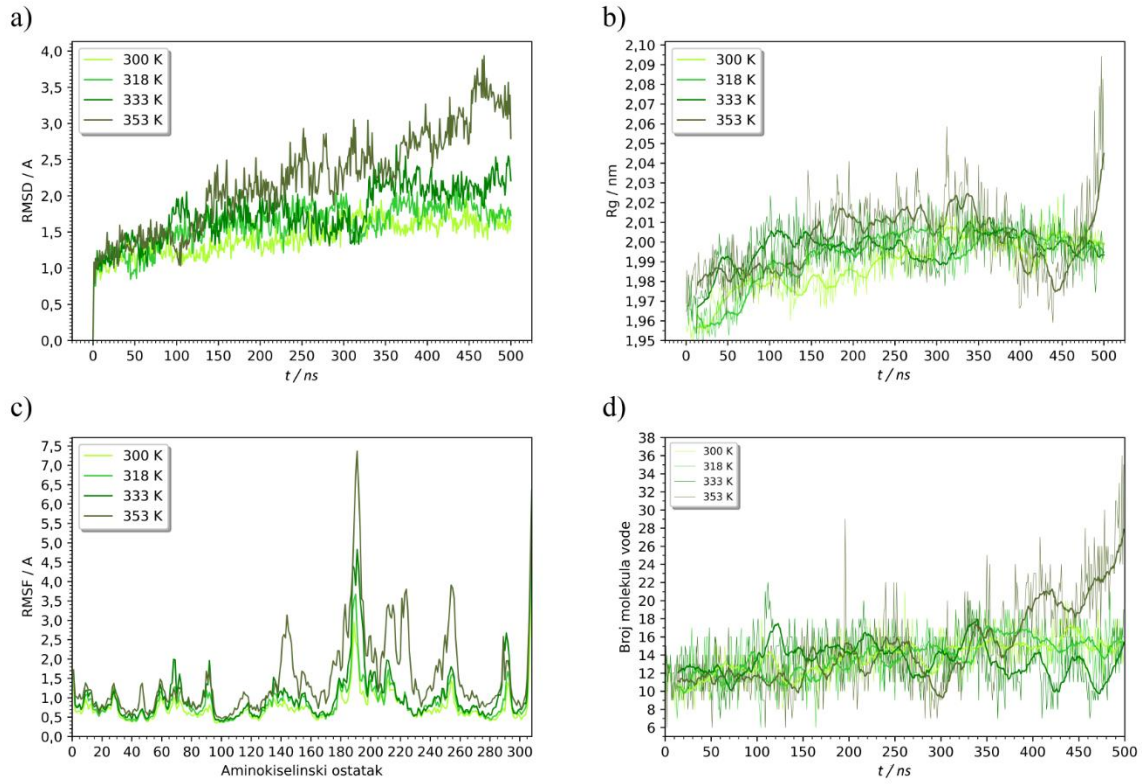
4.2.1. Istraživanje divljeg tipa enzima HRP

Kompletan tijek simulacija enzima HRP pri različitim temperaturama prikazan je mapama trajektorija na slici 10. Vidljivo je kako povećanjem temperature dolazi do novih i jačih promjena u konformaciji koje su predstavljene svijetlim prugama, što je skladno očekivanjima. S obzirom da su simulacije u trajanju od 500 ns, što nije dovoljno dugačko vrijeme za odvijanje cjelokupne denaturacije, očekivano je da bi u realnim sustavima pri višim temperaturama (npr. 333 K i 353 K) došlo do barem djelomične denaturacije i gubitka sekundarne strukture značajno izraženije nego u provedenim simulacijama. Simulacijama je istražen prvi dio prijelaza enzima iz stabilne konformacije (koja je slična onoj iz kristalne strukture), u konformaciju koju se očekuje dobiti pri višim temperaturama. Na temelju tog prijelaza jednog mikroskopskog sustava izvlače se zaključci i poopćuju na očekivanja za realni makroskopski sustav koji je postignuo ravnotežu.



Slika 10. Mape trajektorija (TrajMap) simulacija divljeg tipa glikozilirane varijante enzima HRP pri temperaturama od 26,85 °C (300 K), 44,85 °C (318 K), 59,85 °C (333 K) te 79,85 °C (353 K).

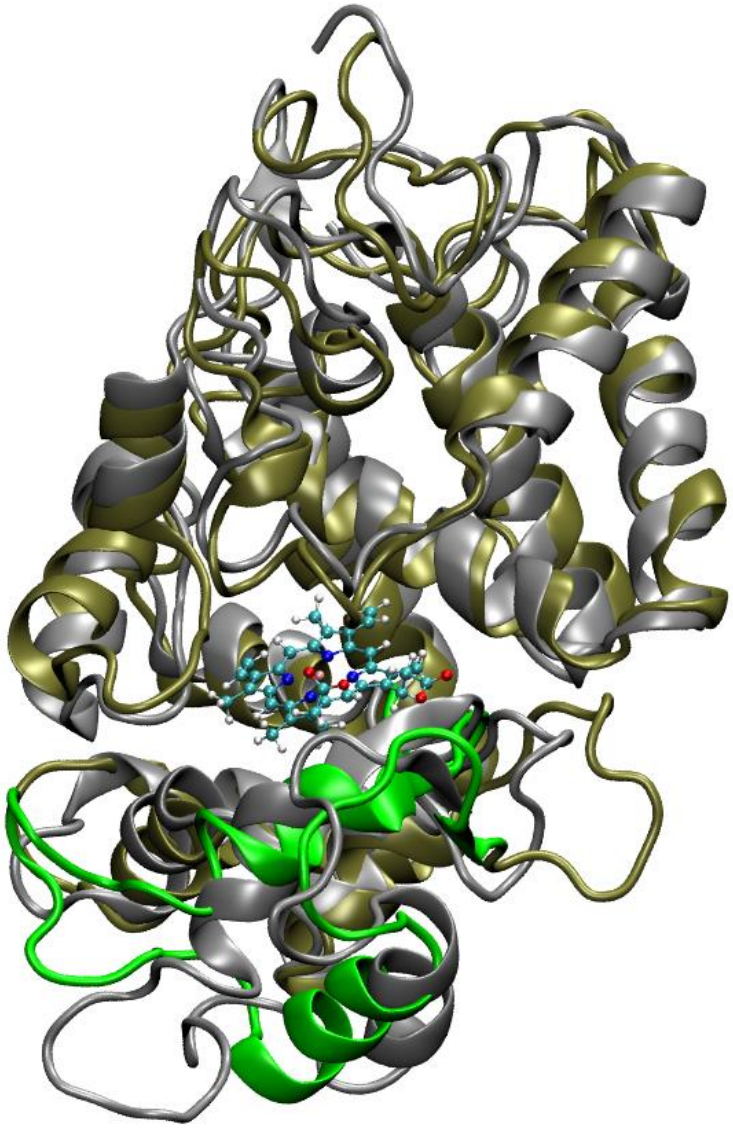
Tijekom simulacija, HRP sustavi pri 300 K, 318K i 333 K postigli su razmjerno stabilnu konformaciju, dok se varijanta pri 353 K nije ustalila nego se krenula raspadati. To je vidljivo iz RMSD grafa prikazanog na slici 11 a, gdje se prve tri temperature drže razmjerno blizu jedna drugoj, a simulacija pri 353 K na trenutke ima i do dvostruko višu RMSD vrijednost. Dodatna potvrda toga je graf radijusa giracije prikazanog prikazom b), gdje je jasno vidljivo kako pri kraju simulacije varijanta na 353 K gubi kompaktnost te samim time i strukturu.



Slika 11. a) RMSD graf HRP-a, b) Graf radijusa giracije, c) RMSF graf, d) graf ovisnosti broja molekula vode u šupljini aktivnog mjesta, 0,5 nm od hema i Arg38 u vremenu. Sve analize su napravljene pri različitim temperaturama. Na krivuljama b) i c) superponiran je pokretni prosjek od 15 ns.

Graf 11d prikazuje broj molekula vode tijekom simulacije koje su na udaljenosti manjoj od 0,5 nm od hema i Arg38, što se uzima kao voda unutar i na ulazu u šupljinu aktivnog mjesta. Vidljivo je kako i taj prikaz prati RMSD, te su temperature od 300 K, 318 K te 333 K stabilne, dok količina vode u aktivnom mjestu pri temperaturi od 353 K naglo skače pri oko 350 ns. To potvrđuje primijećen djelomičan gubitak strukture oko aktivnog mjesta, što omogućuje ulazak dodatnih molekula vode. Sveukupno taj raspad je uzrokovan jačim fluktuacijama zbog više temperature, a lokaliziran je većinom na proksimalnu regiju enzima ispod aktivnog mjesta u odnosu na hem, koja je djelomično odvojena od proteina i ne sadrži dijelove bitne za katalitičku

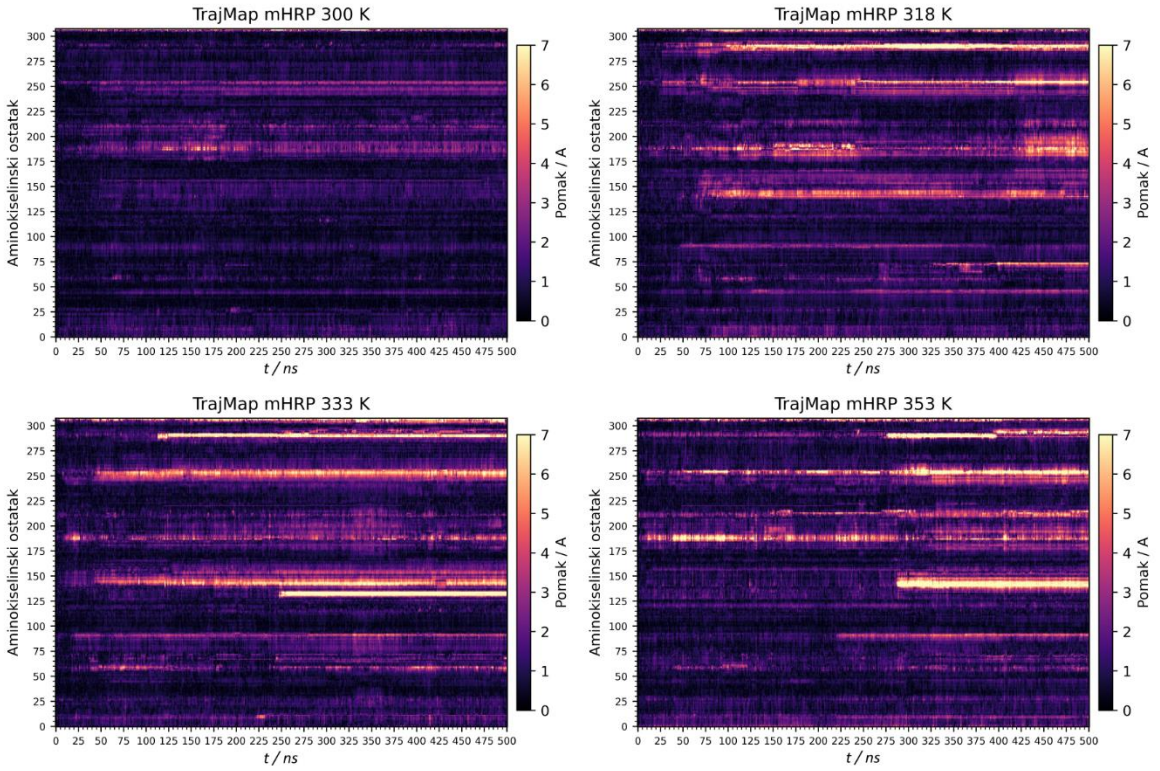
aktivnost (slika 12). Magnituda fluktuacija i pripadajuće regije enzima prikazane su RMSF grafom. To su većinom petlje, npr. regije od 210 to 230 te 240 do 270, što su strukturno fleksibilniji dijelovi enzima. Jedna od značajnijih promjena pripada fluktuaciji pri 145. aminokiselinskom ostatku. To je pomicanje zavojnice E, o kojemu će se podrobno diskutirati u daljnjem poglavlju 4.2.3.



Slika 12. Preklopljene strukture HRP pri 300 K (sivo) i HRP pri 353 K (maslinasto zeleno) na kraju simulacije, s proksimalnom regijom do čijeg parcijalnog raspada dolazi (170. do 220. aminokiseline) označenom intenzivno zelenom bojom.

4.2.2. Istraživanje rekombinantnog tipa enzima HRP (mHRP)

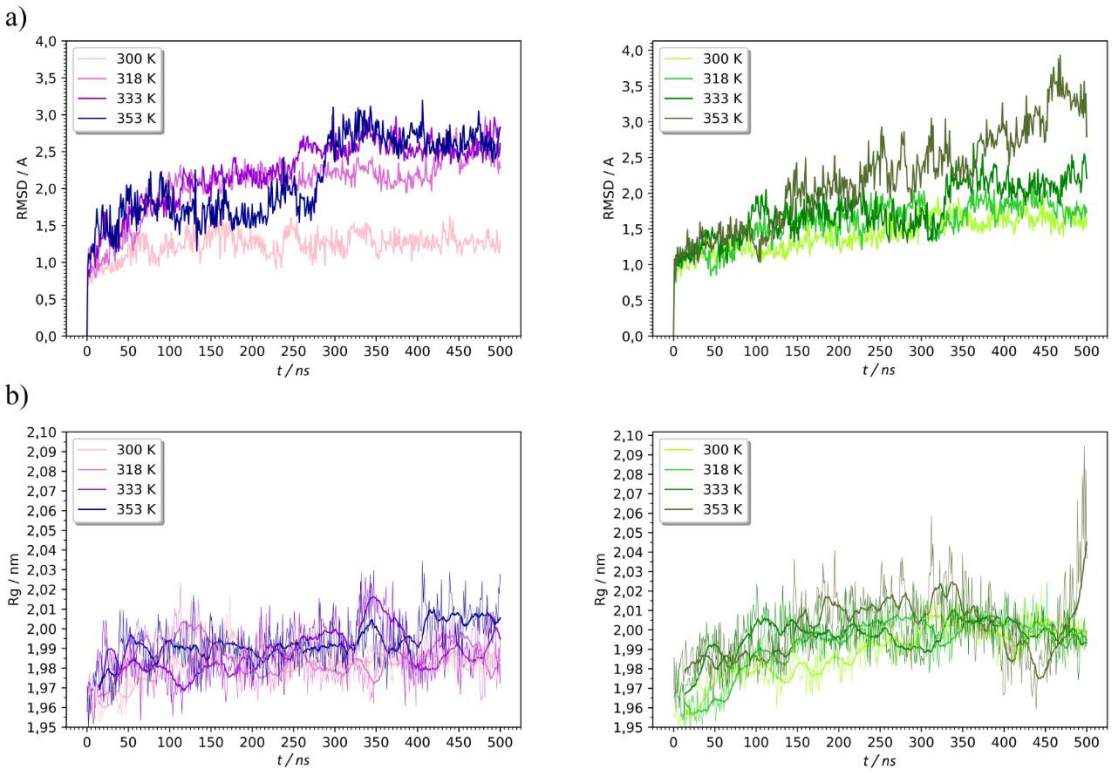
Kompletni tijek simulacija mHRP enzima prikazan je mapama trajektorija na slici 13. Varijanta mHRP od 300 K razmjerno je slična HRP varijanti pri 300 K (slika 11), no u mHRP je došlo do manje konformacijskih promjena nego u HRP. Već iz toga se može zaključuje kako je mHRP stabilniji od HRP pri nižoj temperaturi od 300 K. U kontekstu Martell *et al.* rada[1] u kojemu su pripravljene mutacije, taj zaključak je potpuno razumljiv.



Slika 13. Mape trajektorija simulacija mutirane glikozilirane varijante enzima, mHRP, pri temperaturama od 26,85 °C (300 K), 44,85 °C (318 K), 59,85 °C (333 K) te 79,85 °C (353 K).

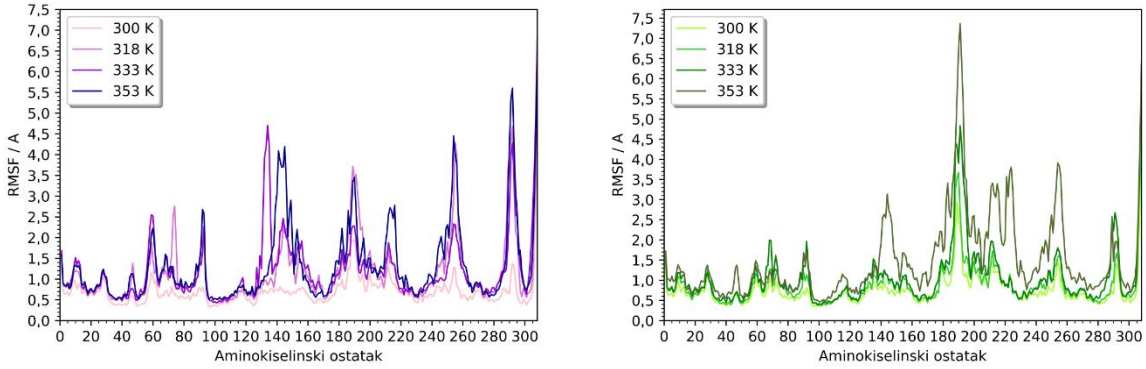
Pri višim temperaturama dolazi do većih konformacijskih promjena što je u skladu sa simulacijama HRP sustava. No neke od opaženih konformacijskih promjena nisu opažene u HRP simulacijama. Iako je došlo do izraženijih konformacijskih promjena, mHRP je sveukupno stabilniji i postojaniji od HRP pri višim temperaturama. Konformacijske promjene samo ukazuju da je stabilna konformacija različita od one srodne kristalnoj strukturi iz koje je simulacija krenula. Kako bi se procijenila stabilnost novih konformacijskih formi potrebno je gledati druge parametre poput RMSD i radijusa giracije (slika 14). mHRP je označen na slici 14 i dalje u svim prikazima ljubičastom bojom, a HRP je označen zelenom bojom. Iz spomenutih RMSD grafova vidljivo je kako u slučaju mHRP pri tri više temperature dolazi do

određenih konformacijskih promjena, a krivulja od 300 K ostaje stabilna bez konformacijskih promjena. U usporedbi s HRP krivulja mHRP pri 300 K stabilnija je od HRP pri svim temperaturama. RMSD mHRP pri 318, 333 i 353 K sličan je krivulji HRP pri 353 K, što je uzrokovano istom promjenom pomicanja zavojnice E, koje će biti detaljno razrađeno u poglavlju 4.2.3. Konkluzivan dokaz veće stabilnosti mHRP od HRP dan je grafovima radijusa giracije koji su niži u slučaju mHRP i viši u slučaju HRP. Opet je vidljiv oštri skok HRP tijekom simulacije na 353 K krivulje koji označava početak raspada, dok takvog skoka nema u mHRP na 353 K. Tu još jednom valja naglasiti kako je za očekivati da pri 353 K dođe do kompletnog raspada enzima, no ovdje je sagledavan vrlo kratak period na početku tog raspada te se izvlače zaključci na temelju postojanosti samo u tom kratkom periodu. Što ranije dođe do raspada, to je energetska barijera za njega manja, te je enzim nestabilniji u odnosu na varijantu gdje je kasnije došlo do raspada. Na temelju ovoga zaključuje se kako je mHRP stabilniji, a s obzirom da su razlike HRP i mHRP u mutacijama, može se zaključiti kako uvedene mutacije termički stabiliziraju mHRP.



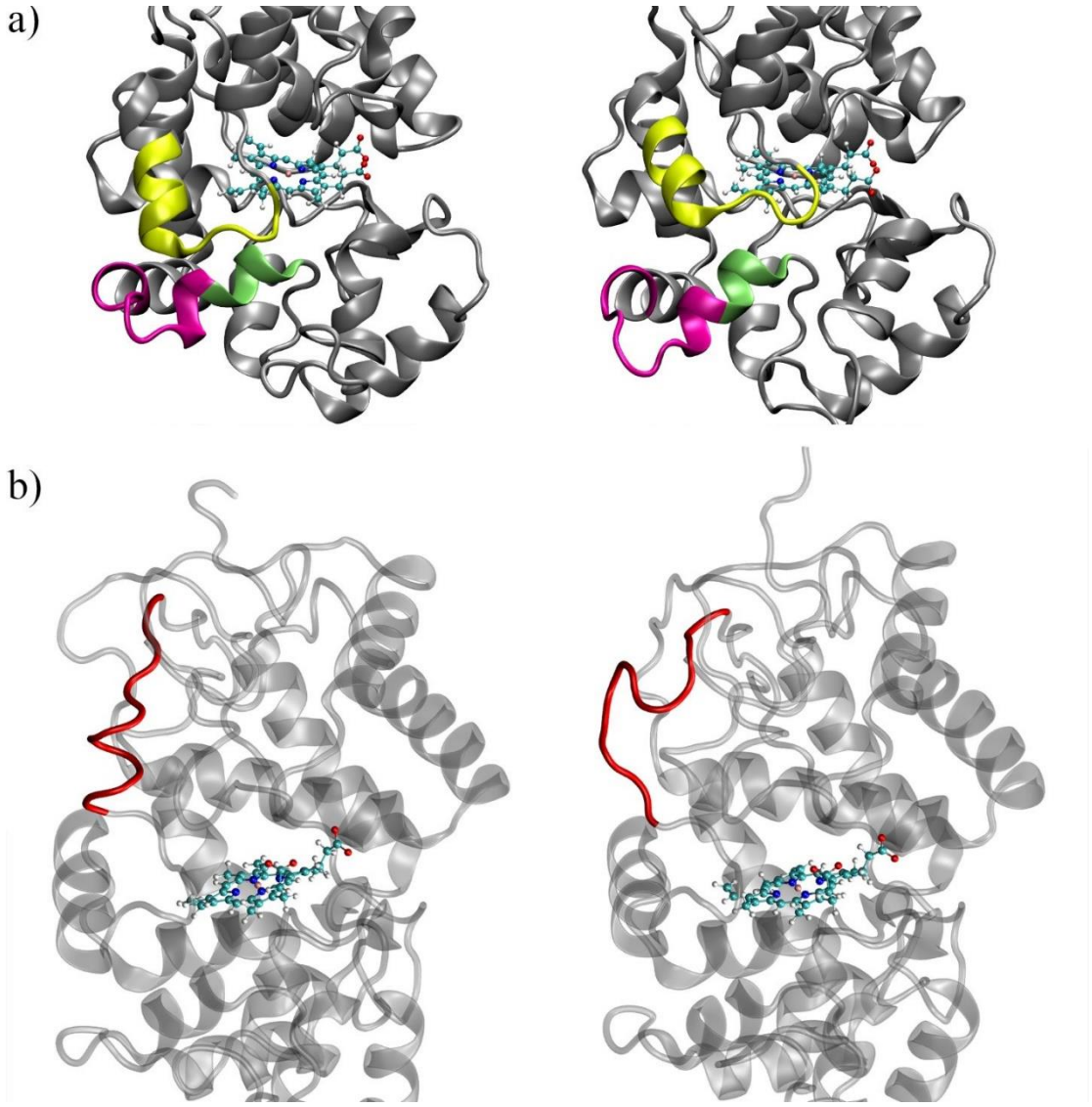
Slika 14. Rezultati simulacija mHRP (ljubičasti) i HRP (zeleni) sustava pri različitim temperaturama. a) RMSD graf, b) graf radijusa giracije s pokretnim prosjekom od 15 ns.

Konkretne regije konformacijskih promjena vidljive su na RMSF grafovima prikazanim na slici 15. Varijanta mHRP ima sveukupno više fluktuacija, no one su lokalizirane većinom na diskretne maksimume koji odgovaraju strukturno fleksibilnim petljama. Za primijetiti je kako u oba slučaja fluktuacije rastu s temperaturom, što je očekivan rezultat. U slučaju mHRP najizraženija promjena je oko 290. aminokiseline, oko 145. aminokiseline, s vrlo izraženim maksimumom pri oko 135. aminokiseline koji se pojavljuje samo pri 333 K. Fluktuacije oko 290. aminokiseline odgovara promjenama u petlji koja se nalazi na drugom kraju enzima od aktivnog mjesta i nije značajna za katalitičku aktivnost i stabilnost enzima. Promjena oko 145. aminokiseline dogodila se u mHRP pri 318 K, 333 K te 353 K, a kod HRP pri 353 K. To je pomicanje zavojnice E, prikazanom žutom bojom na slici 16, koja se nalazi pored ulaza u aktivno mjesto.



Slika 15. RMSF grafovi mHRP (ljubičasto) i HRP (zeleno), pri različitim temperaturama.

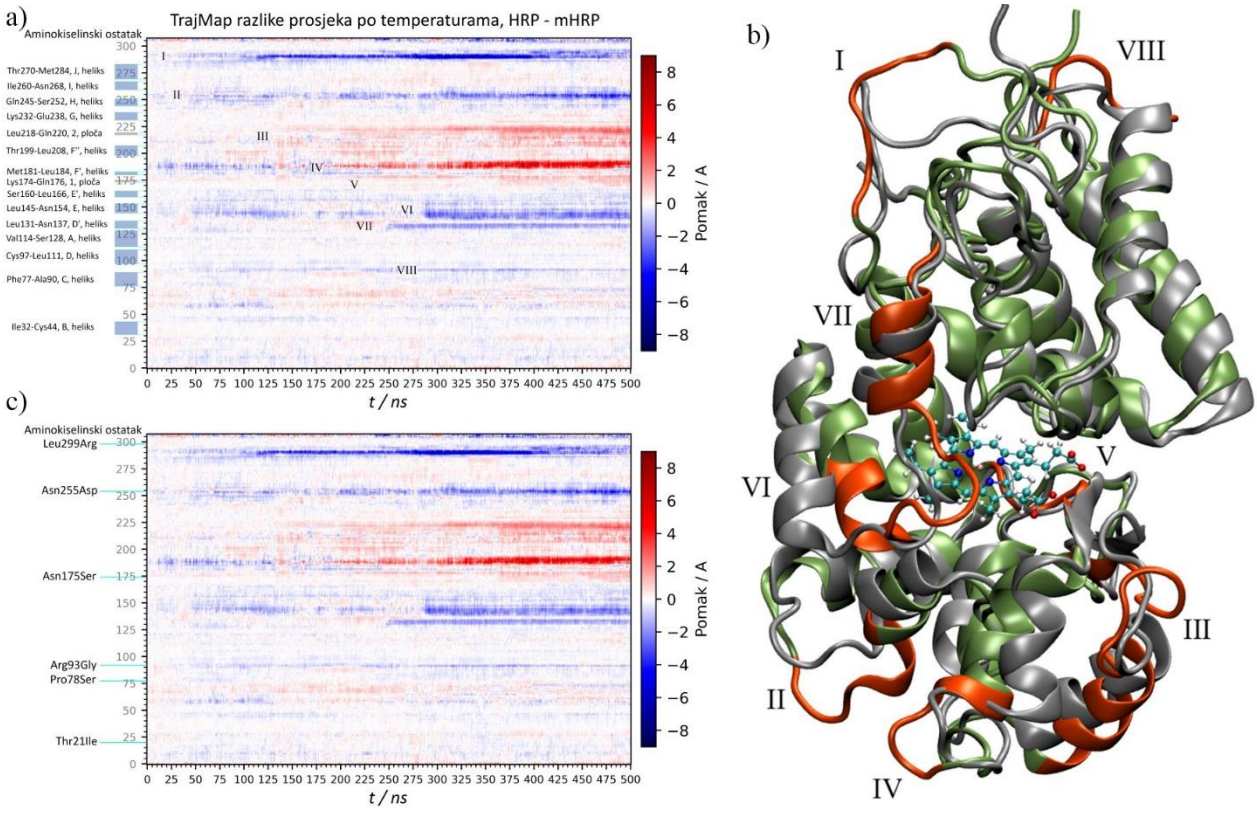
Fluktuacije u mHRP pri 333 K na 135. aminokiselini odgovara djelomičnom odmatanju zavojnice D' koja se nalazi u blizini aktivnog mjesta i nadovezuje se na zavojnicu E preko jedne petlje. Vizualno je prikazana na slici 16, na prikazu b). Očekivano bi bilo da ta promjena uzrokuje smanjenje stabilnosti te da pored toga nema značajnijeg utjecaja. Ona je opažena samo pri 333 K, ne pri 353 K, pa vjerojatno predstavlja lokalni konformacijski minimum. Pri 333 K očekivan je gubitak barem parcijalno sekundarne strukture, tako da iako dobiveni rezultat nije konzistentan i reproduciran kroz druge simulacije, kontekstualno odgovara temperaturi pri kojoj je opažen. Pored spomenutih, ostale promjene odgovaraju petljama i drugim mjestima koja nisu od izravnog interesa za strukturnu stabilnost i značaja za funkcionalnost enzima.



Slika 16. Strukture enzima HRP dobivene tijekom MD simulacija: a) zavojnica E i pripadna petlja označene žuto, aminokiselinskih ostataka 140 do 150, prije (lijevo) i poslije (desno) konformacijske promjene, b) zavojnica D' prije i poslije konformacijske promjene opažene u mHRP pri 333 K.

Da bi se prikazale sve razlike promjena u lokaciji u proteinu i u vremenu tijekom simulacije između mHRP i HRP, napravljena je razlika mapa trajektorija. Uprosječene su mape trajektorija četiri simulacije pri različitim temperaturama za mHRP i za HRP, te je od temperaturnog prosjeka HRP oduzet temperaturni prosjek mHRP. Rezultantna mapa trajektorija prikazana je na slici 17, gdje su na gornjem prikazu na *y* osi označene sve sekundarne strukture, te su rimskim brojevima od I do VIII numerirane pruge koje odgovaraju regijama sa značajnim pomacima. Pozitivne, crvene, regije odgovaraju dijelovima gdje su pomaci bili veći u slučaju HRP, a negativne, plave, regije odgovaraju dijelovima gdje su

pomaci bili veći u slučaju mHRP. Na donjem prikazu označene su lokacije mutacija. Pruga I odgovara fluktuacijama petlje koja se nalazi distalnom djelu enzima kod aminokiselina 290 do 299, daleko od aktivnog mjesta i blizu C-terminalnog kraja. Na djelu od 290 do 296 spomenuta petlja je malo izbočena van enzima i slobodno fluktuira, a oko 299. ona ostvaruje interakcije s petljom oko aminokiselinskog ostatka 93 označenog na slici brojem VIII. Petlja VIII je dio strukturnog motiva zavojnica-petlja-zavojnica (engl. *Helix-Loop-Helix, HLH*). Regijama I i VIII pripadaju mutacije Leu299Arg i Arg93Gly respektivno, što je vidljivo na donjem prikazu, te arginin pri kraju simulacije bočnim ogrankom ostvaruje vodikovu vezu s karbonilnim kisikom okosnice glicina. Tome odgovara izražena plava pruga VIII. U tim petljama došlo je do više promjena u mHRP, a te promjene odgovaraju prijelazu iz kristalne strukture, koja odgovara HRP, u nativnu strukturu za mHRP gdje je ostvarena navedena vodikova veza koja može stabilizirati petlje (budući da je mHRP simulacija temeljena na kristalnoj strukturi HRP, te obje simulacije kreću iz srodne konformacije). Izuzev toga, te petlje nisu od daljnjeg značaja.



Slika 17. Razlikovna mapa trajektorija temperaturnog prosjeka mHRP i HRP (a i c). Negativne plave regije odgovaraju mjestima gdje je pomak bio veći u mHRP, a crvene regije odgovaraju regijama gdje je pomak bio veći u HRP. Bijele regije koje su nula odgovaraju mjestima gdje su pomaci bili isti. Na prikazu a) označene su sekundarne strukture s pripadnim literaturnim[4] jednoznakovnim nazivima i pripadne značajne pruge notirane rimskim brojevima. Na prikazu

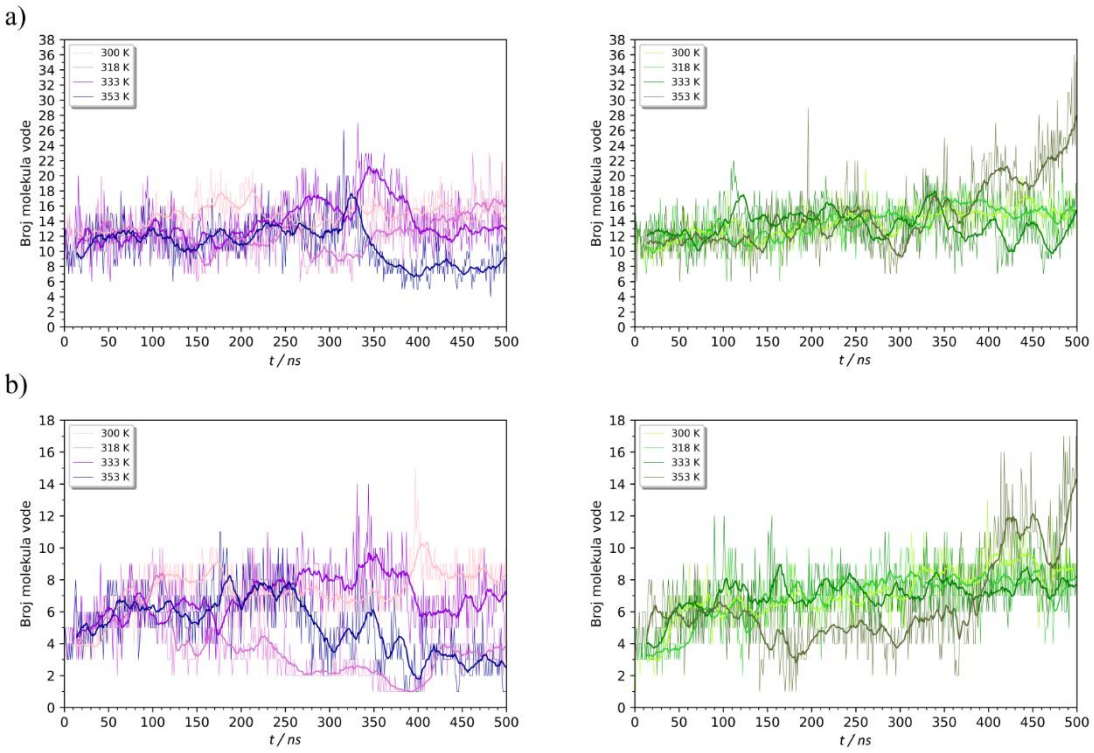
c) označena su mjesta mutacija. Prikazom b) sivom bojom je prikazan HRP pri 300 K, a zelenom HRP pri 353 K. Narančaste regije označene rimskim brojevima odgovaraju značajnim prugama istih oznaka iz prikaza a).

Regiji III odgovara dugačka petlja usred koje je mala ploča (engl. *Beta plane*) literaturno nazvana „ploča 2“ (slika 17). Ploča 2 ostvaruje interakcije s pločom 1 koja se nalazi kod aminokiselinskog ostatka 175. Na mjestu 175 nalazi se mutacija Asn175Ser, te je ta pruga označena oznakom V. Te ploče se nalaze na samom ulazu u aktivno mjesto, te u mHRP mutirani serin može ostvariti vodikovu vezu s karboksilnom skupinom hema bolje nego asparagin. Ta vodikova veza stabilizira ploču 1, što onda stabilizira ploču 2 i pripadnu petlju označenu brojem III. Poboljšana sposobnost ostvarivanja vodikove veze ukazuje na zaključak da mutacija stabilizira tu regije te je mutant stabilniji u tom predjelu. Shodno tome, tu dolazi do više promjena u slučaju ne-mutiranog, odnosno divljeg tipa, enzima HRP. Histidin 170 koordinira s ionom željeza iz hema pa je pogodna što veća stabilnost u toj regiji. Također je vidljivo iz dijela u blizini pruge V kako su pomaci oko aminokiseline 170 izraženiji kod HRP. Mjerenje količine molekula vode u blizini aktivnog mjesta pokazuju dostupnost aktivnog mjesta i prikazani su na slici 18. Ti grafovi potvrđuju kako je šupljina aktivnog mjesta postojanija u slučaju mHRP. U slučaju HRP količina vode u oba slučaja raste, što ukazuje na barem djelomičan pad postojanosti konstrukcije aktivnog mjesta. Rastom temperature izraženije su fluktuacije enzima te molekule vode imaju veću kinetičku energiju i očekivan bi bio pad količine molekula vode s temperaturom, što i je aproksimativno uočeno u mHRP. Grafovi količine molekula vode potvrđuju opažanje da je šupljina aktivnog mjesta stabilnija u mHRP više nego u HRP. Pad količine molekula vode oko His170 u slučaju mHRP uzrokovan je promjenom koja će se objasniti pri kraju ovog rada u poglavlju 4.2.3.

Pruga označena s IV odgovara petlji na samom dnu enzima i ona je nestabilna u obje varijante mHRP i HRP. Značajnija je promjena u HRP, no sama ta regija nije od značaja za katalitičku aktivnost. Regija označena s VII je zavojnica s vanjske strane enzima blizu aktivnog mjesta, koja je prikazana prethodno u tekstu na slici 33b. Kao što je već rečeno, to je promjena koja se opazila iznimno u mHRP pri 333 K te nije od značaja za samu katalitičku aktivnost.

Konačno, ostaju pruge II i VI. Pruga II odgovara petlji 255 u kojoj se nalazi mutacija Asn255Asp, te kao rezultat te mutacije na tom mjestu nedostaje jedan glikan. Drugim riječima, mHRP je neglikoziliran na mjestu 255 u odnosu na HRP. To je napomenuto prije u tekstu, no vrijedi ponoviti. Sama petlja 255 označena je purpurnocrvenom bojom na slici 16b. Strukturno

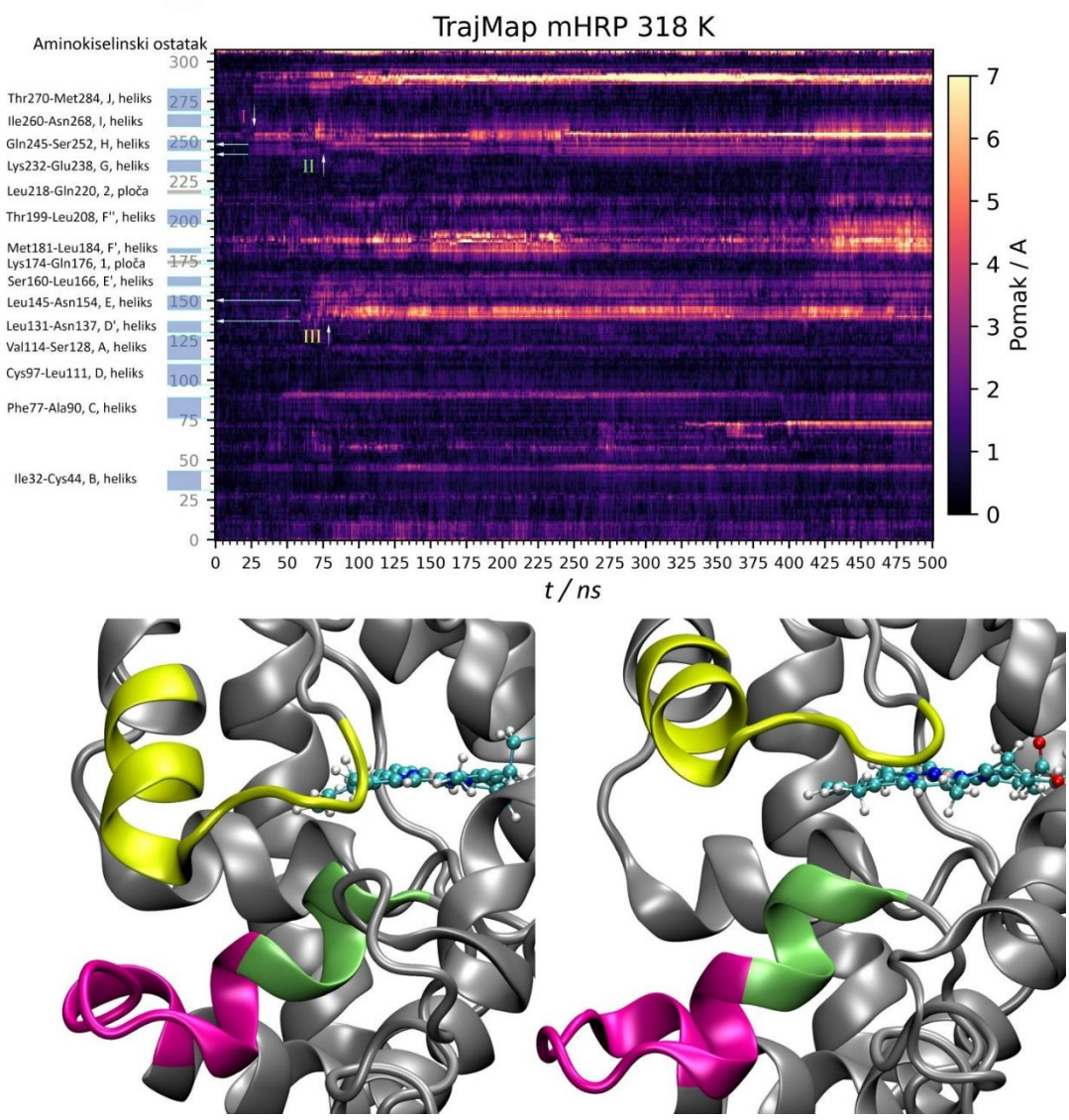
odmah pored petlje 255 smještena je zavojnica E, označena žutom bojom, čija pruga je notirana oznakom VI. Već je spomenuto u paragrafu usporedbe RMSF dijagrama kako se zavojnica E pomiče u mHRP pri 318, 333 te 353 K i u HRP pri 353 K. Petlja 255 ključna je za mehanizam tog dizanja, što će se objasniti u slijedećem podpoglavlju.



Slika 18. Broj molekula vode tijekom MD simulacija pri različitim temperaturama mHRP (ljubičasto) u HRP (zeleno) na udaljenosti manjoj ili jednakoj 0,5 nm od: a) hema i Arg38 što odgovara molekulama vode na ulazu u šupljinu aktivnog mjesta i unutar aktivnog mjesta b) od hema i His170 što odgovara molekulama vode s proksimalne strane aktivnog mjesta.

4.2.3. Mehanizam i posljedice pomicanja zavojnice E

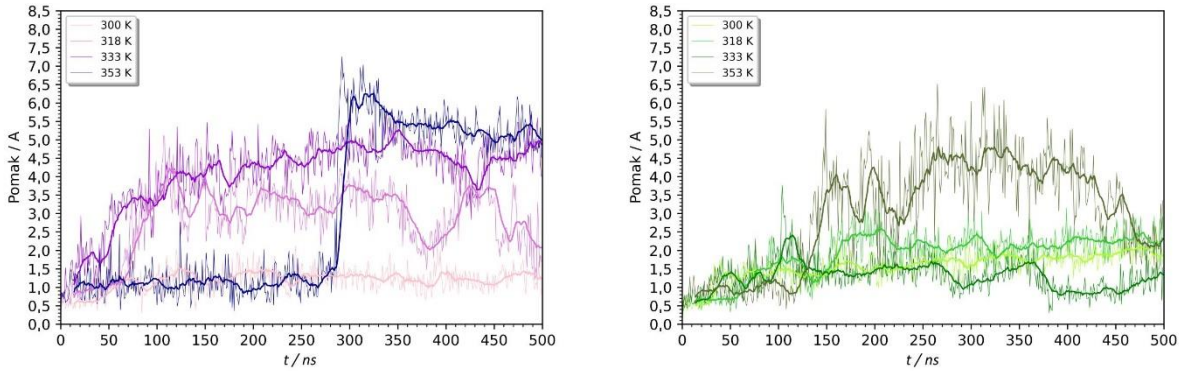
Već je ustanovljeno kako je pomicanje zavojnice E najveća promjena uočena u strukturi enzima. Ta promjena je primijećena u mHRP pri 318 K, 333 K te 353 K i HRP pri 353 K. Kako se ta zavojnica nalazi sa strane otvora aktivnog mjesta, njeno pomicanje dodatno otvara naizgled veliki otvor pored same šupljine aktivnog mjesta, koji je prikazan slikom 19. Usprkos toga, nije opažen povećan ulazak vode u samo aktivno mjesto što se pripisuje hidrofobnom okružju te regije. To je potvrđeno prethodno prikazanom slikom 18. Iako voda ne uspijeva ući u samo aktivnog mjesto, opaženo je povećanje količine vode s bočne strane hema gdje je ta novootvorena šupljina.



Slika 19. Pomak zavojnice E i mapa trajektorija (TrajMap) sa označenim prugama koje sudjeluju u pomaku zavojnice E, prikazane žutom bojom na donjem prikazu. Oznake pruga bojama odgovaraju regijama na donjem prikazu.

Na slici 19 su na mapi trajektorija označene pruge koje sudjeluju u tom pomaku te je vidljiv njihov redoslijed u mehanizmu pomaka: prvo dolazi do pomaka u regiji I označenoj purpurnocrvenom bojom, zatim dolazi do pomaka u regiji II označenoj zelenom bojom i na kraju dolazi do pomaka u regiji III koja odgovara žutoj zavojnici E. Taj mehanizam je razmjerno konzistentan kroz sve simulacije gdje je opažena promjena, no simulacije se međusobno razlikuju u vremenskom periodu između pomaka regija I, II i III. To je potvrđeno iz mapa trajektorija prikazanih slikama 10 i 13. U svim slučajevima, preduvjet za pomicanje zavojnice E je pomak u regiji I (zavojnica 255).

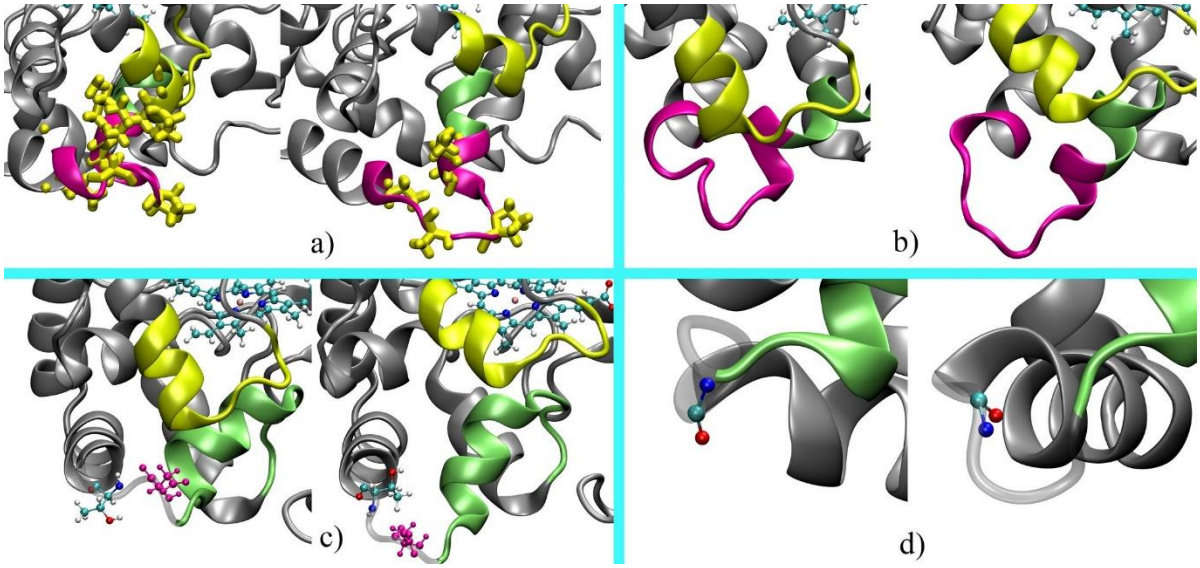
Sama magnituda dizanja zavojnice nije konstanta. Prosječan pomak regije od 140. do 150. aminokiseline, što se interpretira kao zavojnica E i dio pripadne petlje koja se pomiče, prikazan je slikom 20. Jasno je vidljivo kako u tri simulacije kod mHRP i jednoj simulaciji kod HRP dolazi do velikog pomaka, dok u jednoj simulaciji u mHRP i tri simulacije u HRP ne dolazi do značajnog pomaka. Također je jasno vidljivo kako u mHRP magnituda pomaka ovisi o temperaturi, što je u skladu s očekivanjima. Pri svakoj temperaturi simulacije postignuta je određena zasebna konformacija koje je slična, no ne ista konformacijama pri drugim temperaturama simulacija. U slučaju HRP pri 353 K i mHRP pri 318 K također je vidljiva reverzija pomaka, gdje se zavojnica parcijalno krenula vraćati na svoje izvorno mjesto.



Slika 20. Prosječan pomak aminokiselinskih ostataka od 140. do 150. u mHRP (ljubičasto) i HRP (zeleno), što odgovara zavojnici E i pripadnoj petlji, pri različitim temperaturama. Na grafove je superponiran pokretni prosjek od 15 ns.

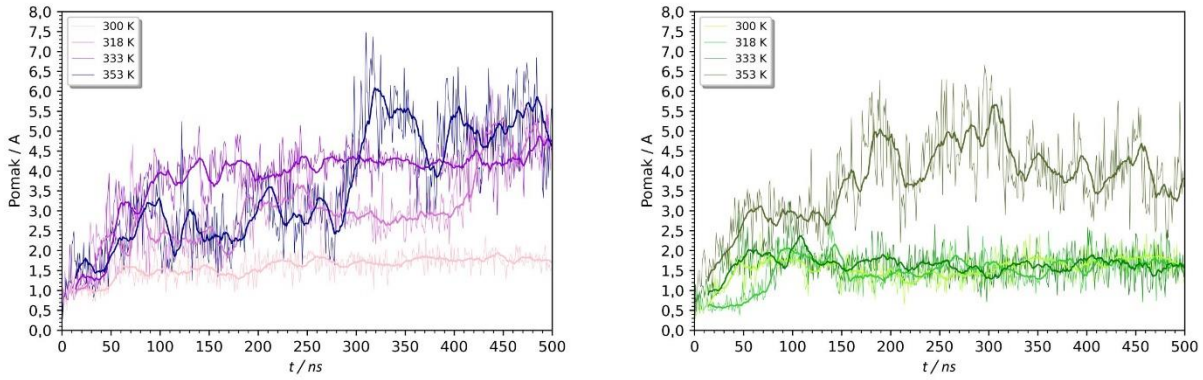
Pomicanje zavojnice E uzrokuje raspad hidrofobnog džepa koje tvore aminokiseline Ala256, Leu145 te Pro146 (slika 21). Regija prije i poslije pomaka s hidrofobnim aminokiselinama označenim žutom bojom vidljiva je na slici 21a. Prikazom 21b prikazana je konformacijska promjena petlje 255 koja uzrokuje cijeli mehanizam. Tu konformacijsku

promjenu uzrokuje rotacija treonina 257, vidljivog na prikazu 21c. Thr257 izvorno bočnim ogrankom strši van proteina, a rotacijom torzijskog kuta prikazanom slikom 21d on se okrene i strši prema proteinu. Promjena torzijskog kuta je rotacija oko σ veze Cα ugljikovog i karbonilnog ugljikovog atoma. Na slici 21c purpurnocrvenom bojom označeno je mjesto mutacije Asn255Asp, gdje kao posljedica mutacije nedostaje glikan $Man_5GlcNAc_2$.



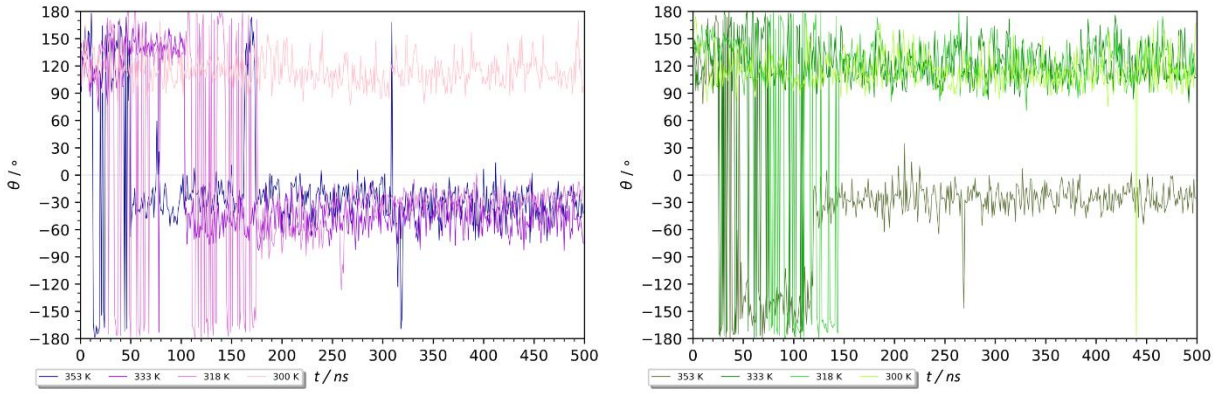
Slika 21. Mehanizam pomicanja petlje E: a) raspad hidrofobnog džepa tvorenog od aminokiselina označenih žutom bojom. b) konformacijske promjene u petlji 255. c) promjena rotacije Thr257, koja uzrokuje konformacijsku promjenu pod b). Purpurnocrvenom bojom označena je mutacija Asn255Asp. d) promjena torzijskog kuta koji svojom rotacijom uzrokuje cijeli mehanizam, gledana kroz σ vezu Cα ugljikovog i karbonilnog ugljikovog atoma Thr257.

Kako na mjestu 255 zbog mutacije nedostaje glikan, petlja 255 je destabilizirana u slučaju mHRP varijante. Time je energetska barijera te konformacijske promjene smanjena pa dizanjem temperature mHRP premošćuje navedenu barijeru prije nego HRP. Slika 22 pokazuje prosječan pomak regije 250. do 260. aminokiseline, što odgovara petlji 255 i neposrednim okolnim regijama te podržava izneseni zaključak. Dio 250 do 260 na prijašnjim slikama označen je purpurnocrvenom bojom. Pomak te regije raste s temperaturom, te izgled cijelog grafa podsjeća na graf pomaka zavojnice E iz slike 20. Za istaknuti je kako na grafu pri kraju mHRP simulacija pri tri simulirane temperature gdje je opažena promjena te simulacije postižu sličnu konformaciju, što je vidljivo iz relativne konvergencije pomaka. Isto tako, sve simulacije gdje nije došlo do dizanja zavojnice E imaju krivulje na istim lokacijama međusobno s HRP i mHRP.



Slika 22. Prosječan pomak regije 250. do 260. aminokiseline tijekom MD simulacije kod mHRP (ljubičasto) i HRP (zeleno), što se uzima kao petlja 255 i neposredno okolna regija.

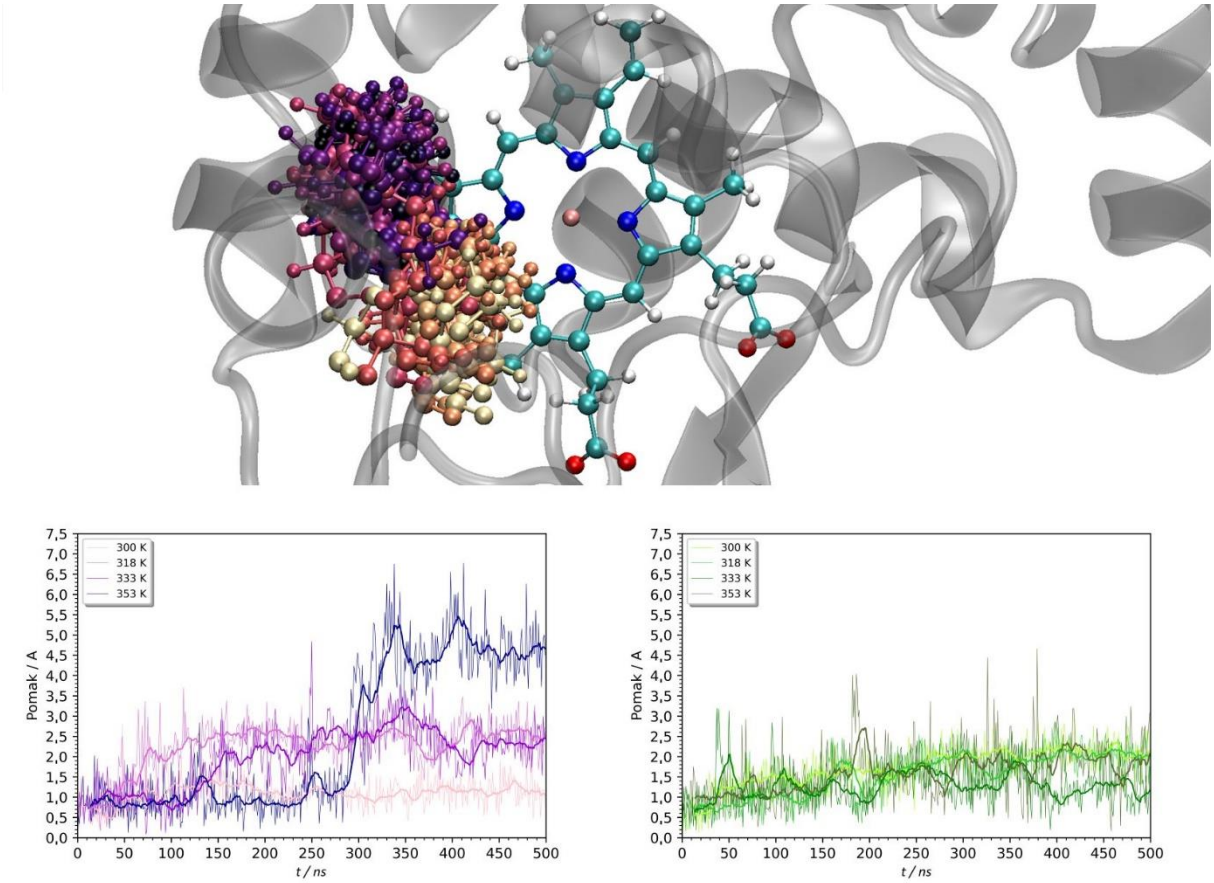
U tako destabiliziranoj zavojnici dolazi do rotacije torzijskog kuta oko kovalentne σ veze Cα ugljikovog – karbonilnog ugljikovog atoma okosnice treonina 257. Upravo je taj torzijski kut ključan u cijelom mehanizmu, te određuje bude li došlo do pomicanja zavojnice. Iznosi navedenog torzijskog kuta tijekom MD simulacija prikazani su na slici 23. Kod mHRP i HRP pri svim temperaturama navedeni torzijski kut počinje pri otprilike 140°. Kod mHRP pri tri temperature pri kojima dolazi do pomicanja zavojnice E on se okrene do -50°. Pri 300 K mHRP ne dolazi do rotacije torzijskog kuta, pa ne dolazi ni do pomicanja zavojnice E. Isto vrijedi i za HRP gdje samo u jednom slučaju dolazi do rotacije, te također samo tada dolazi do pomicanja zavojnice E. Opisani graf pokazuje kako torzijski kut okosnice Thr257 ima ključnu ulogu u prethodno opisanom mehanizmu. Skupno slike 22 i 23 prikazuju kako je petlja 255 nestabilnija kada su prisutne mutacije, što je okidač mehanizma koji dovodi do pomicanja zavojnice E.



Slika 23. Torzijski kut okosnice Thr257 oko σ veze Cα ugljikovog i karbonilnog ugljikovog atoma tijekom MD simulacija, kod mHRP (ljubičasto) i HRP (zeleno). Horizontalne linije označavaju fluktuaciju oko kuta od 180°, pa ovisno o rotaciji poprimaju pozitivnu ili negativnu vrijednost blisku 180°.

Opisana promjena karakterizirana je izmjenom u konstrukciji okoliša aktivnog mjesta, bez ikakvog gubitka sekundarnih struktura. Takav opis i činjenica da je ona opažena pri višim temperaturama (318 K, 333 K i 353 K kod mHRP te 353 K kod HRP), te da magnituda promjene raste s temperaturom (slika 20), ukazuje kako ona ima potencijala biti objašnjenje eksperimentalnog opažanja iz literature [5,6] koje navodi da se termički raspad hrenove peroksidaze događa preko prijelaznog stanja. Kako je navedeno u literaturnom pregledu, to prijelazno stanje karakterizirano je promjenom u tercijarnoj strukturi oko aktivnog mjesta bez ikakvih promjena u sekundarnoj strukturi, te se događa u temperaturnom rasponu od 40 °C do 50 °C što upravo odgovara okarakteriziranom pomicanju zavojnice E. Prema tome, opisani mehanizam pomicanja zavojnice E može se predložiti kao objašnjenje navedenog eksperimentalnog opažanja o intermedijarnom *„pre-molten globule“* stanju termičkog raspada enzima HRP opisanog u literaturi[5,6].

Pomicanje zavojnice E ima dodatni efekt na aktivno mjesto, a to je pomicanje Pro139 koji je značajan za katalitičku reakciju enzima HRP. Literaturno je navedeno da se Pro139 nalazi iznad aktivnog mjesta te ostvaruje vodikovu vezu s vodom i/ili vodikovim peroksidom. Kako se zavojnica E pomiče, tako se i on translatira kao što je prikazano na slici 24. Pozicija Pro139 u vremenu tijekom simulacije mHRP prikazana je skalom u boji koja ide od tamne prema svijetloj. Vidljivo je kako je pozicija Pro139 na početku simulacije drugačija od njegove konačne pozicije nakon što dođe do pomicanja zavojnice E. U slučaju pomaka zavojnice E kod HRP ne dolazi do njegovog pomaka što se može pripisati različitim magnitudama i načinima pomaka. Graf pomaka te regije također je prikazan na slici 24, gdje je uzet prosjek pomaka aminokiselina 138 i 139, kako bi se uklonile eventualne vibracije. Opisana promjena spregnuta s drugim faktorima može doprinijeti smanjenju aktivnost.

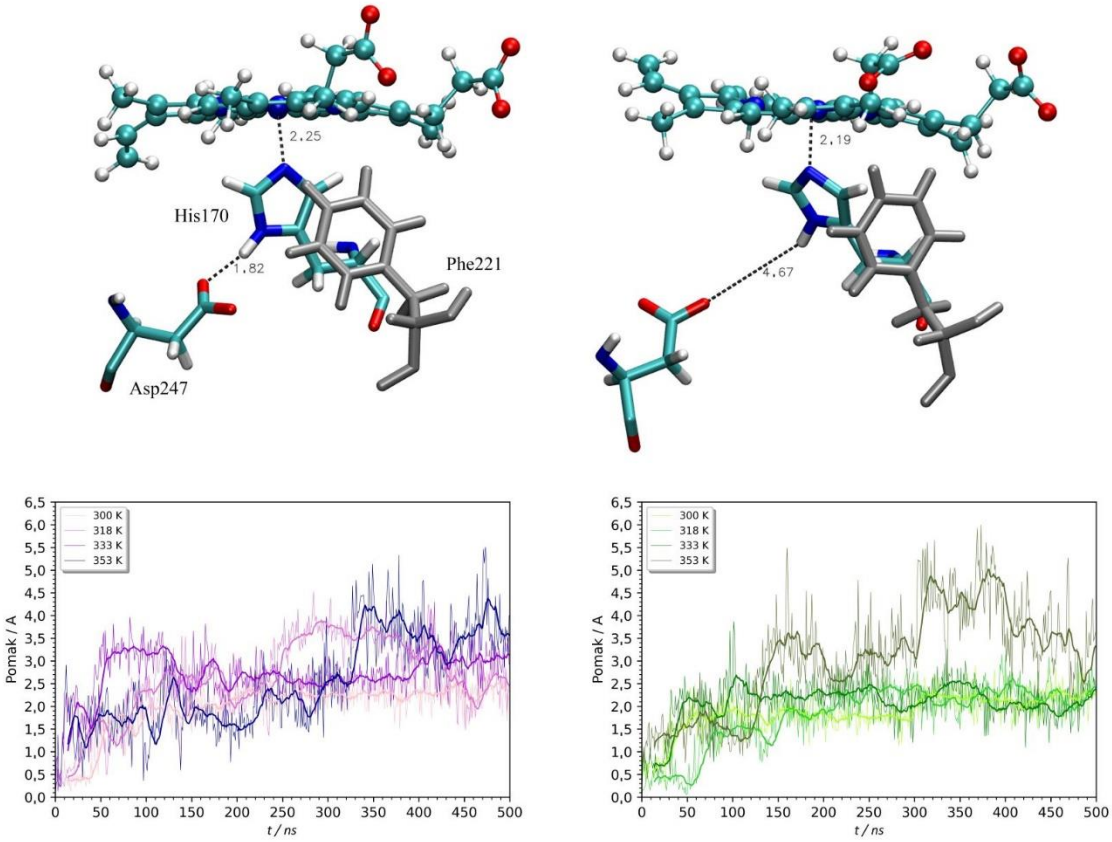


Slika 24. Lokacije Pro139 tijekom simulacija prikazane na skali u boji od tamne prema svijetlom. Najtamnija boja označava početak simulacije, a najsvjetlija kraj simulacije. Ispod je prikazan pomak 138. i 139. aminokiseline tijekom simulacija mHRP (ljubičasto) i HRP (zeleno).

Druga opažena promjena je pucanje vodikove veze između Asp247 i His170. To omogućava histidinu da se rotira, čime dolaze do narušavanja π-π interakcija s Phe221. Promjena je prikazana na slici 25, gdje je još priložen graf pomaka regije neposredno oko Asp247. Asp247 nalazi se na zavojnici H, koja je na prethodno opisanoj slici 16 označena brojem II i zelenom bojom. Iz tog razloga vjerovalo bi se kako je promjena rezultat pomicanja zavojnice E, no u mHRP ona je opažena pri svim temperaturama, uključujući 300 K gdje nije došlo do spomenutog pomaka. U HRP ona je najizraženija pri 353 K kada je došlo do navedenog pomicanja, no i pri nižim temperaturama dolazi do pomaka od 0,2 nm i pucanja te veze. Moguće objašnjenje rezultata je da je spomenuta regija izuzetno osjetljiva na temperaturu, te već pri 300 K može doći do pucanja. Mutacije to eventualno olakšavaju dodatnom destabilizacijom zavojnice H. Narušavanjem tih interakcija omogućen je dodatan ulazak vode oko His170, što je spomenuto u poglavlju 4.2.1. slikom 22. Iako pucanje te veze i

posljedična rotacija His170 nisu odlučujući za katalitički mehanizam, za pretpostaviti je da dolazi do smanjenja stabilnost aktivnog mjesta.

Opisane promjene u strukturi HRP i mHRP pri povišenoj temperaturi, iako zasebno minimalne, kumulativno mogu dovesti do gubitka katalitičke aktivnosti enzima



Slika 25. Prikazi prije i poslije pucanja vodikove veze Asp247 – His170. Sivom bojom označen je Phe221 koji ostvaruje π-π interakcije. Prikazane udaljenosti izražene su u Angstremima. Grafovima su prikazani pomaci prosjeka regije 246. do 247. aminokiseline, što se uzima kao Asp247 i neposredna okolina.

5. Zaključak

Provedeno je istraživanje utjecaja glikolizacije, kao i utjecaja Martellovih mutacija[1] na enzim peroksidazu hrena (HRP). Ustvrđeno je kako glikozilacija nema značajnog utjecaja na konformaciju enzima, u smislu da je sveukupna konformacija očuvana. Minimalne promjene primijećene su u regijama koje nisu od značaja, poput petlji. Konstrukcija aktivnog mjesta je očuvana. Usprkos tome, utvrđeno je da je enzim bez glikozilacije sveukupno nestabilniji, no glavnina dodatnih fluktuacija je lokalizirana na regije poput petlji. Regije od značaja, poput aktivnog mjesta, nisu značajno nestabilnije uslijed nedostatka glikozilacije. Na temelju iznesenih rezultata očekivano bi bilo zadržavanje aktivnosti uslijed gubitka glikozilacije. Smanjena stabilnost mogla bi ukazivati na kraći životni vijek enzima. Iznesena računalna predviđanja u sklopu istraživanja utjecaja glikozilacije na stabilnost enzima u skladu su s eksperimentalnim opažanjima[2,8-12].

Provedena je usporedba simulacija pri različitim temperaturama (26,85 °C (300 K), 44,85 °C (318 K), 59,85 °C (333 K) te 79,85 °C (353 K)) mHRP i HRP. Pri 300 K mHRP je stabilniji, s manje konformacijskih promjena i dodatnih fluktuacija u odnosu na HRP. To opažanje sasvim je u skladu u kontekstu Martellovih mutacija[1] koje su uvedene sa svrhom da stabiliziraju HRP. Pri višim temperaturama opaženo je više konformacijskih promjena u mHRP, te su opažene konformacijske promjene kojih nisu prisutne u HRP. Kod mHRP primijećene su povećane fluktuacije određenih petlji, koje su strukturno fleksibilni elementi te nemaju utjecaja na katalitičku aktivnost. Usprkos razlikama u konformaciji i jačih fluktuacija petlji, analizama simulacija je utvrđeno kako je mHRP sveukupno stabilniji i postojaniji od HRP. Najveća opažena razlika je pomicanje zavojnice E i pripadne petlje (140. do 150. aminokiseline) u mHRP pri 318 K, 333 K te 353 K, koje je kod HRP primijećeno isključivo pri 353 K.

Utvrđeno je da magnituda pomaka zavojnice E raste s temperaturom. To pomicanje zavojnice E nije dovelo do povećanja količine molekula vode u aktivnom mjestu, te oko His170 koji se nalazi ispod šupljine aktivnog mjesta i koordinira s hemom, no povećana je količina molekula vode sa strane hema gdje se nalazi spomenuta zavojnice. Pomicanje zavojnice E ima dodatni efekt na aktivno mjesto, a to je pomicanje Pro139 koji je bitan za katalitičku reakciju HRP. On se nalazi unutar šupljine aktivnog mjesta te ostvaruje vodikove veze s molekulama vode i/ili molekulom vodikovog peroksida, a pomicanjem zavojnice E on se translatira. To pomicanje Pro139 nije opaženo u simulaciji HRP pri 353 K iako je došlo do pomicanja

zavojnice E. Druga opažena promjena je pucanje vodikove veze između Asp247 i His170, što omogućava histidinu da se rotira čime dolaze do narušavanja π–π interakcija s Phe221. Ta promjena je opažena pri svim temperaturama, no izraženije kada je došlo do pomicanja zavojnice. Opisane promjene su zasebno razmjerno malene, no kumulativno mogu dovesti do smanjenja ili čak gubitka katalitičke aktivnosti enzima.

Mehanizam pomicanja zavojnice E događa se u tri koraka: konformacijska promjena petlje 255, pomak zavojnice H te konačno pomak zavojnice E. Konformacijska promjena u petlji 255 raste u magnitudi s temperaturom, te je uzrokovana rotacijom torzijskog kuta σ veze Cα ugljikovog atoma – karbonilnog ugljikovog atoma okosnice Thr257, koji se nalazi pored Asn255Asp mutacije. Ta rotacija opažena je isključivo u varijantama kada je došlo do pomaka zavojnice E, a nije opažena u varijantama gdje nije došlo do spomenutog pomaka. Kako je ta konformacijska promjena prvi korak u mehanizmu, to ukazuje da je ono odlučujući faktor koji se ponaša kao okidač mehanizma. Mutacija Asn255Asp uzrokuje gubitak glikozilacijskog mjesta na poziciji 255. Gubitak glikana destabilizira zavojnicu 255 i omogućava rotaciju torzijskog kuta Thr257, što dalje vodi do pomicanja zavojnice E. Zbog te mutacije u mHRP je snižena energetska barijera pomaka zavojnice E, pa je ono opaženo i pri 318 i 333 K dok u HRP nije opaženo pri tim temperaturama. Iz tog razloga opravdano je vjerovati kako pomicanje zavojnice E nije događaj isključivo povezan s Martellovim mutacijama, već ga one samo promoviraj kroz gubitak glikana i posljedično snižavanje energetske barijere te promjene.

Pomicanje zavojnice E u simulacijama mHRP počinje pri temperaturi od 318 K, te ga karakterizira promjena u tercijarnoj strukturi oko aktivnog mjesta te apsolutna retencija sekundarne strukture. Opažena promjena je u skladu s literaturno opisanim termičkim raspadom HRP enzima koji se odvija preko intermedijernog koraka *pre-molten globule* do čijeg prijelaza dolazi u temperaturnom rasponu od 313 K do 323 K (40 °C do 50 °C) [5,6].

6. Zahvale

Za početak, veliko hvala Antunu na izvrsnom mentorstvu, strpljenju, konstruktivnim kritikama, i toleriranju kasnonoćnih kvazikoherentnih mailova. Hvala Zoe, Katarini, Sanji, i Andrei na ugodnoj radnoj atmosferi i povremenoj pomoći sa tehnikalijama. Također velike zahvale profesoru Bertoši na prilici za izradu ovog rada, uviđanju eventualnog potencijala, vjeri da bi zapravo „moglo nekako ispasti“, zanimljivim predavanjima, i izvrsnom mentorstvu. Za kraj, hvala roditeljima na uvjetima; a najviše hvala baki i dedi na kasnim večerama, povremenom „taksiranju“, i drugim konkretnim formama pomoći koje su mi omogućile da se bavim svime čime se bavim i time na jedan ili drugi način, direktno ili indirektno, više ili manje pridonijele izradi ovog rada.

7. Literatura

[1] J. D. Martell, M. Yamagata, T. J. Deerinck, S. Phan, C. G. Kwa, M. H. Ellisman, J. R. Sanes, A. Y. Ting, *Nature Biotechnology* **34** (2016) 774-780.

[2] S. W. Michnick, P. H. Ear, C. Landry, M. K. Malleshaiah, V. Messier, *Methods in Molecular Biology* **756** (2011) 395-425.

[3] D. Shental-Bechor, Y. Levy, *PNAS* **105** (2008) 8256-8261.

[4] A. M. Azevedo, V. C. Martins, D. M. Prazers, V. Vojinović, J. M. Cabral, L. P. Fonseca, *Biotechnology Annual Review* **9** (2003) 199-247.

[5] K. Chattopadhyay, S. Mazumdar, *Biochemistry* **39** (2000) 263-270.

[6] K. Bamdad, B. Ranjbar, H. Naderi-Manesh, M. Sadeghi, *EXCLI Journal* **13** (2014) 611-622.

[7] P. Campomanes, U. Rothlisberger, M. Alfonso-Prieto, C. Rovira, *J. Am. Chem. Soc*. **137** (2015) 11170-11178.

[8] G. Hernandez-Cancel, D. Suazo-Davila, J. Medina-Guzman, M. Rosado-Gonzalez, L. M. Diaz-Vazquez, K. Griebenow, *Anal. Chim. Acta*. **854** (2015) 129-139.

[9] B. Meyer, H. Moller, *Top. Curr. Chem*. **267** (2007) 187-251.

[10] H. Sun Lee, W. Im, *Scientific Reports* **7** (2017) 12659.

[11] S. Škulj, A. Barišić, N. Mutter, O. Spadiut, I. Barišić, B. Bertošsa, *CSBJ* **20** (2022) 3096-3105.

[12] J. W. Tams, K. G. Welinder, *Analytical Biochemistry* **228** (1995) 48-55.

[13] B. Y. Yang, J. S. S. Gray, R. Montgomery, *Carbohydrate Research* **287** (1996) 203-212.

[14] F. W. Krainer, C. Gmeiner, L. Neutsch, M. Windwarder, R. Pletzenauer, C. Herwig, F. Altmann, A. Glieder, O. Spadiut, *Sci. Rep*. **3** (2013) 3279.

[15] D. Humer, O. Spadiut, *Int. J. Mol. Sci*. **20** (2019) 916.

[16] A. Barišić, Termodinamika procesa na međupovršini inertna tvar/vodena otopina elektorlita, Doktorska disertacija, Prirodoslovno-Matematički Fakultet, Sveučilište u Zagrebu, 2020.

[17] B. Hess, H. Bekker, H. J. C. Berendsen, J. G. E. M. Fraaije, *J. Comput. Chem*. **18** (1997) 1463-1472.

[18] P. Bauer, B. Hess, E. Lindahl, (2022) GROMACS 2022.1 Manual.

[19] G. I. Berglund, G. H. Carlsson, A. T. Smith, H. Szoke, A. Heneriksen, J. Hajdu, *Nature* **417** (2002) 463-468.

[20] A. Pekarsky, L. Veiter, V. Rajamanickam, C. Herwig, C. Grunwald-Gruber, F. Altmann, O. Spadiut, *Microb. Cell. Fact*. **17** (2018) 183.

[21] S. Jo, T. Kim, V. G. Iyer, W. Im, *J. Comput. Chem*. **29** (2008) 1859-1865.

[22] B. R. Brooks, C. L. Brooks III, A. D. Mackerell Jr., L. Nilsson, R. J. Petrella, B. Roux, Y. Won, G. Archontis, C. Bartels, S. Boresch, A. Caflisch, L. Caves, Q. Cui, A. R. Dinner, M. Feig, S. Fischer, J. Gao, M. Hodoscek, W. Im, K. Kuczera, T. Lazaridis, J. Ma, V. Ovchinnikov, E. Paci, R. W. Pastor, C. B. Post, J. Z. Pu, M. Schaefer, B. Tidor, R. M. Venable, H. L. Woodcock, X. Wu, W. Yang, D. M. York, M. Karplus, *J. Comput. Chem*. **30** (2009) 1545-1614.

[23] J. Lee, X. Cheng, J. M. Swails, M. S. Yeom, P. K. Eastman, J. A. Lemkul, S. Wei, J. Buckner, J. C. Jeong, Y. Qi, S. Jo, V. S. Pande, D. A. Case, C. L. Brooks, A. D. MacKerell Jr. J. B. Klauda, W. Im, *J. Chem. Theory Comput.* **12** (2016) 405-413.

[24] J. Huang, S. Rauscher, G. Nawrocki, T. Ran, M. Feig, B. L. de Groot, H. Grubmuller, A. D. MacKerell Jr., *Nature methods* **14** (2017) 71 – 73.

[25] H. J. C. Berendsen, J. P. M. Postma, W. F. van Gusteren, A. DiNola, J. R. Haak, *J. Chem. Phys*. **81** (1984) 3684.

[26] S. Nose, *Molecular Physics*, **52** (1984) 255-268.

[27] M. Parrinello, A. Rahman, *Journal of Applied Physics* **52** (1981) 7182.

[28] U. Essmann, L. Perera, M. L. Berkowitz, T. Darden, H. Lee, L. G. Pedersen, *J. Chem. Phys*. **103** (1995) 8577.

[29] D. Van Der Spoel, E. Lindahl, B. Hess, G. Groenhof, A. E. Mark, H. J. C. Berendsen, *J. Comput. Chem*. **26** (2005) 1701-1718.

[30] W. Humphrey, A. Dalke, K. Schulten, *J. Molec. Graphics* **14** (1996) 33-38.

[31] S. K. Fareny, M. B. Dolinska, Y. V. Sergeev, *J. Anal. Pharm. Res*. **7** (2018) 621-632.

[32] M. Minguez-Toral, B. Cuevas-Zuviria, M. Garrido-Arandia, L. F. Pacios, *Scientific Reports* **10** (2020) 13424.

[33] A. Marce-Grau, J. Dalton, J. Lopez-Pison, M. C. Garcia-Jimenez, L. Monge-Galindo, E. Cuenca-Leon, J. Giraldo, A. Macaya, *Orphanet Journal of Rare Diseases* **11** (2016) 38.

[34] J. D. Hunter, *Computing in Science & Engineering* **9** (2007) 90-95.

[35] P. Raybaut, (2009) Spyder-documentation.

[36] G. Van Rossum, F. L. Drake, Python 3 Reference Manual (2009), Scotts Valley, CA: CreateSpace.

8. Sažetak

SAŽETAK

Sveučilište u Zagrebu

Prirodoslovno-matematički fakultet

Kemijski odsjek

In silico istraživanje utjecaja glikana i mutacija na rekombinantni enzim peroksidaza hrena

Matej Kožić

Zavod za fizikalnu kemiju
Horvatovac 102a, Zagreb, Hrvatska

Provedeno je istraživanje utjecaja glikozilacije, kao i utjecaja Martellovih mutacija (razvijenih u svrhu korištenja enzima peroksidaze hrena (HRP) kao biosenzora) na enzim HRP te njegovu temperaturnu stabilnost. Provedeno je osam simulacija molekulske dinamike, svaka u trajanju od 500 ns, glikoziliranih oblika divljeg tipa i mutiranog oblika HRP pri temperaturama odabranim na temelju dostupnih eksperimentalnih podataka: 300 K, 318 K, 333 K i 353 K. Pored toga, provedene su i simulacije neglikoziliranog divljeg tipa HRP pri 300 K i neglikoziliranog rekombinantnog tipa HRP pri 300 K, također svaka u trajanju od 500 ns. Predstavljena je i opisana metoda analize trajektorija nazvana mapa trajektorija (engl. *Trajectory Map*) za koju je u sklopu ovog rada napisana skripta u programskom jeziku Python, uz koju su također provedene i uobičajene metode analize trajektorija. Analizom simulacija utvrđen je utjecaj glikozilacija na divlju i rekombinantnu varijantu enzima HRP, kao i utjecaj mutacija na enzim i na njegovu temperaturnu stabilnost. Na temelju rezultata simulacija i usporedbe s eksperimentalno dostupnim podacima predložen je mehanizam promjene u tercijarnoj strukturi kao i objašnjenje literaturno dostupnog podatka o intermedijernom stanju termičkog raspada enzima HRP nazvanog engl. *pre-molten globule*.

(47 stranica, 25 slika, 36 literaturna navoda, izvornik na hrvatskom jeziku)

Ključne riječi: peroksidaza hrena, biosenzor, molekulska dinamika, mapa trajektorija, temperaturna stabilnost enzima HRP

Mentor: prof. dr. sc. Branimir Bertoša

9. Summary

SUMMARY

University of Zagreb

Faculty of Science

Department of Chemistry

In silico study of the effects of glycans and mutations on a recombinant enzyme horseradish peroxidase

Matej Kožić

Division of Physical Chemistry
Horvatovac 102a, Zagreb, Croatia

The goal of the presented research was to study the effects of glycosylation, as well as Martell mutations (developed for the purpose of using horseradish peroxidase (HRP) enzyme as a biosenzor in protein-fragment complementation assays), on the enzyme and its thermal stability. Molecular dynamics (MD) simulations (the duration of 500 ns) of eight systems were conducted: wild and recombinant types of HRP at different temperatures selected in accordance with available experimental data: 300 K, 318 K, 333 K, and 353 K. In addition, MD simulations of deglycosylated wild type and recombinant type of HRP at 300 K were conducted. A novel method of trajectory analysis (made using custom scripts for Python programing language) was developed in the frame of this study ("trajectory map analysis") and used alongside standard trajectory analysis methods. Effects of glycosylation on the wild type and the recombinant type of HRP were determined, as well as the effects of mutations on the enzyme and its thermal stability. Finally, a change in tertiary structure around the catalytic center of the enzyme was presented and described in detail. That conformational change and its respective molecular mechanism was proposed as a theoretical explanation of a literary experimental result that describes the thermal unfolding of the HRP enzyme as a two-step process involving an intermediate state referenced in literature as pre-molten globule state.

(47 pages, 25 figures, 36 references, the original of Croatian)

Keywords: Horseradish Peroxidase, Protein-fragment Complementation Assay, Molecular Dynamics, Pre-Molten Globule, Trajectory Map

Mentor: Dr. Branimir Bertoša

10. Dodatak

Dodatak 1. Korišteni brojač molekula vode za program VMD.

```
set output [open "353_5a_his.dat" w]

for {set i 1 } { $i < 501 } {incr i } {

set sel [atomselect top "name OH2 and within 5 of resid 170" frame $i]

puts $output [$sel num]

}
close $output
```

Dodatak 2. Korišteni kalkulator torzijskih kutova definirane regije okosnice proteina za program VMD.

```
set output [open "diheds_w353.dat" w]

for {set i 1 } { $i < 501 } {incr i } {

set v1 [measure dihed {3870 3872 3879 3880} frame $i]
set v2 [measure dihed {3880 3879 3881 3885} frame $i]
set v3 [measure dihed {3879 3881 3885 3893} frame $i]
set v4 [measure dihed {3881 3885 3893 3894} frame $i]
set v5 [measure dihed {3894 3893 3895 3897} frame $i]
set v6 [measure dihed {3893 3895 3897 3906} frame $i]
set v7 [measure dihed {3895 3897 3906 3907} frame $i]
set v8 [measure dihed {3907 3906 3908 3910} frame $i]
set v9 [measure dihed {3906 3908 3910 3916} frame $i]
set v10 [measure dihed {3908 3910 3916 3917} frame $i]
set v11 [measure dihed {3917 3916 3918 3920} frame $i]
set v12 [measure dihed {3916 3918 3920 3930} frame $i]
set v13 [measure dihed {3918 3920 3930 3931} frame $i]
puts $output "$v1|$v2|$v3|$v4|$v5|$v6|$v7|$v8|$v9|$v10|$v11|$v12|$v13"

}
close $output

# source ../../diheds/diheds.txt
```

Dodatak 3. Korištena skripta za izradu grafova vremenske ovisnosti količine molekula vode oko regija od interesa na temelju skripte za VMD (dodatak 1), za programski jezik Python.

```
# -*- coding: utf-8 -*-
"""
Created on Sat Jan 29 14:07:45 2022

@author: mtjkzc
"""

#####    water counter     ####

import numpy as np
import pandas as pd
from numpy import loadtxt
import matplotlib.pyplot as plt
from matplotlib.ticker import (MultipleLocator)


file1 = "../wat_dat/300_5a_as38.dat"
file2 = "../wat_dat/318_5a_as38.dat"
file3 = "../wat_dat/333_5a_as38.dat"
file4 = "../wat_dat/353_5a_as38.dat"

title1 = "Broj molekula vode 5 A od hema i 38"
savefig1 = "../figs/w_water_5A_as38.png"
size = 800

l1 = "300 K"
l2 = "318 K"
l3 = "333 K"
l4 = "353 K"
##########################################################################
######

ax = plt.subplot(111)

lines1 = loadtxt(file1, comments="#", delimiter=",", unpack=False)
df1 = pd.DataFrame(lines1)
y1=df1[0]

lines2 = loadtxt(file2, comments="#", delimiter=",", unpack=False)
df2 = pd.DataFrame(lines2)
y2=df2[0]

lines3 = loadtxt(file3, comments="#", delimiter=",", unpack=False)
df3 = pd.DataFrame(lines3)
y3=df3[0]
```

```
lines4 = loadtxt(file4, comments="#", delimiter=",", unpack=False)
df4 = pd.DataFrame(lines4)
y4=df4[0]

x = np.array(np.arange(0,500,1))

avg = 15
y1a = y1.rolling(avg).mean()
y2a = y2.rolling(avg).mean()
y3a = y3.rolling(avg).mean()
y4a = y4.rolling(avg).mean()

color1 = "greenyellow"
color2 = "limegreen"
color3 = "green"
color4 = "darkolivegreen"

plt.plot(x,y1, linewidth=0.3, color=color1, label = l1)
plt.plot(x,y2, linewidth=0.3, color=color2, label = l2)
plt.plot(x,y3, linewidth=0.3, color=color3, label = l3)
plt.plot(x,y4, linewidth=0.3, color=color4, label = l4)

ax.legend(loc='upper left',
          fancybox=True, shadow=True, prop={'size': 7.5})


plt.plot(x,y1a, linewidth=1, color=color1, label = l1)
plt.plot(x,y2a, linewidth=1, color=color2, label = l2)
plt.plot(x,y3a, linewidth=1, color=color3, label = l3)
plt.plot(x,y4a, linewidth=1, color=color4, label = l4)

ax.set_xlim(0, 500)
plt.yticks(np.arange(0, 40, step=2))
plt.xticks(np.arange(0, 550, step=50))
plt.title(title1)
ax.set_ylim(0)
plt.xlabel("t / ns")
plt.ylabel ("Broj molekula vode")
ax.yaxis.set_minor_locator(MultipleLocator(1))
ax.xaxis.set_minor_locator(MultipleLocator(10))


plt.savefig(savefig1, dpi= size)


##########################################################################
##########################################################################
```

Dodatak4. Korištena skripta za izradu grafova promjena torzijskih kutova na temelju skripte za program VMD (dodatak 2), za programski jezik Python.

```
# -*- coding: utf-8 -*-
"""
Created on Sat Jan 29 14:07:45 2022

@author: mtjkzc
"""


import numpy as np
import pandas as pd
from numpy import loadtxt
import matplotlib.pyplot as plt
from matplotlib.ticker import (MultipleLocator)

file1 = "diheds_w300.dat"
file2 = "diheds_w318.dat"
file3 = "diheds_w333.dat"
file4 = "diheds_w353.dat"

savefig = "../figs/w_dihed_12.png"
num = 12
avg = 15

##########################################################################
######

matrix1 = loadtxt(file1, comments="#", delimiter="|", unpack=False)
matrix1 = np.transpose(matrix1)

matrix2 = loadtxt(file2, comments="#", delimiter="|", unpack=False)
matrix2 = np.transpose(matrix2)

matrix3 = loadtxt(file3, comments="#", delimiter="|", unpack=False)
matrix3 = np.transpose(matrix3)

matrix4 = loadtxt(file4, comments="#", delimiter="|", unpack=False)
matrix4 = np.transpose(matrix4)


##########################################################################
######

graph1 = matrix1[num,:]
```

```
graph2 = matrix2[num,:]
graph3 = matrix3[num,:]
graph4 = matrix4[num,:]

y1 = pd.DataFrame(graph1)
y2 = pd.DataFrame(graph2)
y3 = pd.DataFrame(graph3)
y4 = pd.DataFrame(graph4)
y1a = y1.rolling(avg).mean()
y2a = y2.rolling(avg).mean()
y3a = y3.rolling(avg).mean()
y4a = y4.rolling(avg).mean()

color1 = "greenyellow"
color2 = "limegreen"
color3 = "green"
color4 = "darkolivegreen"
ax2 = plt.subplot(111)
ax2.axhline(0, color = "gray", linewidth = 0.3, linestyle = "--")

ax2.plot(graph4, linewidth = 0.4, color = color4,label = "353 K")
ax2.plot(graph3, linewidth = 0.4, color = color3,label = "333 K")
ax2.plot(graph2, linewidth = 0.4, color = color2,label = "318 K")
ax2.plot(graph1, linewidth = 0.4, color = color1,label = "300 K")

#plt.plot(y1a, linewidth=1, color=color1)
#plt.plot(y2a, linewidth=1, color=color2)
#plt.plot(y3a, linewidth=1, color=color3)
#plt.plot(y4a, linewidth=1, color=color4)


ax2.set_ylim(-180,180)
ax2.set_xlim(0,500)
ax2.set_yticks(np.arange(-180, 190, step=30))
ax2.set_xticks(np.arange(0, 501, step=50))
ax2.yaxis.set_minor_locator(MultipleLocator(10))
ax2.xaxis.set_minor_locator(MultipleLocator(10))
ax2.set_ylabel (r'$\theta \ / \circ$')
ax2.set_xlabel ("t / ns")

plt.legend( bbox_to_anchor=(0.45, -0.08, 0, 0), shadow = True, ncol = 4,
fontsize = 6)

plt.savefig(savefig, dpi = 800, bbox_inches='tight')

#########################################################################
#########################################################################
```

Dodatak 5. Korištena skripta za izradu Rg grafova na temelju .xvg datoteke dobivene GROMACS programom, za programski jezik Python.

```
-*- coding: utf-8 -*-
"""
Created on Sat Jan 29 10:32:09 2022

@author: mtjkzc
"""
import numpy as np
from numpy import loadtxt
import pandas as pd
import matplotlib.pyplot as plt
from pandas import Series
import matplotlib.pyplot as plt
from matplotlib.ticker import (MultipleLocator)

path = "../xvg/"
filename = "w300K_gyr.xvg"

file = str(str(path) + str(filename))
xvg_file= open(file)
dataframe = pd.DataFrame(xvg_file)
data = pd.Series()
i = 0
limit = len(dataframe) + 1
marker = "@"
try:
    while i != limit :
        #reader(i, data)
        read = dataframe.iloc[i]
        if marker in str(read):
            i=i+1
            while i != limit:
                data = data.append(read)
                print(i)
                read = dataframe.iloc[i]
                i = i + 1
            else:
                print ("end")
        i=i+1
except IndexError: pass
data = data.iloc[14:] #-----------------------------------------------
---
i = 0
limit = len(dataframe)
coords = pd.Series()
try:
    while i <= limit :
```

```
        line = data.iloc[i]
        line = line.replace(" ",",")
        line = line.replace(",,,,,",",")
        line = line.replace(",,,,",",")
        line = line.replace(",,,",",")
        line = line.replace(",,",",")
        line = line[1:] #-----------------------------------------------
--
        coords = coords.append(pd.Series(line))
        i = i + 1
        print("Reading row:", i)
except IndexError : pass
coords = coords.str.split(",", expand = True )
coords = np.array(coords, dtype = float)
x1 = coords[:,0] / 1000
y1 = coords[:,1]
xa1 = coords[:,0] / 1000
ya1 = coords[:,2]
xb1 = coords[:,0] / 1000
yb1 = coords[:,3]
xc1 = coords[:,0] / 1000
yc1 = coords[:,4]
################################SECOND
GRAPH########################################

path = "../xvg/"
filename = "w318K_gyr.xvg"

file = str(str(path) + str(filename))
xvg_file= open(file)
dataframe = pd.DataFrame(xvg_file)
data = pd.Series()
i = 0
limit = len(dataframe) + 1
marker = "@"
try:
    while i != limit :
        #reader(i, data)
        read = dataframe.iloc[i]
        if marker in str(read):
            i=i+1
            while i != limit:
                data = data.append(read)
                print(i)
                read = dataframe.iloc[i]
                i = i + 1
            else:
                print ("end")
        i=i+1
```

```
except IndexError: pass
data = data.iloc[14:] #-----------------------------------------------
---
i = 0
limit = len(dataframe)
coords = pd.Series()
try:
    while i <= limit :
        line = data.iloc[i]
        line = line.replace(" ",",")
        line = line.replace(",,,,,",",")
        line = line.replace(",,,,",",")
        line = line.replace(",,,",",")
        line = line.replace(",,",",")
        line = line[1:] #-----------------------------------------------
--
        coords = coords.append(pd.Series(line))
        i = i + 1
        print("Reading row:", i)
except IndexError : pass
coords = coords.str.split(",", expand = True )
coords = np.array(coords, dtype = float)



x2 = coords[:,0] / 1000
y2 = coords[:,1]
xa2 = coords[:,0] / 1000
ya2 = coords[:,2]
xb2 = coords[:,0] / 1000
yb2 = coords[:,3]
xc2 = coords[:,0] / 1000
yc2 = coords[:,4]

##############################################################################
#####

path = "../xvg/"
filename = "w333K_gyr.xvg"

file = str(str(path) + str(filename))
xvg_file= open(file)
dataframe = pd.DataFrame(xvg_file)
data = pd.Series()
i = 0
limit = len(dataframe) + 1
marker = "@"
try:
    while i != limit :
```

```
        #reader(i, data)
        read = dataframe.iloc[i]
        if marker in str(read):
            i=i+1
            while i != limit:
                data = data.append(read)
                print(i)
                read = dataframe.iloc[i]
                i = i + 1
            else:
                print ("end")
        i=i+1
except IndexError: pass
data = data.iloc[14:] #-------------------------------------------------
---
i = 0
limit = len(dataframe)
coords = pd.Series()
try:
    while i <= limit :
        line = data.iloc[i]
        line = line.replace(" ",",")
        line = line.replace(",,,,,",",")
        line = line.replace(",,,,",",")
        line = line.replace(",,,",",")
        line = line.replace(",,",",")
        line = line[1:] #-----------------------------------------------
--
        coords = coords.append(pd.Series(line))
        i = i + 1
        print("Reading row:", i)
except IndexError : pass
coords = coords.str.split(",", expand = True )
coords = np.array(coords, dtype = float)



x3 = coords[:,0] / 1000
y3 = coords[:,1]
xa3 = coords[:,0] / 1000
ya3 = coords[:,2]
xb3 = coords[:,0] / 1000
yb3 = coords[:,3]
xc3 = coords[:,0] / 1000
yc3 = coords[:,4]

#########################################################################
######
```

```
path = "../xvg/"
filename = "w353K_gyr.xvg"

file = str(str(path) + str(filename))
xvg_file= open(file)
dataframe = pd.DataFrame(xvg_file)
data = pd.Series()
i = 0
limit = len(dataframe) + 1
marker = "@"
try:
    while i != limit :
        #reader(i, data)
        read = dataframe.iloc[i]
        if marker in str(read):
            i=i+1
            while i != limit:
                data = data.append(read)
                print(i)
                read = dataframe.iloc[i]
                i = i + 1
            else:
                print ("end")
        i=i+1
except IndexError: pass
data = data.iloc[14:] #-------------------------------------------------
---
i = 0
limit = len(dataframe)
coords = pd.Series()
try:
    while i <= limit :
        line = data.iloc[i]
        line = line.replace(" ",",")
        line = line.replace(",,,,,",",")
        line = line.replace(",,,,",",")
        line = line.replace(",,,",",")
        line = line.replace(",,",",")
        line = line[1:] #-------------------------------------------------
--
        coords = coords.append(pd.Series(line))
        i = i + 1
        print("Reading row:", i)
except IndexError : pass
coords = coords.str.split(",", expand = True )
coords = np.array(coords, dtype = float)
```

```
x4 = coords[:,0] / 1000
y4 = coords[:,1]
xa4 = coords[:,0] / 1000
ya4 = coords[:,2]
xb4 = coords[:,0] / 1000
yb4 = coords[:,3]
xc4 = coords[:,0] / 1000
yc4 = coords[:,4]

avg = 15
y1x = pd.Series(data=y1)
y2x = pd.Series(data=y2)
y3x = pd.Series(data=y3)
y4x = pd.Series(data=y4)

y1x = y1x.rolling(avg).mean()
y2x = y2x.rolling(avg).mean()
y3x = y3x.rolling(avg).mean()
y4x = y4x.rolling(avg).mean()


##########################################################################
######

title = "c"
xlabel = "t / ns"
ylabel = "Rg / nm"
size = 800
savefig = "../figs/w_gyr_master.png"
legend_loc = "upper left"

color1 = "greenyellow"
color2 = "limegreen"
color3 = "green"
color4 = "darkolivegreen"

plt.plot(x1,y1x, linewidth = 1, color = color1, label = "300 K")
plt.plot(x2,y2x, linewidth = 1, color = color2, label = "318 K")
plt.plot(x3,y3x, linewidth = 1, color = color3, label = "333 K")
plt.plot(x4,y4x, linewidth = 1, color = color4, label = "353 K")

plt.plot(x1,y1, linewidth = 0.3, color = color1, )
plt.plot(x2,y2, linewidth = 0.3, color = color2, )
plt.plot(x3,y3, linewidth = 0.3, color = color3, )
plt.plot(x4,y4, linewidth = 0.3, color = color4, )


plt.xlabel(xlabel)
plt.ylabel (ylabel)
```

```
plt.yticks(np.arange(1.94, 2.11, step=0.01))
plt.xticks(np.arange(0, 550, step=50))
ax = plt.subplot(111)
ax.set_ylim(1.95)
#ax.set_xlim(0)
ax.yaxis.set_minor_locator(MultipleLocator(0.005))
ax.xaxis.set_minor_locator(MultipleLocator(5))

legend = plt.legend(loc= legend_loc, fancybox=True, shadow=True)

plt.savefig(savefig, dpi=size)
##########################################################################
######
```

Dodatak 6. Korištena skripta za izradu RMSD grafova na temelju .xvg datoteke dobivene GROMACS programom, za programski jezik Python.

```
# -*- coding: utf-8 -*-
"""
Created on Sat Jan 29 10:32:09 2022

@author: mtjkzc
"""
import numpy as np
from numpy import loadtxt
import pandas as pd
import matplotlib.pyplot as plt
from pandas import Series
import matplotlib.pyplot as plt
from matplotlib.ticker import (MultipleLocator)

path = "../xvg/"
filename = "w300K_rmsd.xvg"

file = str(str(path) + str(filename))
xvg_file= open(file)
dataframe = pd.DataFrame(xvg_file)
data = pd.Series()
i = 0
limit = len(dataframe) + 1
marker = "@TYPE xy"
try:
    while i != limit :
        #reader(i, data)
        read = dataframe.iloc[i]
```

```
        if marker in str(read):
            i=i+1
            while i != limit:
                data = data.append(read)
                print(i)
                read = dataframe.iloc[i]
                i = i + 1
            else:
                print ("end")
        i=i+1
except IndexError: pass
data = data.iloc[2:] #-----------------------------------------------------
--
i = 0
limit = len(dataframe)
coords = pd.Series()
try:
    while i <= limit :
        line = data.iloc[i]
        line = line.replace(" ",",")
        line = line.replace(",,,,,",",")
        line = line.replace(",,,,",",")
        line = line.replace(",,,",",")
        line = line.replace(",,",",")
        if i == 0 :
            line = line[1:] #-----------------------------------------
------
        coords = coords.append(pd.Series(line))
        i = i + 1
        print("Reading row:", i)
except IndexError : pass
coords = coords.str.split(",", expand = True )
coords = np.array(coords, dtype = float)
x1 = coords[:,0] / 1000
y1 = coords[:,1] * 10

################################SECOND
GRAPH########################################

path = "../xvg/"
filename = "w318K_rmsd.xvg"


file = str(str(path) + str(filename))
xvg_file= open(file)
dataframe = pd.DataFrame(xvg_file)
data = pd.Series()
i = 0
limit = len(dataframe) + 1
```

```
marker = "@TYPE xy"
try:
    while i != limit :
        #reader(i, data)
        read = dataframe.iloc[i]
        if marker in str(read):
            i=i+1
            while i != limit:
                data = data.append(read)
                print(i)
                read = dataframe.iloc[i]
                i = i + 1
            else:
                print ("end")
        i=i+1
except IndexError: pass
data = data.iloc[2:] #-------------------------------------------------
--
i = 0
limit = len(dataframe)
coords = pd.Series()
try:
    while i <= limit :
        line = data.iloc[i]
        line = line.replace(" ",",")
        line = line.replace(",,,,,",",")
        line = line.replace(",,,,",",")
        line = line.replace(",,,",",")
        line = line.replace(",,",",")
        if i == 0 :
            line = line[1:] #-----------------------------------------
------
        coords = coords.append(pd.Series(line))
        i = i + 1
        print("Reading row:", i)
except IndexError : pass
coords = coords.str.split(",", expand = True )
coords = np.array(coords, dtype = float)


x2 = coords[:,0] / 1000
y2 = coords[:,1] * 10

#########################################################################
####################

path = "../xvg/"
filename = "w333K_rmsd.xvg"
```

```
file = str(str(path) + str(filename))
xvg_file= open(file)
dataframe = pd.DataFrame(xvg_file)
data = pd.Series()
i = 0
limit = len(dataframe) + 1
marker = "@TYPE xy"
try:
    while i != limit :
        #reader(i, data)
        read = dataframe.iloc[i]
        if marker in str(read):
            i=i+1
            while i != limit:
                data = data.append(read)
                print(i)
                read = dataframe.iloc[i]
                i = i + 1
            else:
                print ("end")
        i=i+1
except IndexError: pass
data = data.iloc[2:] #-----------------------------------------------------
--
i = 0
limit = len(dataframe)
coords = pd.Series()
try:
    while i <= limit :
        line = data.iloc[i]
        line = line.replace(" ",",")
        line = line.replace(",,,,,",",")
        line = line.replace(",,,,",",")
        line = line.replace(",,,",",")
        line = line.replace(",,",",")
        if i == 0 :
            line = line[1:] #-----------------------------------------
------
        coords = coords.append(pd.Series(line))
        i = i + 1
        print("Reading row:", i)
except IndexError : pass
coords = coords.str.split(",", expand = True )
coords = np.array(coords, dtype = float)
x3 = coords[:,0] / 1000
y3 = coords[:,1] * 10

################################THIRD
GRAPH########################################
```

```
path = "../xvg/"
filename = "w353K_rmsd.xvg"

file = str(str(path) + str(filename))
xvg_file= open(file)
dataframe = pd.DataFrame(xvg_file)
data = pd.Series()
i = 0
limit = len(dataframe) + 1
marker = "@TYPE xy"
try:
    while i != limit :
        #reader(i, data)
        read = dataframe.iloc[i]
        if marker in str(read):
            i=i+1
            while i != limit:
                data = data.append(read)
                print(i)
                read = dataframe.iloc[i]
                i = i + 1
            else:
                print ("end")
        i=i+1
except IndexError: pass
data = data.iloc[2:] #-------------------------------------------------
--
i = 0
limit = len(dataframe)
coords = pd.Series()
try:
    while i <= limit :
        line = data.iloc[i]
        line = line.replace(" ",",")
        line = line.replace(",,,,,",",")
        line = line.replace(",,,,",",")
        line = line.replace(",,,",",")
        line = line.replace(",,",",")
        if i == 0 :
            line = line[1:] #-----------------------------------------
------
        coords = coords.append(pd.Series(line))
        i = i + 1
        print("Reading row:", i)
except IndexError : pass
coords = coords.str.split(",", expand = True )
coords = np.array(coords, dtype = float)
```

```
x4 = coords[:,0] / 1000
y4 = coords[:,1] * 10

################################SECOND
GRAPH########################################
#########################################################################
######

title = "a)"
xlabel = "t / ns"
ylabel = "RMSD / A"
size = 800
savefig = "../figs/w_rmsd_master.png"
legend_loc = "upper left"
color1 = "greenyellow"
color2 = "limegreen"
color3 = "green"
color4 = "darkolivegreen"

plt.plot(x1,y1, linewidth = 1, color = color1, label = "300 K")
plt.plot(x2,y2, linewidth = 1, color = color2, label = "318 K")
plt.plot(x3,y3, linewidth = 1, color = color3, label = "333 K")
plt.plot(x4,y4, linewidth = 1, color = color4, label = "353 K")


#plt.title(title, loc="center")
plt.xlabel(xlabel)
plt.ylabel (ylabel)
plt.yticks(np.arange(0, 4.5, step=0.5))
plt.xticks(np.arange(0, 550, step=50))
ax = plt.subplot(111)
ax.set_ylim(0)
#ax.set_xlim(0)
ax.yaxis.set_minor_locator(MultipleLocator(0.05))
ax.xaxis.set_minor_locator(MultipleLocator(5))

legend = plt.legend(loc= legend_loc, fancybox=True, shadow=True)

plt.savefig(savefig, dpi=size)
#########################################################################
```

Dodatak 7. Korištena skripta za izradu RMSF grafova na temelju .xvg datoteke dobivene GROMACS programom, za programski jezik Python.

```
# -*- coding: utf-8 -*-
"""
Created on Sat Jan 29 10:32:09 2022
```

```
@author: mtjkzc
"""
import numpy as np
from numpy import loadtxt
import pandas as pd
import matplotlib.pyplot as plt
from pandas import Series
import matplotlib.pyplot as plt
from matplotlib.ticker import (MultipleLocator)

path = "../xvg/"
filename = "w300K_rmsf.xvg"

file = str(str(path) + str(filename))
xvg_file= open(file)
dataframe = pd.DataFrame(xvg_file)
data = pd.Series()
i = 0
limit = len(dataframe) + 1
marker = "@TYPE xy"
try:
    while i != limit :
        #reader(i, data)
        read = dataframe.iloc[i]
        if marker in str(read):
            i=i+1
            while i != limit:
                data = data.append(read)
                print(i)
                read = dataframe.iloc[i]
                i = i + 1
            else:
                print ("end")
        i=i+1
except IndexError: pass
data = data.iloc[1:] #-------------------------------------------------
--
i = 0
limit = len(dataframe)
coords = pd.Series()
try:
    while i <= limit :
        line = data.iloc[i]
        line = line.replace(" ",",")
        line = line.replace(",,,,,",",")
        line = line.replace(",,,,",",")
        line = line.replace(",,,",",")
        line = line.replace(",,",",")
        if i == 0 :
```

```
            line = line[0:] #-----------------------------------------
------
        coords = coords.append(pd.Series(line))
        i = i + 1
        print("Reading row:", i)
except IndexError : pass
coords = coords.str.split(",", expand = True )
coords = np.array(coords, dtype = float)
x1 = coords[:,0]
y1 = coords[:,1] * 10

##############################SECOND
GRAPH########################################

path = "../xvg/"
filename = "w318K_rmsf.xvg"

file = str(str(path) + str(filename))
xvg_file= open(file)
dataframe = pd.DataFrame(xvg_file)
data = pd.Series()
i = 0
limit = len(dataframe) + 1
marker = "@TYPE xy"
try:
    while i != limit :
        #reader(i, data)
        read = dataframe.iloc[i]
        if marker in str(read):
            i=i+1
            while i != limit:
                data = data.append(read)
                print(i)
                read = dataframe.iloc[i]
                i = i + 1
            else:
                print ("end")
        i=i+1
except IndexError: pass
data = data.iloc[1:] #-----------------------------------------------
--
i = 0
limit = len(dataframe)
coords = pd.Series()
try:
    while i <= limit :
        line = data.iloc[i]
        line = line.replace(" ",",")
        line = line.replace(",,,,,",",")
```

```
        line = line.replace(",,,,",",")
        line = line.replace(",,,",",")
        line = line.replace(",,",",")
        if i == 0 :
            line = line[0:] #-----------------------------------------
------
        coords = coords.append(pd.Series(line))
        i = i + 1
        print("Reading row:", i)
except IndexError : pass
coords = coords.str.split(",", expand = True )
coords = np.array(coords, dtype = float)
x2 = coords[:,0]
y2 = coords[:,1] * 10

#########################################################################
#######

path = "../xvg/"
filename = "w333K_rmsf.xvg"

file = str(str(path) + str(filename))
xvg_file= open(file)
dataframe = pd.DataFrame(xvg_file)
data = pd.Series()
i = 0
limit = len(dataframe) + 1
marker = "@TYPE xy"
try:
    while i != limit :
        #reader(i, data)
        read = dataframe.iloc[i]
        if marker in str(read):
            i=i+1
            while i != limit:
                data = data.append(read)
                print(i)
                read = dataframe.iloc[i]
                i = i + 1
            else:
                print ("end")
        i=i+1
except IndexError: pass
data = data.iloc[1:] #-----------------------------------------------
--
i = 0
limit = len(dataframe)
coords = pd.Series()
try:
```

```
    while i <= limit :
        line = data.iloc[i]
        line = line.replace(" ",",")
        line = line.replace(",,,,,",",")
        line = line.replace(",,,,",",")
        line = line.replace(",,,",",")
        line = line.replace(",,",",")
        if i == 0 :
            line = line[0:] #-----------------------------------------
------
        coords = coords.append(pd.Series(line))
        i = i + 1
        print("Reading row:", i)
except IndexError : pass
coords = coords.str.split(",", expand = True )
coords = np.array(coords, dtype = float)
x3 = coords[:,0]
y3 = coords[:,1] * 10

#########################################################################
######
path = "../xvg/"
filename = "w353K_rmsf.xvg"

file = str(str(path) + str(filename))
xvg_file= open(file)
dataframe = pd.DataFrame(xvg_file)
data = pd.Series()
i = 0
limit = len(dataframe) + 1
marker = "@TYPE xy"
try:
    while i != limit :
        #reader(i, data)
        read = dataframe.iloc[i]
        if marker in str(read):
            i=i+1
            while i != limit:
                data = data.append(read)
                print(i)
                read = dataframe.iloc[i]
                i = i + 1
            else:
                print ("end")
        i=i+1
except IndexError: pass
data = data.iloc[1:] #-----------------------------------------------
--
i = 0
```

```
limit = len(dataframe)
coords = pd.Series()
try:
    while i <= limit :
        line = data.iloc[i]
        line = line.replace(" ",",")
        line = line.replace(",,,,,",",")
        line = line.replace(",,,,",",")
        line = line.replace(",,,",",")
        line = line.replace(",,",",")
        if i == 0 :
            line = line[0:] #----------------------------------------
------
        coords = coords.append(pd.Series(line))
        i = i + 1
        print("Reading row:", i)
except IndexError : pass
coords = coords.str.split(",", expand = True )
coords = np.array(coords, dtype = float)
x4 = coords[:,0]
y4 = coords[:,1] * 10



##########################################################################
######

title = "b)"
xlabel = "Aminokiselinski ostatak"
ylabel = "RMSF / A"
size = 800
savefig = "../figs/w_rmsf_master.png"
legend_loc = "upper left"
color1 = "greenyellow"
color2 = "limegreen"
color3 = "green"
color4 = "darkolivegreen"

plt.plot(x1,y1, linewidth = 1, color = color1, label = "300 K")
plt.plot(x2,y2, linewidth = 1, color = color2, label = "318 K")
plt.plot(x3,y3, linewidth = 1, color = color3, label = "333 K")
plt.plot(x4,y4, linewidth = 1, color = color4, label = "353 K")

#plt.title(title)
plt.xlabel(xlabel)
plt.ylabel (ylabel)
plt.yticks(np.arange(0, 8, step=0.5))
plt.xticks(np.arange(0, 308, step=20))
ax = plt.subplot(111)
```

```
#ax.set_ylim(0)
ax.set_xlim(0,308)
ax.yaxis.set_minor_locator(MultipleLocator(0.1))
ax.xaxis.set_minor_locator(MultipleLocator(5))

legend = plt.legend(loc= legend_loc, fancybox=True, shadow=True)

plt.savefig(savefig, dpi=size)
#############################################################################
```

Dodatak 8. Glavna korištena skripta za izradu mapa trajektorije na temelju postupka opisanog u poglavlju 3.3. za programski jezik Python.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.ticker import (MultipleLocator)
import timeit
start_all = timeit.default_timer()

preprocess = True
save = True
save_name = "data/w300.csv"
load = False
load_name = "name.csv"

'''-----------------------------------------------------'''

map_type = "first step"  #first step, previous step
path = "./data/"
filename = "w300.pdb"
savefig = "figs/w300"
residues = 308

######--- Importing data from pdb file ----
####################################
file = str(path) + str(filename)
if preprocess == True :
    print ( "Preprocess:", preprocess)
    start_all = timeit.default_timer()
    start_read = timeit.default_timer()

    dataframe = pd.DataFrame(open(file))
    data = pd.Series()

    i = 0
    limit = len(dataframe)
    try:
```

```
        while i <= limit :
            line = dataframe.iloc[i]
            line = line.replace(" ",",",regex=True)
            line = line.replace(",,,,,",",",regex=True)
            line = line.replace(",,,,",",",regex=True)
            line = line.replace(",,,",",",regex=True)
            line = line.replace(",,",",",regex=True)
            data = pd.concat([data, line], axis=0, join="outer")
            i = i + 1
            print("Reading row:", i)
    except IndexError : pass

    data = data.str.split(",", expand = True )
    stop_read = timeit.default_timer()
    print("Reading time: ", (start_read - stop_read) / 60 , "min")

    start_calc = timeit.default_timer()
    graph_data = pd.DataFrame()
    limit = len(data)
    atom = "ATOM"
    end = "END"
    time = 0
    N = 14
    CA = C = 12
    O = 16
    a = 6
    b = 7
    c = 8
    i = 1235
    h = 1

    try:
        while i < limit :
            if atom in data.iloc[i,0]:

                residue = data.iloc[i,5]
                print("Row: ", i ,"; Step: " , time, "; Residue: ",
residue)

                if int(residue) == 213 :
                    x_i2 = float(data.iloc[i,a]) #N
                    y_i2 = float(data.iloc[i,b])
                    z_i2 = float(data.iloc[i,c])

                    x_i1 = float(data.iloc[i+3,a]) #O
                    y_i1 = float(data.iloc[i+3,b])
                    z_i1 = float(data.iloc[i+3,c])

                    x_i3 = float(data.iloc[i+1,a]) #CA
```

```
                    y_i3 = float(data.iloc[i+1,b])
                    z_i3 = float(data.iloc[i+1,c])

                    x_i4 = float(data.iloc[i+2,a]) #C
                    y_i4 = float(data.iloc[i+2,b])
                    z_i4 = float(data.iloc[i+2,c])

                    x_h2 = float(data.iloc[h,a])
                    y_h2 = float(data.iloc[h,b])
                    z_h2 = float(data.iloc[h,c])

                    x_h1 = float(data.iloc[h+3,a])
                    y_h1 = float(data.iloc[h+3,b])
                    z_h1 = float(data.iloc[h+3,c])

                    x_h3 = float(data.iloc[h+1,a])
                    y_h3 = float(data.iloc[h+1,b])
                    z_h3 = float(data.iloc[h+1,c])

                    x_h4 = float(data.iloc[h+2,a])
                    y_h4 = float(data.iloc[h+2,b])
                    z_h4 = float(data.iloc[h+2,c])

                    x_cm_i = (x_i1*O + x_i2*N + x_i3*C + x_i4*C) /
(O+N+CA+C)
                    y_cm_i = (y_i1*O + y_i2*N + y_i3*C + y_i4*C) /
(O+N+CA+C)
                    z_cm_i = (z_i1*O + z_i2*N + z_i3*C + z_i4*C) /
(O+N+CA+C)

                    x_cm_h = (x_h1*O + x_h2*N + x_h3*C + x_h4*C) /
(O+N+CA+C)
                    y_cm_h = (y_h1*O + y_h2*N + y_h3*C + y_h4*C) /
(O+N+CA+C)
                    z_cm_h = (z_h1*O + z_h2*N + z_h3*C + z_h4*C) /
(O+N+CA+C)

                    value =  ( ((x_cm_i - x_cm_h )**2 +
                                (y_cm_i - y_cm_h)**2 +
                                (z_cm_i - z_cm_h)**2 ) )**(1/2)
                    h = h + 4
                    i = i + 4
                elif int(residue) == 308 :
                    x_i2 = float(data.iloc[i,a]) #N
                    y_i2 = float(data.iloc[i,b])
                    z_i2 = float(data.iloc[i,c])

                    x_i1 = float(data.iloc[i+3,a]) #O
                    y_i1 = float(data.iloc[i+3,b])
```

```
                    z_i1 = float(data.iloc[i+3,c])

                    x_i3 = float(data.iloc[i+1,a]) #CA
                    y_i3 = float(data.iloc[i+1,b])
                    z_i3 = float(data.iloc[i+1,c])

                    x_i4 = float(data.iloc[i+2,a]) #C
                    y_i4 = float(data.iloc[i+2,b])
                    z_i4 = float(data.iloc[i+2,c])

                    x_i5 = float(data.iloc[i+4,a]) #O
                    y_i5 = float(data.iloc[i+4,b])
                    z_i5 = float(data.iloc[i+4,c])

                    x_h2 = float(data.iloc[h,a])
                    y_h2 = float(data.iloc[h,b])
                    z_h2 = float(data.iloc[h,c])

                    x_h1 = float(data.iloc[h+3,a])
                    y_h1 = float(data.iloc[h+3,b])
                    z_h1 = float(data.iloc[h+3,c])

                    x_h3 = float(data.iloc[h+1,a])
                    y_h3 = float(data.iloc[h+1,b])
                    z_h3 = float(data.iloc[h+1,c])

                    x_h4 = float(data.iloc[h+2,a])
                    y_h4 = float(data.iloc[h+2,b])
                    z_h4 = float(data.iloc[h+2,c])

                    x_h5 = float(data.iloc[h+4,a]) #O
                    y_h5 = float(data.iloc[h+4,b])
                    z_h5 = float(data.iloc[h+4,c])


                    x_cm_i = (x_i1*O + x_i2*N + x_i3*C + x_i4*C + x_i5*O)
/ (O+N+CA+C+O)
                    y_cm_i = (y_i1*O + y_i2*N + y_i3*C + y_i4*C + y_i5*O)
/ (O+N+CA+C+O)
                    z_cm_i = (z_i1*O + z_i2*N + z_i3*C + z_i4*C + z_i5*O)
/ (O+N+CA+C+O)

                    x_cm_h = (x_h1*O + x_h2*N + x_h3*C + x_h4*C + x_h5*O)
/ (O+N+CA+C+O)
                    y_cm_h = (y_h1*O + y_h2*N + y_h3*C + y_h4*C + y_h5*O)
/ (O+N+CA+C+O)
                    z_cm_h = (z_h1*O + z_h2*N + z_h3*C + z_h4*C + z_h5*O)
/ (O+N+CA+C+O)
```

```
                    value =  ( ((x_cm_i - x_cm_h )**2 +
                               (y_cm_i - y_cm_h)**2 +
                               (z_cm_i - z_cm_h)**2 ) )**(1/2)

                    i = i + 5
                    h = h + 5
                else:
                    x_i2 = float(data.iloc[i,a]) #N
                    y_i2 = float(data.iloc[i,b])
                    z_i2 = float(data.iloc[i,c])

                    x_i1 = float(data.iloc[i+3,a]) #O
                    y_i1 = float(data.iloc[i+3,b])
                    z_i1 = float(data.iloc[i+3,c])

                    x_i3 = float(data.iloc[i+1,a]) #CA
                    y_i3 = float(data.iloc[i+1,b])
                    z_i3 = float(data.iloc[i+1,c])

                    x_i4 = float(data.iloc[i+2,a]) #C
                    y_i4 = float(data.iloc[i+2,b])
                    z_i4 = float(data.iloc[i+2,c])

                    x_h2 = float(data.iloc[h,a])
                    y_h2 = float(data.iloc[h,b])
                    z_h2 = float(data.iloc[h,c])

                    x_h1 = float(data.iloc[h+3,a])
                    y_h1 = float(data.iloc[h+3,b])
                    z_h1 = float(data.iloc[h+3,c])

                    x_h3 = float(data.iloc[h+1,a])
                    y_h3 = float(data.iloc[h+1,b])
                    z_h3 = float(data.iloc[h+1,c])

                    x_h4 = float(data.iloc[h+2,a])
                    y_h4 = float(data.iloc[h+2,b])
                    z_h4 = float(data.iloc[h+2,c])

                    x_cm_i = (x_i1*O + x_i2*N + x_i3*C + x_i4*C) /
(O+N+CA+C)
                    y_cm_i = (y_i1*O + y_i2*N + y_i3*C + y_i4*C) /
(O+N+CA+C)
                    z_cm_i = (z_i1*O + z_i2*N + z_i3*C + z_i4*C) /
(O+N+CA+C)

                    x_cm_h = (x_h1*O + x_h2*N + x_h3*C + x_h4*C) /
(O+N+CA+C)
```

```
                    y_cm_h = (y_h1*O + y_h2*N + y_h3*C + y_h4*C) /
(O+N+CA+C)
                    z_cm_h = (z_h1*O + z_h2*N + z_h3*C + z_h4*C) /
(O+N+CA+C)

                    value =  ( ((x_cm_i - x_cm_h )**2 +
                                (y_cm_i - y_cm_h)**2 +
                                (z_cm_i - z_cm_h)**2 ) )**(1/2)
                    h = h + 4
                    i = i + 4
                coords= pd.DataFrame([residue, time, value])
                coords = coords.transpose()

                graph_data =
pd.concat([graph_data,coords],axis=0,join="outer")

            elif end in str(data.iloc[i,0]) :
                print("---------Finished step ", time, "----------")
                time = time + 1
                i = i + 1
                if map_type == "previous step" :
                    h = h + 1
                elif map_type == "first step" :
                    h = 1
    except IndexError : pass
    stop_calc = timeit.default_timer()
    print(" Calculation time: ", (start_calc - stop_calc) / 60 , "min")

    if save == True:
        graph_data.to_csv(save_name)
        print("Saved graph_data to: ",save_name)
    else:
        print("----did not save graph_data---")
else:
    print("Preprocessing: ",preprocess)
################---data to matrix-----------
##################################

start_trcb = timeit.default_timer()

if load == True:
    print("Loading data: ", load_name)
    graph_data = open(load_name)
    graph_data = pd.Series(graph_data)
    graph_data = graph_data.str.split(",", expand = True)
    graph_data = graph_data.iloc[:,1:]
else:
    print("load data - OFF")
```

```
time = graph_data.iloc[len(graph_data)-10,1]
graph_data = graph_data.set_index(np.arange(0,
                                    len(graph_data), step = 1))
matrix = pd.DataFrame(data=None,
                      index = np.arange(int(time)+1, 0, -1),
                      columns = np.arange(0,int(residues) + 1,1))
k = 0
try:
    while k <= len(graph_data) :
        x = int(graph_data.iloc[k,0])
        y = int(graph_data.iloc[k,1])
        z = float(graph_data.iloc[k,2])
        matrix.iloc[y,x] = z
        print("Transcribing row: ", k)
        k = k + 1
except IndexError: pass

matrix = np.array(matrix.iloc[:,1:], dtype = "float64")
stop_trcb = timeit.default_timer()
stop_all = timeit.default_timer()
print("Transcribing time: ", (start_trcb - stop_trcb) / 60 , "s")
print ("-----------------------------------------")
print("Total time:", (start_all - stop_all) / 60 , "min")

matrix = np.transpose(matrix)

print ("Preprocessing status: ", preprocess)
print("Map type:", map_type )
print ("PDB file used: ", file)
print ("Save status: ", save, "; Save name:", save_name)
print ("Load status: ", load, "; Load name:", load_name)
print("Savefig: ", savefig)
#print ("Figure title: ", title)
#print ("-----Name match check: ", file, save_name, load_name, savefig,
title
#       , "--------")

##################---HEATMAP-------
###############################################

xlabel = "Aminokiselinski ostatak"
ylabel = "t / ns"
custom_legend = True
vmin = 0
vmax = 7
size = 800
cmap = "magma"
aspect = "auto"
title_ = filename
```

```
title = str( str(title_) + " " + str(map_type) + str(" ") +
str("TrajMap") )

ax = plt.subplot(111)
if custom_legend == True :
    print ("Custom legend paramters: ", vmin, "to", vmax)
    plt.imshow(matrix, cmap = cmap, vmin = vmax, vmax = vmin, aspect =
aspect)
else:
    plt.imshow(matrix, cmap = cmap, aspect = aspect)
    print("Default legend parameters")
ax.set_title(title)
ax.invert_yaxis()
plt.colorbar( label = "Pomak / A", fraction=0.029, pad=0.02)
plt.xticks(np.arange(0, 525, step=25), fontsize = 7)
plt.yticks(np.arange(0, 307, step=25), fontsize = 7)
ax.yaxis.set_minor_locator(MultipleLocator(5))
ax.xaxis.set_minor_locator(MultipleLocator(5))
plt.ylabel(xlabel)
plt.xlabel (ylabel)

plt.savefig(savefig, dpi = size, bbox_inches='tight')

#############################################################################
```

Dodatak 9. Korištena skripta za izradu razlikovnih mapa trajektorija na temelju .csv datoteke dobivene skriptom iz dodatka 8, za programski jezik Python.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.ticker import (MultipleLocator)

def Converter (file, residues):
    graph_data = open(file)
    graph_data = pd.Series(graph_data)
    graph_data = graph_data.str.split(",", expand = True)
    graph_data = graph_data.iloc[:,1:]

    time = graph_data.iloc[len(graph_data)-10,1]
    graph_data = graph_data.set_index(np.arange(0,
                                    len(graph_data), step = 1))
    matrix = pd.DataFrame(data=None,
                          index = np.arange(int(time)+1, 0, -1),
                          columns = np.arange(0,int(residues) + 1,1))
    k = 0
    try:
        while k <= len(graph_data) :
```

```
            x = int(graph_data.iloc[k,0])
            y = int(graph_data.iloc[k,1])
            z = float(graph_data.iloc[k,2])
            matrix.iloc[y,x] = z
            print("Transcribing row ", k, file)
            k = k + 1
    except IndexError: pass

    matrix = np.array(matrix.iloc[:,1:], dtype = "float64")
    return(matrix)

#########################################################################
######

hm1 = "../data/w300.csv"
hm2 = "../data/w318.csv"
hm3 ="../data/w333.csv"
hm4 ="../data/w353.csv"
hm5 ="../data/mut/300.csv"
hm6 ="../data/mut/318.csv"
hm7 ="../data/mut/333.csv"
hm8 ="../data/mut/353.csv"
number_of_heatmaps = 8

save = False
save_file = "filename"
#title = "split_nogly - native_nogly"
#savefig = "./figs/DRMSTRS split_nogly - splig_gly"

i = 0
residues = 308
while i < number_of_heatmaps :
    if i == 0 :
        hm_1 = Converter(hm1, residues)
        print ("Loaded m1")
        i = i + 1
    elif i == 1:
        print ("-----------------------------------------")
        hm_2 = Converter(hm2, residues)
        print("Loaded m2")
        i = i + 1
    elif i == 2:
        print ("-----------------------------------------")
        hm_3 = Converter(hm3, residues)
        print("Loaded m3")
        i = i + 1
    elif i == 3:
        print ("-----------------------------------------")
        hm_4 = Converter(hm4, residues)
```

```
        print("Loaded m4")
        i = i + 1
    elif i == 4:
        print ("-----------------------------------------")
        hm_5 = Converter(hm5, residues)
        print("Loaded m5")
        i = i + 1
    elif i == 5:
        print ("-----------------------------------------")
        hm_6 = Converter(hm6, residues)
        print("Loaded m6")
        i = i + 1
    elif i == 6:
        print ("-----------------------------------------")
        hm_7 = Converter(hm7, residues)
        print("Loaded m7")
        i = i + 1
    elif i == 7:
        print ("-----------------------------------------")
        hm_8 = Converter(hm8, residues)
        print("Loaded m8")
        i = i + 1




matrix = (hm_1 + hm_2 + hm_3 + hm_4 - hm_5 - hm_6 - hm_7 - hm_8) / 4
matrix = np.transpose(matrix)

if save == True :
    print ("Saving the matrix to ", save_file )
    pd.DataFrame(matrix).to_csv(save_file)
    print("Finished")

##########################################################################
######

savefig = "../figs/average_w-m"
title_ = "razlike prosjeka po temperaturama divlji - mutirani"
xlabel = "Aminokiselinski ostatak"
ylabel = "t / ns"
size = 800
vmin = -9
vmax = 9
cmap = "seismic"
title = str(str("TrajMap") + " " + str(title_) )

ax = plt.subplot(111)
```

```
plt.imshow(matrix, cmap = cmap, vmin = vmax, vmax = vmin, aspect =
"auto")
ax.set_title(title)

ax.invert_yaxis()
plt.colorbar( label = "Pomak / A", fraction=0.029, pad=0.02)
plt.xticks(np.arange(0, 501, step=25), fontsize = 7)
plt.yticks(np.arange(0, 308, step=25), fontsize = 7)
ax.yaxis.set_minor_locator(MultipleLocator(5))
ax.xaxis.set_minor_locator(MultipleLocator(5))
plt.ylabel(xlabel)
plt.xlabel (ylabel)

plt.savefig(savefig, dpi = size, bbox_inches='tight')

##############################################################################
##############################################################################
```

Dodatak 10. Korištena skripta za izradu grafova pomaka definiranih regija tokom simulacije na temelju .csv datoteke dobivene skriptom iz dodatka 8, za programski jezik Python.

```
# -*- coding: utf-8 -*-
"""
Created on Sat Jan 29 14:07:45 2022

@author: mtjkzc
"""

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.ticker import (MultipleLocator)


file1 = "../data/w300.csv"
file2 = "../data/w318.csv"
file3 = "../data/w333.csv"
file4 = "../data/w353.csv"

title1 = "Prosječan pomak resid 250-260"
savefig1 = "../figs/w_255_shift.png"
size = 800

l1 = "300 K"
l2 = "318 K"
l3 = "333 K"
l4 = "353 K"
```

```
Min = 250
Max = 260

#########################################################################
######

def converter (file) :
    residues = 308
    load = True

    graph_data = open(file)
    graph_data = pd.Series(graph_data)
    graph_data = graph_data.str.split(",", expand = True)
    graph_data = graph_data.iloc[:,1:]

    time = graph_data.iloc[len(graph_data)-10,1]
    graph_data = graph_data.set_index(np.arange(0,
                                    len(graph_data), step = 1))
    matrix = pd.DataFrame(data=None,
                          index = np.arange(int(time)+1, 0, -1),
                          columns = np.arange(0,int(residues) + 1,1))
    k = 0
    try:
        while k <= len(graph_data) :
            x = int(graph_data.iloc[k,0])
            y = int(graph_data.iloc[k,1])
            z = float(graph_data.iloc[k,2])
            matrix.iloc[y,x] = z
            print("Transcribing row: ", k)
            k = k + 1
    except IndexError: pass

    matrix = np.array(matrix.iloc[:,1:], dtype = "float64")
    matrix = np.transpose(matrix)
    return matrix

#########################################################################
######

matrix1 = converter(file1)
matrix2 = converter(file2)
matrix3 = converter(file3)
matrix4 = converter(file4)

ax = plt.subplot(111)

d1 = pd.DataFrame(data = np.arange(0,500,1))
d2 = pd.DataFrame(data = np.arange(0,500,1))
d3 = pd.DataFrame(data = np.arange(0,500,1))
```

```
d4 = pd.DataFrame(data = np.arange(0,500,1))

def avg_gen(matrix, res1, res2, time1, time2, output):
    y = matrix[res1:res2,time1:time2]
    i = 0
    while i < len(output) :
        output.iloc[i] = np.average(y[:,i])
        i = i + 1
    return output


y1 = avg_gen(matrix1, Min, Max, 0, 500, d1)
y2 = avg_gen(matrix2, Min, Max, 0, 500, d2)
y3 = avg_gen(matrix3, Min, Max, 0, 500, d3)
y4 = avg_gen(matrix4, Min, Max, 0, 500, d4)

avg = 15
y1a = y1.rolling(avg).mean()
y2a = y2.rolling(avg).mean()
y3a = y3.rolling(avg).mean()
y4a = y4.rolling(avg).mean()


########################################################################
######
color1 = "greenyellow"
color2 = "limegreen"
color3 = "green"
color4 = "darkolivegreen"

ax = plt.subplot(111)


plt.plot(y1, linewidth=0.3, color=color1, label = l1)
plt.plot(y2, linewidth=0.3, color=color2, label = l2)
plt.plot(y3, linewidth=0.3, color=color3, label = l3)
plt.plot(y4, linewidth=0.3, color=color4, label = l4)

ax.legend(loc='upper left',
          fancybox=True, shadow=True, prop={'size': 7.5})

plt.plot(y1a, linewidth=1, color=color1, label = l1)
plt.plot(y2a, linewidth=1, color=color2, label = l2)
plt.plot(y3a, linewidth=1, color=color3, label = l3)
plt.plot(y4a, linewidth=1, color=color4, label = l4)

ax.set_xlim(0, 500)
ax.set_ylim(0)
plt.yticks(np.arange(0, 8.5, step=0.5))
```

```
plt.xticks(np.arange(0, 550, step=50))
plt.title(title1)
plt.xlabel("t / ns")
plt.ylabel ("Pomak / A")
ax.yaxis.set_minor_locator(MultipleLocator(0.1))
ax.xaxis.set_minor_locator(MultipleLocator(10))

plt.savefig(savefig1, dpi= size)


##############################################################################
```

Dodatak 11. Modularizirana biblioteka mapa trajektorija namijenjena za daljnju oporabu koja omogućava izradu razlikovnih mapi trajektorija prema predlošku danom u dodatku 12. Ova skripta za programski jezik Python je uređena verzija skripti iz dodataka 8 i 9, no u obliku importabilnog lokalnog modula temeljenog na funkcijama.

```
'''
        Trajectory Map v2
        2022.06.11
        Matej Kožić

                        '''
#Currently works only w a single peptide chain

#Reqs: Numpy, Pandas, Matplotlib

##############################################################################

def pdb2csv (filename, savename, residues):

    import numpy as np
    import pandas as pd

    file = filename
    save_name = savename
    residues = residues

    dataframe = pd.DataFrame(open(file))
    data = pd.Series()

    i = 0
    limit = len(dataframe)
    try:
        while i <= limit :
            line = dataframe.iloc[i]
            line = line.replace(" ",",",regex=True)
            line = line.replace(",,,,,",",",regex=True)
```

```
            line = line.replace(",,,,",",",regex=True)
            line = line.replace(",,,",",",regex=True)
            line = line.replace(",,",",",regex=True)
            data = pd.concat([data, line], axis=0, join="outer")
            i = i + 1
            print("Reading row:", i, "of ", limit, "for", filename)
    except IndexError : pass

    data = data.str.split(",", expand = True )

    graph_data = pd.DataFrame()
    limit = len(data)
    atom = "ATOM"
    end = "END"
    time = 0
    N = 14
    CA = C = 12
    O = 16
    a = 6
    b = 7
    c = 8
    i = int(residues) * 4 + 3
    h = 1

    try:
        while i < limit :
            if atom in data.iloc[i,0]:

                residue = data.iloc[i,5]
                print("Calculating: Row: ", i ,"; Timestep: " , time, "; 
Residue: ", residue)

                if int(residue) == residues :

                    x_i1 = float(data.iloc[i,a]) #N
                    y_i1 = float(data.iloc[i,b])
                    z_i1 = float(data.iloc[i,c])

                    x_i2 = float(data.iloc[i+1,a]) #CA
                    y_i2 = float(data.iloc[i+1,b])
                    z_i2 = float(data.iloc[i+1,c])

                    x_i3 = float(data.iloc[i+2,a]) #C
                    y_i3 = float(data.iloc[i+2,b])
                    z_i3 = float(data.iloc[i+2,c])

                    x_i4 = float(data.iloc[i+3,a]) #O
                    y_i4 = float(data.iloc[i+3,b])
                    z_i4 = float(data.iloc[i+3,c])
```

```
                    x_i5 = float(data.iloc[i+4,a]) #O
                    y_i5 = float(data.iloc[i+4,b])
                    z_i5 = float(data.iloc[i+4,c])

                    x_h1 = float(data.iloc[h,a])
                    y_h1 = float(data.iloc[h,b])
                    z_h1 = float(data.iloc[h,c])

                    x_h2 = float(data.iloc[h+1,a])
                    y_h2 = float(data.iloc[h+1,b])
                    z_h2 = float(data.iloc[h+1,c])

                    x_h3 = float(data.iloc[h+2,a])
                    y_h3 = float(data.iloc[h+2,b])
                    z_h3 = float(data.iloc[h+2,c])

                    x_h4 = float(data.iloc[h+3,a])
                    y_h4 = float(data.iloc[h+3,b])
                    z_h4 = float(data.iloc[h+3,c])

                    x_h5 = float(data.iloc[h+4,a])
                    y_h5 = float(data.iloc[h+4,b])
                    z_h5 = float(data.iloc[h+4,c])


                    x_cm_i = (x_i1*N + x_i2*CA + x_i3*C + x_i4*O +
x_i5*O) / (O+N+CA+C+O)
                    y_cm_i = (y_i1*N + y_i2*CA + y_i3*C + y_i4*O +
y_i5*O) / (O+N+CA+C+O)
                    z_cm_i = (z_i1*N + z_i2*CA + z_i3*C + z_i4*O +
z_i5*O) / (O+N+CA+C+O)

                    x_cm_h = (x_h1*N + x_h2*CA + x_h3*C + x_h4*O +
x_h5*O) / (O+N+CA+C+O)
                    y_cm_h = (y_h1*N + y_h2*CA + y_h3*C + y_h4*O +
y_h5*O) / (O+N+CA+C+O)
                    z_cm_h = (z_h1*N + z_h2*CA + z_h3*C + z_h4*O +
z_h5*O) / (O+N+CA+C+O)

                    value =  ( ((x_cm_i - x_cm_h )**2 +
                                (y_cm_i - y_cm_h)**2 +
                                (z_cm_i - z_cm_h)**2 ) )**(1/2)

                    i = i + 5
                    h = h + 5
                else:
                    x_i1 = float(data.iloc[i,a]) #N
                    y_i1 = float(data.iloc[i,b])
```

```
                    z_i1 = float(data.iloc[i,c])

                    x_i2 = float(data.iloc[i+1,a]) #CA
                    y_i2 = float(data.iloc[i+1,b])
                    z_i2 = float(data.iloc[i+1,c])

                    x_i3 = float(data.iloc[i+2,a]) #C
                    y_i3 = float(data.iloc[i+2,b])
                    z_i3 = float(data.iloc[i+2,c])

                    x_i4 = float(data.iloc[i+3,a]) #O
                    y_i4 = float(data.iloc[i+3,b])
                    z_i4 = float(data.iloc[i+3,c])

                    x_h1 = float(data.iloc[h,a])
                    y_h1 = float(data.iloc[h,b])
                    z_h1 = float(data.iloc[h,c])

                    x_h2 = float(data.iloc[h+1,a])
                    y_h2 = float(data.iloc[h+1,b])
                    z_h2 = float(data.iloc[h+1,c])

                    x_h3 = float(data.iloc[h+2,a])
                    y_h3 = float(data.iloc[h+2,b])
                    z_h3 = float(data.iloc[h+2,c])

                    x_h4 = float(data.iloc[h+3,a])
                    y_h4 = float(data.iloc[h+3,b])
                    z_h4 = float(data.iloc[h+3,c])

                    x_cm_i = (x_i1*N + x_i2*CA + x_i3*C + x_i4*O) /
(O+N+CA+C)
                    y_cm_i = (y_i1*N + y_i2*CA + y_i3*C + y_i4*O) /
(O+N+CA+C)
                    z_cm_i = (z_i1*N + z_i2*CA + z_i3*C + z_i4*O) /
(O+N+CA+C)

                    x_cm_h = (x_h1*N + x_h2*CA + x_h3*C + x_h4*O) /
(O+N+CA+C)
                    y_cm_h = (y_h1*N + y_h2*CA + y_h3*C + y_h4*O) /
(O+N+CA+C)
                    z_cm_h = (z_h1*N + z_h2*CA + z_h3*C + z_h4*O) /
(O+N+CA+C)

                    value =  ( ((x_cm_i - x_cm_h )**2 +
                                (y_cm_i - y_cm_h)**2 +
                                (z_cm_i - z_cm_h)**2 ) )**(1/2)
                    h = h + 4
                    i = i + 4
```

```
                coords= pd.DataFrame([residue, time, value])
                coords = coords.transpose()

                graph_data =
pd.concat([graph_data,coords],axis=0,join="outer")

            elif end in str(data.iloc[i,0]) :
                print("---------Finished step ", time, "----------")
                time = time + 1
                i = i + 1
                h = 1

    except IndexError : pass
    graph_data.to_csv(save_name)
    print("Saved coordinates from pdb to: ",save_name)

##########################################################################
######

def csv2matrix (filename, residues):
    import numpy as np
    import pandas as pd
    graph_data = open(filename)
    graph_data = pd.Series(graph_data)
    graph_data = graph_data.str.split(",", expand = True)
    graph_data = graph_data.iloc[:,1:]

    time = graph_data.iloc[len(graph_data)-10,1]
    graph_data = graph_data.set_index(np.arange(0,
                                    len(graph_data), step = 1))
    matrix = pd.DataFrame(data=None,
                          index = np.arange(int(time)+1, 0, -1),
                          columns = np.arange(0,int(residues) + 1,1))
    k = 0
    try:
        while k <= len(graph_data) :
            x = int(graph_data.iloc[k,0])
            y = int(graph_data.iloc[k,1])
            z = float(graph_data.iloc[k,2])
            matrix.iloc[y,x] = z
            print("Transcribing row ", k, filename)
            k = k + 1
    except IndexError: pass

    matrix = np.array(matrix.iloc[:,1:], dtype = "float64")
    matrix = np.transpose(matrix)
    return(matrix)
```

```
##########################################################################
######

# params [x_major, x_minor, y_major, y_minor, vmin, vmax, residues, cmap]

def matrix2map (matrix, savefig, title, params):

    print("Making the TrajMap...")

    x_major = params[0]
    x_minor = params[1]
    y_major = params[2]
    y_minor = params[3]
    vmin = params[4]
    vmax = params[5]
    residues = params[6]
    cmap = params[7]

    if cmap == 0 :
        cmap = "magma"
    elif cmap == 1 :
        cmap = "seismic"

    import numpy as np
    import matplotlib.pyplot as plt
    from matplotlib.ticker import (MultipleLocator)
    xlabel = "Aminokiselinski ostatak"
    ylabel = "Kadar"
    size = 800
    frames = len(matrix[0])
    map_title = str(str("TrajMap") + " " + str(title) )

    ax = plt.subplot(111)
    plt.imshow(matrix, cmap = cmap, vmin = vmin, vmax = vmax, aspect =
"auto")
    ax.set_title(map_title)

    ax.invert_yaxis()
    plt.colorbar( label = "Pomak / A")
    plt.xticks(np.arange(0, frames + 1, step= x_major), fontsize = 7)
    plt.yticks(np.arange(0, residues, step=y_major), fontsize = 7)
    ax.yaxis.set_minor_locator(MultipleLocator(y_minor))
    ax.xaxis.set_minor_locator(MultipleLocator(x_minor))
    plt.ylabel(xlabel)
    plt.xlabel (ylabel)

    plt.savefig(savefig, dpi = size, bbox_inches='tight')
    print("TrajMap saved to:", savefig)
```

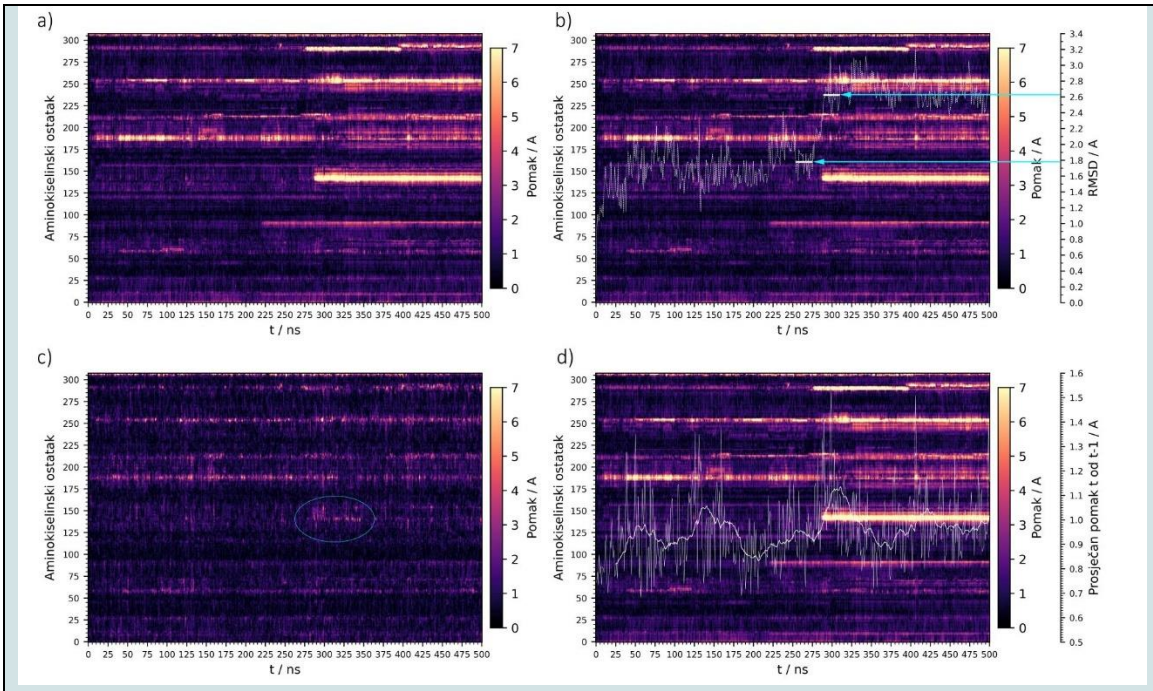
```
##########################################################################
```

Dodatak 13. Predloška za korištenje biblioteke dane dodatkom 12.

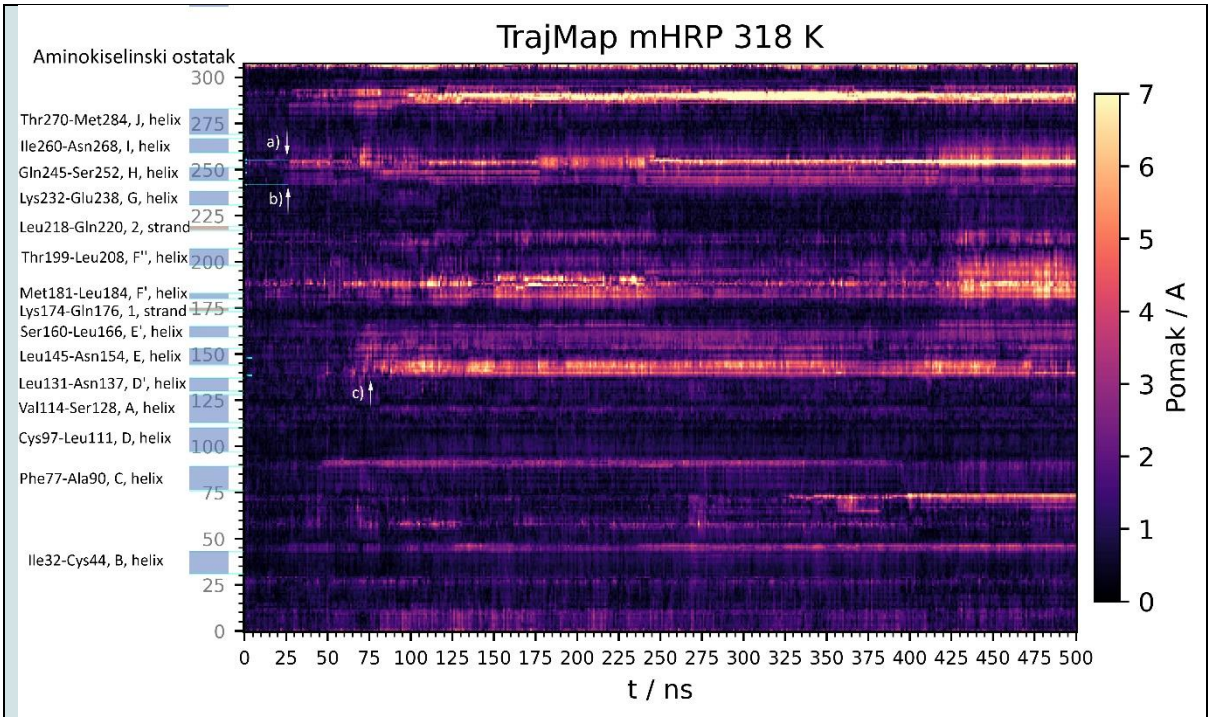
```
'''

Modul ima 3 funkcije: pdb2csv, što kao input uzima aligned koordinate
backbone-a
proteina u pdb formatu. Mora biti aligned inaće ne radi; ja sam to iz VMD
exportao.
Konvezija 50 kadrova za 300ak protein traje minutu dve, a 500 kadrova za
isti traje dva sata. Što više kadrova to bolja rezolucija mape, ali
dulje traje... Tako da za početak najbolje nešto dvoznamenkasto ili nisko
troznamenkasto.
Za početak preporučam 50 kadrova, pa ako radi onda se može rezolucija
povećati.
Ja sam radio sa 500 kadrova za 500ns, i to je super izgledalo. Msm da
više od 500 ne treba.

Druga funkcija csv2matrix kao input ima taj konvertirani csv file, a
output joj je matrica.

Treća funkcija matrix2map kao input ima tu prethodno napravljenu matricu.
"params" parametar uzima array oblika:
    [x_major, x_minor, y_major, y_minor, vmin, vmax, residues, cmap]

    Prvo četvoro su tickovi za osi, vmin i vmax je value za colorbar
heatmap-a.
    residues je broj residua, a cmap je colormap koj može biti 0 ili 1.

    0 je klasična mapa, a 1 je plavo-bijelo-crvena divergentna koja se
koristi
    kada se radi diffmatrix da se pokažu razlike u pomacima npr apo/holo,
gly/nogly
    mut/wld, i sl. Ako se radi klasična mapa, onda ide cmap 0 i vmin 0; a
ako se
    se radi diffmap onda ide cmap 1 i vmin je negativno vmax.

    Array za params sve trebaju biti integeri u točno ovom redosljedu.

Sve te params je najbolje poštelati tako da se samo runna ta linija koda
i oni
se ugađaju.

Konkretno, sve funkcije idu:

    pdb2.csv(filename, savename, residues)
```

```
    csv2matrix(filename, residues)
    matrix2map(matrix, savefig, title, params)

Eto, sretno i valjda će sve raditi. Za napomenuti je kako kod podržava
samo jednolančani peptid sa jednim terminusom. Terminus u sredni ga
ubije.

 Matej Kožić, 2022.06.x
 mtjkzc@gmail.com
 mkozic@chem.pmf.hr

Primjer sa apo i holo strukturama:
'''

import TrajMap as tm

tm.pdb2csv(filename = "apo.pdb", savename = "apo.csv", residues = 245)
tm.pdb2csv(filename = "holo.pdb", savename = "holo.csv", residues = 245)

apo_matrix = tm.csv2matrix(filename = "./data/apo.csv", residues = 245)
holo_matrix = tm.csv2matrix(filename = "./data/holo.csv", residues = 245)

# x_major, x_minor, y_major, y_minor, vmin, vmax, residues, colormap (0
for regular, 1 for diff)
params = [5, 1, 20, 5, 0, 7, 245, 0]
apo = "apo"
tm.matrix2map(matrix = apo_matrix, savefig = apo, title = apo, params =
params )
holo = "holo"
tm.matrix2map(matrix = holo_matrix, savefig = holo, title = holo, params
= params )

diffmatrix = apo_matrix - holo_matrix
title1 = "apo - holo"
params2 = [5, 1, 20, 5, -7, 7, 245, 1]

tm.matrix2map(diffmatrix,title1, title1, params2)
```

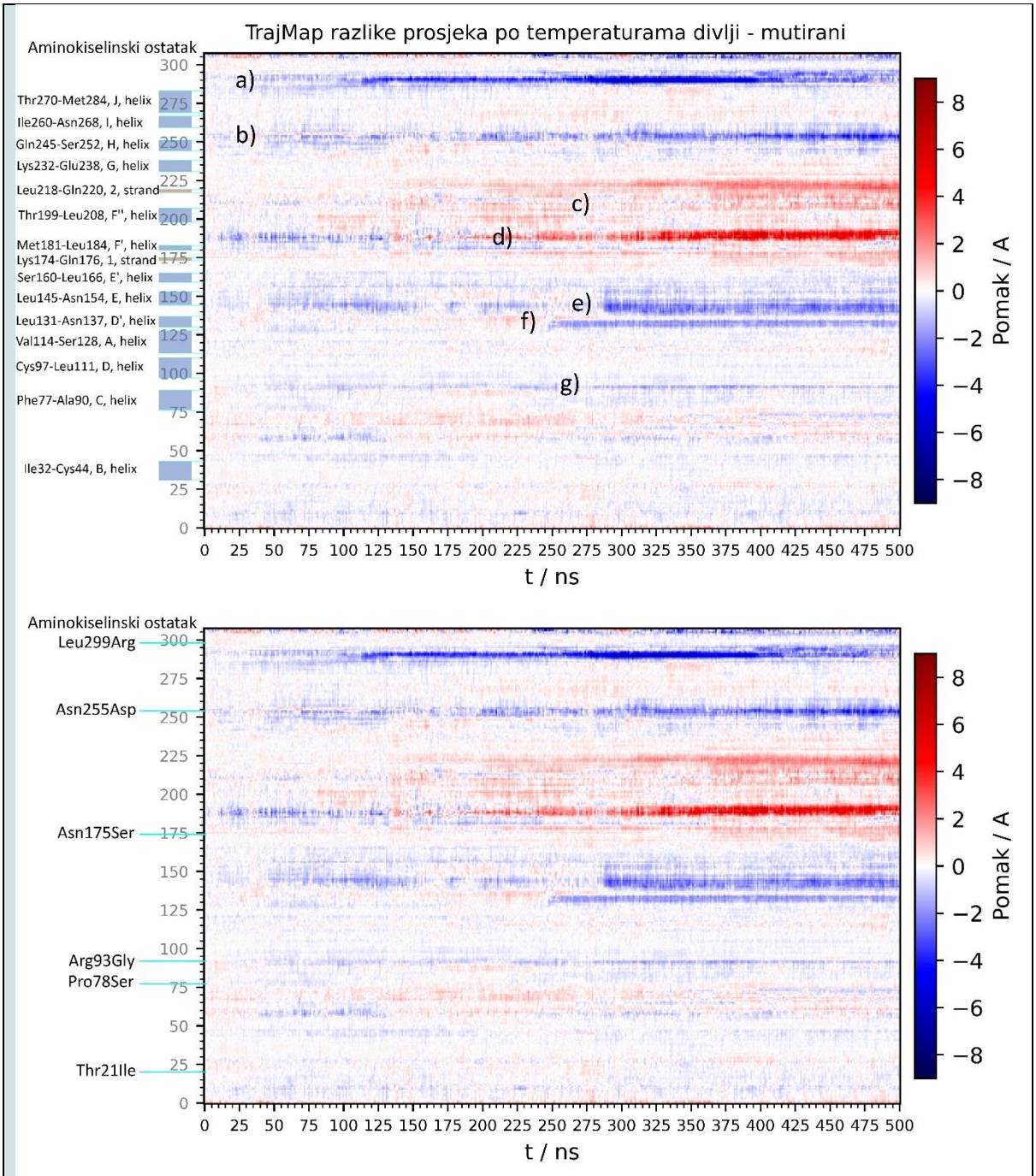
Dodatak 14. Pokazno, različiti tipovi mapa trajektorija i njihove mogućnosti uporabe.



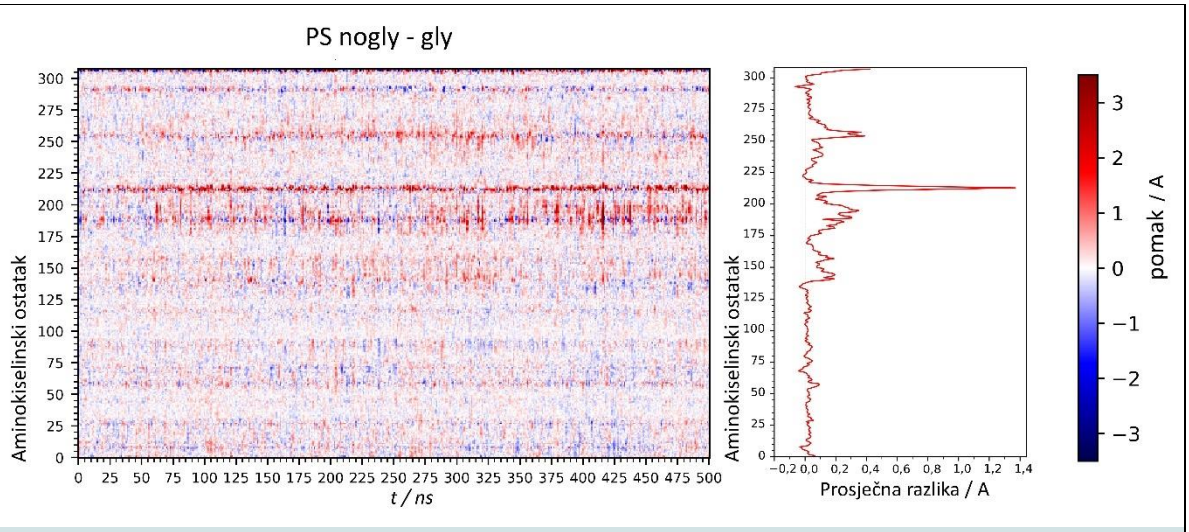
Slika D1. Pokazni set mapa trajektorija: a) standardna FS mapa gdje je vidljivo vrijeme, lokacija, i magnituda svakog pomaka okosnice proteina od izvornog položaja u kadru nula. b) FS mapa trajektorije sa superponiranom RMSD krivuljom na kojoj su označene regije prije i poslije značajnog pomaka. c) PS mapa trajektorije gdje su vidljivi pomaci iz prethodnog u aktualni kadar. Zaokružena je regija intenzivnih pomaka koja odgovara periodu nestabilnosti zavojnice uslijed mijenjanja konformacije. d) FS mapa trajektorije sa superponiranom krivuljom prosječnih pomaka iz prethodnog kadra, dobivenom uprosječavanjem PS prikaza kroz *y* os, sa pokretnim prosjekom od 15 kadrova.



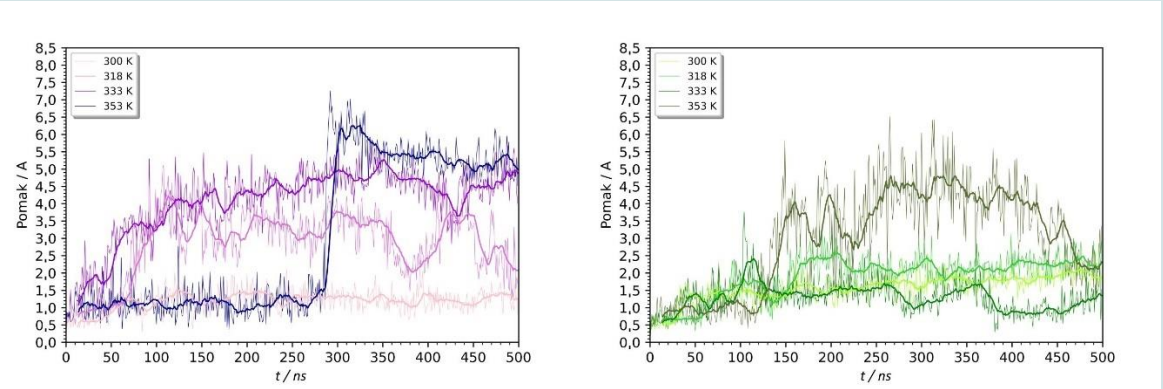
Slika D2. Prikazivanje redoslijeda promjena tijekom simulacije.



Slika D3. Isticanje razlika između simulacija (ili prosjeka simulacija) na prikazu sa označenim sekundarnim strukturama i aminokiselinama od interesa.



Slika D4. Isticanje razlika u simulacijama od interesa u vidu pomaka iz kadra *i* u kadar *i+1*, te prikazivanje prosjeka razlika kretanja aminokiselinskih ostataka tijekom simulacija.



Slika D5. Prikazivanje pomicanja regija od interesa tijekom simulacija.

11. Životopis

Rođen na Badnjak 1998. godine od majke kemičarke, Mateja je od malena zanimala priroda te je odrastao sa naklonostima i predispozicijama za prirodne znanosti. Vođen tim interesima upisuje srednju školu Vladimira Preloga u Zagrebu na smjeru Ekološki tehničar, i 2017. godine dobiva nagradu Faust Vrančić za osvojeno prvo mjesto na državnom natjecanju ekoloških tehničara. Iste godine završava srednju školu, te upisuje preddiplomski studij kemije na Prirodoslovnom Matematičkom Fakultetu u Zagrebu. Godine 2020. završava preddiplomski studij i stiče titulu prvostupnika, i te iste godine upisuje diplomski studij kemije na zavodima za fizikalnu kemiju i biokemiju, gdje se trenutno nalazi.

Mateja najviše zanima spoj fizikalne kemije i biokemije u proučavanju bioorganomašinerije najmodernijim računalnim metodama. Pozadina u kemiji omogućava mu sagledavanje problema iz biofizike i molekularne biologije kemijskom intuicijom, a znanje programiranja i veliki interes u podatkovnoj znanosti omogućava mu osmišljavanje kreativnih načina za rješavanje različitih problema. Upravo tu interdisciplinarnost Matej smatra svojim najjačim atributom. Planira ostati u znanosti, te nakon diplomiranja krenuti u smjeru stjecanja doktorata; i nada se raditi na području računalne biofizike. Konkretno, najviše ga privlači razvoj i usavršavanje metoda rukovanja bioinformatički i kemijski relevantnim podatcima: prevođenje apstraktnih teorijskih koncepata iz matematike, fizike, i kemije u alate koji vode do konkretnih eksperimentalnih rezultata.