Decision Transformer for Plate Heat Exchanger control

Jay Bojič Burgos, Matevž Pustišek, Igor Škrjanc

Univerza v Ljubljani, Fakulteta za Elektrotehniko
E-pošta: jb84908@student.uni-lj.si

Abstract

Plate Heat Exchangers (PHEs) are crucial in various industrial processes, yet their nonlinear nature poses significant control challenges. This paper explores the use of Decision Transformers (DTs) for PHE control, leveraging the ability of transformers to learn complex patterns and make decisions based on long sequences of data. Our research aims to evaluate the effectiveness and feasibility of DTs in PHE control. We developed a simulation environment to generate extensive training data and test control strategies. The DT architecture was implemented and adapted to the specific requirements of PHE control, with performance evaluated based on the accuracy and efficiency of the DT. Our findings indicate that DTs can maintain desired temperature ranges under varying operational conditions, achieving stability for temperature setpoint changes in under 100 timesteps and in under 150 timesteps for changes in cold water valve openings. However, we note that challenges such as the need for large quality datasets remain. Additionally, we touched upon how additional work can further improve the efficiency in control.

1 Uvod

Ploščni toplotni izmenjevalniki (PHE) so ključne komponente v številnih industrijskih procesih, od prehrambene industrije do energetskih sistemov. Kljub njihovi široki uporabi pa učinkovito krmiljenje PHE ostaja zahtevna naloga zaradi njihove nelinearne narave [1]. Tradicionalne metode krmiljenja PHE, kot so PID regulacija, modelno prediktivno krmiljenje (MPC), nelinearni kontrolni sistemi itd., temeljijo na natančnih matematičnih modelih sistema. Vendar pa te metode pogosto zahtevajo kompleksno nastavljanje in se soočajo z izzivi pri obvladovanju nelinearnosti ter dinamičnih sprememb v sistemu [2, 3].

V tem kontekstu se pojavi potreba po pristopu, ki bi omogočil učinkovito krmiljenje brez potrebe po natančnem modeliranju, tudi za druge industrijske sisteme. V tem članku predstavljamo pristop k krmiljenju PHE z uporabo odločitvenih transformerjev (ang. decision transformer, DT). DT je prilagoditev arhitekture transformerjev za naloge spodbujevalnega učenja, ki omogoča učenje kontrolnih strategij neposredno iz podatkov. Namesto, da bi ustvarili specifičen model za krmiljenje določenega PHE, lahko preprosto ustvarimo podatke, iz katerih se DT uči krmiljenja.

Ena izmed ključnih prednosti DT je njegova sposobnost učenja iz preteklih izkušenj, tako dobrih kot slabih, na podlagi nagrad. Ta pristop omogoča DT, da se prilagaja in izboljšuje svoje strategije na podlagi pridobljenih podatkov.

Glavni cilj te raziskave je bil preveriti učinkovitost in izvedljivost uporabe DT za krmiljenje PHE. Specifično smo si zastavili naslednje cilje: razviti simulacijsko okolje PHE, ki bo omogočalo generiranje obsežnih učnih podatkov in testiranje krmilnih strategij; implementirati in prilagoditi arhitekturo DT za specifične zahteve krmiljenja PHE; oceniti pravilnost delovanja DT. Prav tako smo analizirali prednosti in omejitve pristopa DT v kontekstu industrijskih procesov.

Medtem ko so bile do sedaj za nadzor PHE uporabljene različne metode, kot so klasične metode (npr. PID krmilniki) [4, 5] in inteligentne metode (npr. nevronske mreže, genetski algoritmi) [6, 7], smo pri naši raziskavi uporabili napredne metode strojnega učenja, specifično Decision Transformerje (DT).

Doprinosi naše raziskave so naslednji:

1. Raziskava učinkovitosti DT, specifično pri nadzoru PHE, prispeva k razumevanju zmogljivosti te arhitekture v kontekstu industrijske kontrole.

2. Prikaz hitrega (20 minut) in učinkovitega učenja na potrošniški opremi (Macbook M1 pro) kljub minimalni pripravi podatkovnega nabora nakazuje na potencial uporabe DT v okoljih z omejeno računalniško opremo.

3. Razvoj integriranega testnega okolja, ki združuje naučeni DT model z Matlab simulacijo v realnem času, omogoča ne le testiranje, temveč tudi odpira možnost, da z definicijo novega sistema nagrajevanja omogočamo nadaljnjo generacijo učnih podatkov, s katerim bi lahko dodatno zboljšali delovanje.

Z našim raziskovalnim delom ne le, da prikazujemo učinkovito uporabo Decision Transformerjev v nadzoru PHE, temveč tudi postavljamo temelje za nadaljnjo raziskavo uporabe te arhitekture v drugih industrijskih procesih, kjer so prisotne kompleksne nelinearnosti in potrebe po obvladovanju dolgih zaporedij podatkov.

2 Teorija transformerjev

Transformerji [9] so nova arhitektura nevronskih mrež, ki je bila prvotno zasnovana za obdelavo naravnega jezika, vendar so se zaradi svoje učinkovitosti in prilagodljivosti razširili na številna druga področja.

Ključne komponente arhitekture transformerjev:

1. Mehanizem pozornosti (ang. Attention Mechanism): Osrednji del transformerjev, ki omogoča modelu, da se osredotoči na različne dele vhodnega zaporedja pri ustvarjanju izhoda. To je doseženo z izračunom uteži pomembnosti med vsemi elementi zaporedja.

2. Večglava pozornost (ang. Multi-Head Attention): Ta nadgradnja osnovnega mehanizma pozornosti omogoča modelu, da hkrati upošteva različne vidike informacij iz vhodnega zaporedja, kar poveča zmogljivost učenja kompleksnih vzorcev.

3. Posredovalne nevronske mreže (ang. Feed-Forward Networks): Po vsakem sloju pozornosti sledi plast posredovalne nevronske mreže, ki dodatno obdela informacije in poveča nelinearno zmogljivost modela.

4. Pozicijsko kodiranje (ang. Positional Encoding): Ker transformerji obdelujejo vhodne podatke vzporedno in ne zaporedno, je potrebno dodati informacijo o položaju vsakega elementa v zaporedju podatkov. To je doseženo s pozicijskim kodiranjem.

Transformerji obdelujejo zaporedne podatke drugače kot tradicionalni modeli, kot so rekurentne nevronske mreže (RNN). Namesto zaporedne obdelave, lahko transformerji vzporedno obdelujejo celotno zaporedje, kar omogoča hitrejše učenje in boljše zajemanje odvisnosti na dolge razdalje v podatkih. To vzporedno procesiranje oziroma paralelizacija tudi omogoča učinkovitejše in hitrejše učenje na velikih podatkovnih naborih, posebej z uporabo grafičnih karticah (GPU).

Zaradi teh prednosti imajo tudi večjo prilagodljivost in možnost prenosa na različne domene, od procesiranja slike, zvoka, in naravnega jezika. Vse kar je potrebno narediti je nabor podatkov spraviti v sekvenco podatkov ter dodati pozicijsko enkodiranje.

Primer druge uporabe je Decision Transformerj (DT) [8], ki je prilagoditev arhitekture transformerjev za naloge spodbujevalnega učenja (ang. reinforcement learning). Ključna inovacija DT je preoblikovanje problema spodbujevalnega učenja v problem napovedovanja zaporedja. DT se uči na podlagi preteklih trajektorij, ki vključujejo stanja, akcije in nagrade, ter poskuša napovedati optimalne akcije za doseganje določenega cilja.

DT deluje na podlagi vhodnih podatkov, ki vključujejo stanja, akcije, nagrade in ciljni donos (ang. return-to-go, RTG). Model se uči napovedovati akcije glede na pretekla stanja in ciljni donos, pri čemer lahko med izvajanjem prilagajamo ciljni donos in s tem vplivamo na vedenje modela. Ta pristop se razlikuje od klasičnih metod spodbujevalnega učenja. DT ne zahteva eksplicitnega učenja funkcije vrednosti ali politike, temveč je proces učenja bolj podoben nadzorovanemu učenju, kar lahko poenostavi optimizacijo. Poleg tega lahko DT izkoristi prednosti prenosa znanja iz prednaučenih jezikovnih modelov, kar lahko potencialno dodatno izboljša njegovo učinkovitost.

Ključna inovacija DT je torej uporaba ciljne vrednosti nagrade (ang. Return-to-go, RTG) kot dodatnega vhodnega podatka. RTG vrednost je seštevek vseh nagrad, ki jih pričakujemo do konca epizode. Uporaba RTG modelu omogoča, da svoje akcije prilagaja glede na želeni končni rezultat. To je posebej koristno pri upravljanju sistemov, kot je toplotni izmenjevalnik, kjer lahko model upošteva dolgoročne posledice akcij in optimizira strategijo upravljanja za daljše časovno obdobje. Za boljše razumevanje koncepta RTG si lahko predstavljamo uporabo natreniranega DT modela v računalniških igrah. Pri podajanju RTG za določeno epizodo majhna vrednost povzroči, da agent deluje kot začetnik, velika vrednost pa kot zelo izkušen igralec. To pomeni, da se model lahko nauči tako iz slabih kot tudi dobrih podatkov in akcij. Uporaba RTG prinaša tako prednosti kot slabosti. Glavna prednost je fleksibilnost modela, ki se lahko prilagaja različnim ciljem. Slabost pa je, da ni vedno jasno, kolikšna je optimalna RTG vrednost za določeno okolje ali sistem. Raziskave kažejo, da uporaba arbitrarno velike RTG vrednosti ne zagotavlja nujno boljših rezultatov. V mnogih primerih lahko prevelika RTG vrednost povzroči enako ali celo slabše vedenje, odvisno od specifičnosti sistema, ki ga upravljamo.

Uporaba DT v krmiljenju sistemov, kot je ploščni toplotni izmenjevalnik (PHE), ponuja več potencialnih prednosti. Model je sposoben učenja kompleksnih vzorcev iz dolgih zaporedij podatkov in se lahko prilagaja različnim pogojem delovanja brez potrebe po eksplicitnem modeliranju sistema. Poleg tega omogoča vključevanje dolgotrajnih odvisnosti v strategijo krmiljenja, kar je lahko ključno za optimalno delovanje v dinamičnih okoljih.

Kljub tem obetavnim lastnostim pa uporaba DT v industrijskih aplikacijah prinaša tudi določene izzive. Eden glavnih je potreba po veliki količini kakovostnih učnih podatkov, kar lahko predstavlja oviro v okoljih, kjer so takšni podatki težko dostopni ali dragi za pridobivanje. Dodatno, kompleksnost modela oteži interpretacijo in analizo njegovega delovanja, kar je lahko problematično v industrijskem okolju, kjer je razumevanje in zaupanje v krmilni sistem ključnega pomena.

V nadaljevanju bomo podrobneje predstavili našo implementacijo DT za krmiljenje PHE ter si pogledali njegovo delovanje. S tem bomo poskušali ovrednotiti potencial te tehnologije za praktično uporabo v industrijskih sistemih in identificirati morebitne omejitve ali področja za nadaljnje izboljšave.

3 Implementacija

Cilj je poskus uporabe DT modela, s katerim bi lahko upravljali toplotni izmenjevalec pri raznih spremembah (odpiranje ventila hladne vode) in ciljnih temperaturah. V sistemu torej upravljamo električni tok, na strani vroče vode, kjer večanje toka pomeni večanja vodnega pretoka v vroči vodi zanke sistema. Za doseganje cilja je bilo

potrebno razviti več delov. To je vključevalo:

Razvoj simulacije, kjer smo modificirali obstoječo Matlab skripto simulacije PHE z namenom generiranja obsežnega nabora podatkov. Podatki so vključevali naslednja stanja: temperature vroče in hladne vode, akcije (spremembe električnega toka) ter nagrade za pravilnost akcij, pri vsakem časovnem koraku. Podatke je bilo potrebno korektno očistiti, razdeliti na učno in validacijsko množico podatkov ter formatirati v pravilno obliko za vnos/uporabo pri učenju DT modela. Sledila je implementacija modela kjer so smo razvili DT model s pomočjo PyTorch knjižnice, vključno z definicijo arhitekture mreže in parametrov učenja.

Pri učenju smo za oceno napredka beležili validacijsko izgubo (ang. validation loss) in učno izgubo (ang. training loss) za vsako epoho. Za kasnejšo uporabo smo uteži modela shranjevali pri vsaki epohi ter na koncu obdržali samo najboljši verziji, ki smo ju izbrali glede na vrednost validacijske izgube. Lahko bi shranjevali vsako verzijo pri vsaki epohi, vendar zaradi omejitev z računalniškim pomnilnikom tega nismo storili. Za učinkovito testiranje smo namesto uporabe testne množice podatkov raje razvili integrirano testno okolje, kjer Python (verzija 3.9.6) upravlja celotno stanje sistema, Matlab (verzija R2024a) pa izvaja simulacijo sprememb stanja. Za učinkovito komunikacijo med Python kodo in Matlab simulacijo smo uporabili knjižnico Matlab Engine API for Python (verzija 24.1.2).

4 Natančnejša razlaga komponent

4.1 Simulacija PHE

Simulacija PHE je bila napisana v Matlabu, kjer smo jo modificirali za dva namena.

4.1.1 Generiranje učnih podatkov

V Matlabu smo napisali skripto, ki krmili simulacijo z navadnim PID krmilnikom. Cilj skripte je bila le generacija podatkov (stanj) - temperatur, električnega toka, akcij (tj. sprememba toka) in nagrad. Slednje so bile izračunane na podlagi temperaturne ciljne vrednosti in dejanskih temperatur po vsaki akciji. Ker se DT lahko uči tako iz slabih kot dobrih podatkov/izkušenj, PID krmilnika nismo optimizirali. Potrebno je bilo le oceniti akcije krmilnika, bodisi ali so bile akcije dobre (pozitivna ocene) ali slabe (negativna ocena).

Za učenje našega DT modela smo generirali 3000 simulacij, vsaka s 10000 časovnimi koraki. To število simulacij smo določili na podlagi kompromisa med reprezentativnostjo podatkov in računsko zahtevnostjo. S 3000 simulacijami smo zajeli širok spekter možnih scenarijev delovanja PHE, vključno z različnimi začetnimi pogoji, spremembami ciljne temperature in odpiranji ventila hladne vode. Preliminarni eksperimenti so pokazali, da je to število simulacij zadostno za doseganje dobrih rezultatov, medtem ko bi bistveno večje število simulacij nesorazmerno povečalo čas učenja brez znatnega izboljšanja zmogljivosti modela.

Vse začetne parametre stanj, ciljnih temperatur (med $10°\text{C}$ in $35°\text{C}$) in premikov (odpiranje ventila hladne vode) v parametrih sistema smo naključno generirali za vsako epizodo simuliranja. Zaradi potreb DT modela po nagradah opravljenih akcij smo definirali sistem nagrajevanja.

Nagrado $r = 10$ smo postavili, če je trenutna temperatura T znotraj dovoljenega intervala, definiranega kot $T_{ciljna} \pm 1{,}5°\text{C}$. Če temperatura hladne vode T ni znotraj tega intervala, smo uvedli skalirano nagrado r, ki je izračunana po enačbi (1). Cilj nagrajevanja je bil nagraditi premikanje temperature proti ciljni temperaturi T_{target} in kaznovati, ko se je temperatura gibala stran od ciljne vrednosti.

Dodatno smo uvedli negativno nagrado $r = -20$, kot je prikazano v enačbi (2), v primeru, ko se je temperatura T oddaljila za več kot dvakratnik dovoljenega praga/intervala P. Cilj te nagrade je preprečiti preveliko prekoračenje temperature (ang. overshoot).

$$r = \frac{\Delta T_{t-1} - \Delta T_t}{\Delta T_{t-1}} \tag{1}$$

$$|\Delta T| > 2 \times P \tag{2}$$

4.1.2 Testna funkcija

Namen testne funkcije je realno testiranje natreniranega DT modela brez uporabe testne množice podatkov. Funkciji za vsak časovni korak podamo vhodne podatke in kot rezultat dobimo izhodne podatke, ki jih nato uporabimo za testiranje DT modela. Izhodni padatek je naslednje stanje, katerega DT uporabi za napovedovanje naslednje akcije. Prav tako dobimo izračunano nagrado za vsak časovni korak, ki jo potrebujemo na strani DT pri izračunu RTG.

- Vhodni podatki: Vključujejo napovedano akcijo, trenutno in prejšnje stanje (temperaturi, tok), nov časovni korak, časovne točke za premike parametrov ter samo vrednost sistemskega parametra za premike (odpiranje ventila hladne vode).
- Izhodni podatki: Nova stanja po izvedbi akcije in skalarna nagrada glede na uspešnost regulacije temperature.

4.2 DT transformer

Vse v zvezi z implementacijo regulatorja, v našem primeru DT, je bilo opravljeno v Python-u. Implementirati je bilo potrebno sam DT ter oblikovati in razdeliti podatke za učenje. DT je bil implementiran z uporabo knjižnice PyTorch in Hugging Face Transformers. Prav tako smo postavili testno okolje, kjer smo lahko testirali implemetiran DT model na dejanski simulaciji toplotnega izmenjevalca, ki je bil implementiran v Matlab-u.

Arhitektura našega DT modela temelji na implementaciji, prilagojeni za specifične zahteve krmiljenja PHE. Model ima naslednje ključne parametre:

- Dimenzija stanja (state_dim): 3 (temperatura hladne vode, električni tok, ciljna temperatura)
- Dimenzija akcije (act_dim): 1 (sprememba električnega toka)
- Velikost skritega sloja (hidden_size): 128

- Število slojev (n_layer): 3
- Število glav pozornosti (n_head): 1
- Aktivacijska funkcija: ReLU

Te parametre smo izbrali na podlagi eksperimentiranja in upoštevanja kompleksnosti našega problema. Manjše število slojev in glav pozornosti v primerjavi z običajnimi jezikovnimi modeli odraža relativno preprostost našega problema v primerjavi s procesiranjem naravnega jezika, hkrati pa zagotavlja zadostno zmogljivost za učenje kompleksnih vzorcev v podatkih PHE.

4.2.1 Priprava podatkov in učenje

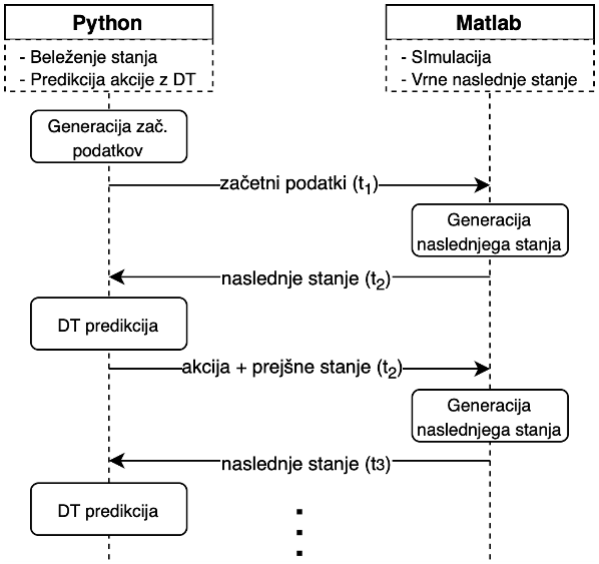
Priprava podatkov vključuje branje in obdelavo podatkov iz parquet datoteke, generirane v Matlab-u ter razdelitev podatkov na učno in validacijsko množico. Implementirali smo zbiralnik podatkov za ustrezno oblikovanje podatkov za učenje. Izvedli smo tudi normalizacijo podatkov, ki je ključnega pomena za učinkovito učenje DT modela. S preslikavo vhodnih podatkov na standardno skalo zagotavljamo, da imajo vse značilnosti primerljiv vpliv na model, ne glede na njihove izvirne razpone vrednosti. To je še posebej pomembno pri obravnavi različnih fizikalnih veličin, kot so temperatura in električni tok, ki imajo lahko zelo različne skale. V našem primeru smo uporabili standardizacijo, kjer smo od vsake vrednosti odšteli povprečje in rezultat delili s standardnim odklonom, kar je zagotovilo, da imajo vsi vhodni podatki povprečje 0 in standardni odklon 1.

4.2.2 Testiranje modela

Testiranje modela vključuje implementacijo funkcije za napovedovanje akcij ter integracijo z Matlab simulacijo preko Python-Matlab vmesnika (Matlab Engine API for Python).

Namesto testne množice podatkov smo se odločili za razvoj integriranega testnega okolja iz več razlogov. Prvič, integrirano testno okolje nam omogoča oceno zmogljivosti modela v realnem času in v različnih scenarijih, ki morda niso bili zajeti v prvotnem naboru podatkov. Drugič, ta pristop omogoča testiranje modela v zaprtozančnem načinu, kjer lahko opazujemo, kako se model odziva na lastne akcije skozi čas. Tretjič, ta pristop bolje posnema realno implementacijo sistema, kar nam daje boljši vpogled v praktično uporabnost našega DT modela za krmiljenje PHE.

Slika 1 prikazuje sekvenčni diagram delovanja kode v fazi testiranja. Vidimo, kako na strani Pythona skrbimo za generacijo začetnih pogojev (časovni korak 1), beleženje stanj in za napovedovanje naslednjega ukaza (ang. action). Na Matlab strani imamo simulacijo, s katero generiramo naslednje stanje glede na prejšnja stanja in akcije.



Slika 1: Sekvenčni diagram delovanja kode

5 Rezultati

Parametri učenja so bili naslednji:

Model DT je bil treniran z naslednjimi ključnimi parametri:

- Število epoh: 4
- Velikost serije (batch size): 32
- Hitrost učenja (learning rate): 1×10^{-4}
- Upad uteži (weight decay): 1×10^{-4}
- Optimizacijski algoritem: AdamW

Učenje je potekalo na procesorju (CPU) zaradi omejitev strojne opreme. Uporabljen je bil prenosnik Macbook M1 pro, čas učenja pa je trajal 20 minut. Med učenjem smo spremljali izgubo na učni in validacijski množici. Zaradi omejitve shranjevanja modelov smo ohranili le najboljši model glede na validacijsko izgubo. Po zaključenem učenju smo model testirali na dejanski Matlab simulaciji. Čas izvršbe upravljanja izmenjevalca z DT za 5000 časovnih korakov je znašal 36s.

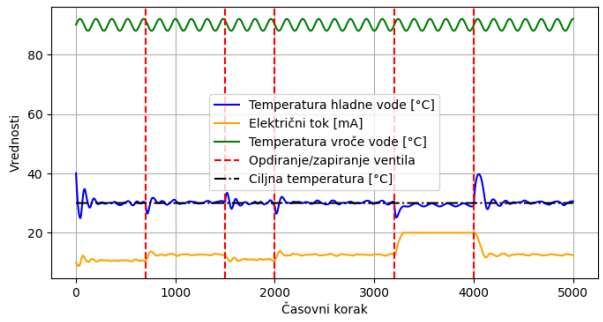
5.1 Upravljanje pri konstantni ciljni temperaturi

Predstavljamo dva testna primera, kjer imamo nastavljeno samo eno ciljno temperaturo.

Na sliki 2 je prikazan primer s petimi spremembami ventila hladne vode. Vidimo, da kljub pogostejšim spremembam model večino časa uspešno vzdržuje temperaturo blizu 30°C. Parameter $M.F_p$ (rdeče črtkane črte) prikazuje, kje se spremembe zgodijo. Pri vrednosti parametra $M.F_p$ = 0,55 je ventil zaprt, pri vrednosti 0,85 pa odprt. Vidimo, da pri velikih spremembah parametra pride do večje spremembe temperature, vendar se ta stabilizira v največ 150 časovnih korakih.

Spremembe sistema so naslednje:

- Časovni korak 700: $M.F_p = 0{,}65$
- Časovni korak 1500: $M.F_p = 0{,}55$
- Časovni korak 2000: $M.F_p = 0{,}65$
- Časovni korak 3200: $M.F_p = 0{,}85$
- Časovni korak 4000: $M.F_p = 0{,}65$



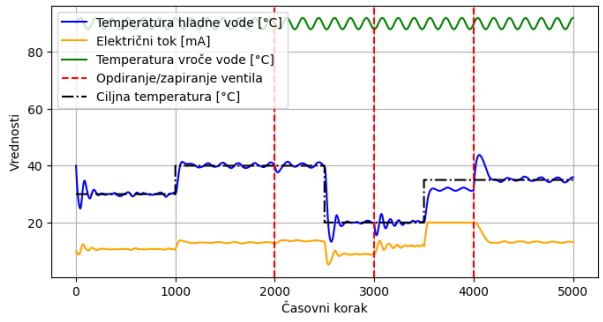
Slika 2: Scenarij upravljanja s petimi spremembami ventila

5.2 Upravljanje pri dinamičnih ciljnih temperaturah

Slika 3 prikazuje scenarij, kjer imamo štiri spremembe ciljne temperature in tri spremembe $M.F_p$. Tudi tu se model dobro odziva, je pa vidno, da je odziv boljši pri spremembah temperature, kot pri velikih spremembah odprtosti ventila ($M.F_p$). Iz slike je prav tako razvidno, da med časovnim korakom 3500 in 4000 sistem doseže limit, kjer je električni tok v maksimalni vrednosti, vendar ciljne vrednosti ni možno doseči. Razlog za to je odprt ventil hladne vode. Ko tega zapremo, pride do velike spremembe, ki se stabilizira v manj kot 150 časovnih korakih.

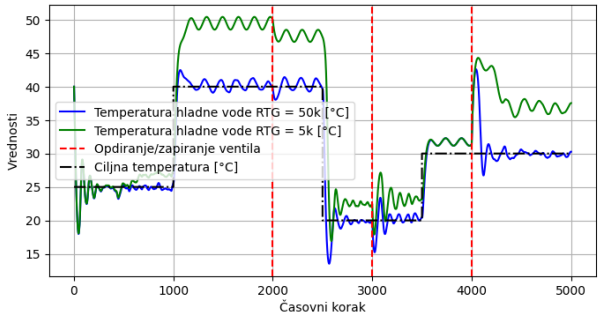
Spremembe sistema:

- Časovni korak 0-1000: Ciljna temperatura = 25°C
- Časovni korak 1000: Ciljna temperatura = 40°C
- Časovni korak 2000: $M.F_p = 0{,}55$
- Časovni korak 2500: Ciljna temperatura = 20°C
- Časovni korak 3000: $M.F_p = 0{,}80$
- Časovni korak 3500: Ciljna temperatura = 30°C
- Časovni korak 4000: $M.F_p = 0{,}60$



Slika 3: Scenarij upravljanja PHE s tremi spremembami ventila in štirimi ciljnimi temperaturami

Slika 4 prikazuje delovanje modela pri različnih vrednostih RTG. Vidno je, da z zmanjševanjem vrednosti RTG model slabše deluje. To je smiselno, saj RTG predstavlja seštevek vseh nagrad, ki naj bi jih model želel pridobiti. Če je ta vrednost manjša, bodo narejene suboptimalne akcije. Z večanjem vrednosti RTG se v našem primeru učinkovitost ni vidno spremenila, zato tega nismo prikazali.



Slika 4: Učinkovitost modela pri različnih RTG vrednostih

6 Zaključek

V primerjavi z obstoječo literaturo, uporaba Decision Transformerjev (DT) za nadzor ploščnih toplotnih izmenjevalnikov (PHE) predstavlja novost v področju industrijske kontrole. Prejšnje metode, kot so PID, MPC ter metode strojnega učenja, zahtevajo natančno modeliranje sistema ali kompleksne algoritme, medtem ko DT omogoča učenje neposredno iz podatkov in sprejema odločitve na podlagi preteklih izkušenj. Integrirano testno okolje in izboljšanje sistema za nagrajevanje akcij omogoča nadaljnjo generacijo učnih podatkov, kar bi lahko še dodatno izboljšalo učinkovitost upravljanja v realnih industrijskih okoljih.

Rezultati kažejo, da model ohranja temperaturo hladne vode v območju ±1,5°C od ciljne vrednosti, kljub znatnim spremembam v sistemu. Pri spremembah odprtosti ventila (M.Fp med 0,55 in 0,85) model stabilizira temperaturo v povprečju v manj kot 150 časovnih korakih. Pri nenadnih spremembah ciljne temperature za do 20°C model doseže novo ciljno območje v manj kot 100 časovnih korakih, z minimalnim preseganjem cilja. DT tako predstavlja zanimiv pristop za vodenje procesov. Še posebej zanimiva sposobnost je upoštevanje večjega števila stanj iz zgodovine, ki potencialno omogoča bolj sofisticirano strategijo upravljanja v primerjavi z drugimi RL metodami. Celoten RL problem je tako postavljen kot problem sekvenc stanj, kjer je naloga napovedovanje naslednjega stanja, specifično z uporabo DT pa je naloga napovedovanje naslednje akcije. Prav tako DT dobro generalizira oziroma se nauči sistema, ki ga mora upravljati samo iz učnih podatkov.

Kljub svojim prednostim pa ima uporaba DT tudi nekaj omejitev. Ena glavnih slabosti je potreba po obsežnih učnih podatkih. Za učinkovito delovanje DT so potrebne velike količine kakovostnih podatkov, ki zajemajo širok spekter možnih scenarijev. V našem primeru zaradi obstoječe simulacije to ni bil problem, vendar je zbiranje takih podatkov lahko časovno in finančno zahtevno, še posebej v realnih industrijskih okoljih.

Druga omejitev je RTG, kjer moramo na začetku upravljanja poleg stanj posredovati nek ciljni donos. To je lahko zelo problematično, če ne vemo, kolikšna naj bi bila maksimalna akumulirana nagrada za določen problem in dolžino epizode.

Čeprav model ohranja temperaturo znotraj dovoljenega intervala, bi lahko še dodatno izboljšali njegovo delovanje. To bi lahko storili z izboljšanjem učnega podatkovnega nabora ter z boljšim sistemom nagrajevanja v samem učnem naboru. Primer poskusa nagrajevanja bi lahko bil, da poleg kazni za prekoračitev praga (ang. overshoot) dodamo kazen pri oscilacijah in/ali nagrade za stabilnost temperature okoli ciljne temperature, kjer bi dodelili čim višjo nagrado, tem dlje ostane temperatura znotraj dovoljenega intervala. Prav tako bi lahko izboljšali njegovo učinkovitost z implementacijo sprotnega učenja (ang. online learning), kjer bi model nadalje usposabljali na podlagi sprotnih podatkov iz okolja. Sprotnemu učenju bi dodali sistem nagrajevanja za njegove akcije, kar bi omogočilo modelu, da prejme povratne informacije o kakovosti svojih odločitev. Model bi zbiral nove podatke o stanju, akcijah in nagradah ter te informacije uporabil za fino uglaševanje svojih parametrov. Redno in periodično ponovno usposabljanje bi zagotovilo, da se model stalno prilagaja spremembam v okolju ter izboljšuje svoje delovanje skozi čas.

Literatura

[1] J. Iqbal, M. Ullah, S. Khan, B. Khelifa, S. Ćuković. Nonlinear control systems - A brief overview of historical and recent advances. Nonlinear Engineering. 2017;6(4): 301-312. https://doi.org/10.1515/nleng-2016-0077

[2] F. Borrelli, A. Bemporad, and M. Morari, Predictive Control for Linear and Hybrid Systems. Cambridge: Cambridge University Press, 2017.

[3] S. J. Qin, T. A. Badgwell (2003). A survey of industrial model predictive control technology. Control engineering practice, 11(7), 733-764. https://doi.org/10.1016/S0967-0661(02)00186-7.

[4] J. R. Raol, R. Ayyagari (2019). Control systems: classical, modern, and AI-based approaches. CRC Press.

[5] K. Ogata (2020). Modern control engineering.

[6] M. Ghalandari, M.S. Irandoost, A. Maleki, et al. Applications of intelligent methods in various types of heat exchangers: a review. J Therm Anal Calorim 145, 1837–1848 (2021). https://doi.org/10.1007/s10973-020-10425-3

[7] S. G. Tzafestas (2012). Methods and applications of intelligent control (Vol. 16). Springer Science Business Media.

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin: Attention is All You Need, 2027, https://arxiv.org/abs/1706.03762

[9] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, I. Mordatch: Decision Transformer: Reinforcement Learning via Sequence Modeling, 2021, https://arxiv.org/abs/2106.01345