VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS INSTITUTAS
KOMPIUTERINIO IR DUOMENŲ MODELIAVIMO KATEDRA

Magistrinis baigiamasis darbas

Kompiuterinis biojutiklių modeliavimas

Atliko:
Danielius Dvareckas parašas

Vadovas:
Prof. dr. Tadas Meškauskas

Vilnius
2019

Turinys

Santrauka

Biojutiklis – analitinis įrenginys, nustatantis biologinės kilmės signalą jį paverčiant elektriniu signalu [34]. *In vitro* biojutiklių kūrimas ir gamyba reikalauja daug ilgai trunkančių eksperimentų bei, dažnai, brangių reagentų [5], [48]. Tiesioginiai ir netiesioginiai kaštai gali būti sumažinti, kuriant biojutiklių matematinius modelius ir juos sprendžiant skaitiniu būdu.

Magistriniame darbe tiriamas matematinis biojutiklio modelis, įskaitantis substrato, produkto, fermento ir fermento-produkto komplekso kinetiką. Sprendžiama trijų netiesinių difuzijos-reakcijos lygčių sistema. Aprašomi skaitiniai metodai: diferencialinių lygčių pakeitimas į skirtumines schemas, iteracinis metodas, skaitinis diferencijavimas. Pateikiamas modeliuojamos sistemos biocheminis modelis: fermentinių reakcijų kinetika, difuzijos procesas.

Skaitinis eksperimentas, randantis biojutiklio atsaką, realizuotas PYTHON programavimo kalba. Sudaryta biojutiklio atsako priklausomybės nuo pradinių sąlygų duomenų bazė. Atlikta biocheminių charakteristikų įtakos biojutiklio atsakui analizė ir nagrinėtas pradinių sąlygų nustatymas, išvedant biojutiklio atsako ir pradinių sąlygų sąryšių formules bei pasitelkiant mašininį mokymą.

Summary

Biosensor is an analytical device that detects a biological signal by converting it into an electrical signal [34]. The development and production of *in vitro* biosensors requires time-consuming experiments and often expensive reagents [5], [48]. Direct and indirect costs can be reduced by developing mathematical models for biosensors and numerically solving them.

During this work, a biosensors mathematical model, which incorporates substrate, product, enzyme and enzyme-product complex kinetics, is analysed. The system of three nonlinear diffusion-reaction equations is solved. Different numerical methods are described: Crank–Nicolson finite difference method, iterative method, numerical differentiation. The biochemical model of the modeled system is presented: kinetics of enzymatic reactions, diffusion process.

The numerical experiment, which focuses on finding the response of the biosensor, was implemented in PYTHON programming language. A database of biosensor response dependence on the initial conditions is created.

The biosensor's response to the influence of biochemical characteristics was analysed. Determination of the initial conditions was elaborated by formulating the equations of the relationship between the response and the initial conditions of the biosensor and by using the machine learning.

Įvadas

Biojutikliai – analitiniai įrenginiai, kurie nustato analitę, paversdami stebimos reakcijos metu gaunamą biologinį atsaką į elektrinį signalą [34]. Vienas pirmųjų biojutiklių, dar vadinamas Klarko elektrodu (angl.: *Clark electrode*), buvo išrastas prof. Lelando Klarko (angl.: *Leland C. Clark Jr.*). Tai amperometrinis elektrodas, leidžiantis nustatyti deguonies įsotinimo lygį kraujyje. Šis biojutiklis veikia matuodamas elektros srovės pokyčius, susidarančius dėl platinos elektrodo paviršiuje besiredukuojančių deguonies molekulių [22].

Biojutikliai plačiai taikomi aplinkosaugoje, maisto kokybės patikroje ir kontrolėje [51]. Jie ypatingai svarbūs medicinoje dėl savo biologinės kilmės, galimybės operuoti sudėtingose sistemose, trumpo atsako laiko bei mažo dydžio [50]. Todėl yra aktualu nagrinėti modelines biojutiklių sistemas, ir jose vykstančius procesus: difuziją ir fermentines reakcijas. Ir žinoma, šiuos procesus lemiančius biocheminius parametrus: produkto degradacijos laipsnį, reakcijos greitį apibrėžiančias konstantas, pH įtaką, produkto ir substrato koncentracijas.

Biojutikliai yra sudėtingos daugiasluoksnės, heterogeninės analitinės sistemos, todėl biojutiklio atsako įvertinimas nėra trivialus [35]. Biojutiklio kūrimas, gamyba, ir eksperimento metu naudojami reagentai, dažniausiai, yra brangūs, o didelis eksperimentų kartojimas – neefektyvus [5], [48]. Kaštus galima sumažinti biojutikliams sukuriant matematinius modelius ir juos išsprendžiant analitiniu arba skaitiniu būdu. Modelio sprendimas analitiniu būdu turi trūkumų: nepritaikomas esant sudėtingoms biokatalitinėms sistemoms ar kraštinės sąlygoms, riboto reaguojančių medžiagų koncentracijos intervalo [7]. Todėl neretai pasirenkamas pastarasis – skaitinis metodas – leidžiantis spręsti uždavinius pasitelkiant kompiuterio skaičiuojamąją galią. Biojutiklis modeliuojamas kompiuteriniais metodais, pakeičiant matematinio modelio diferencialines lygtis į skirtumines schemas, o pats biojutiklis simuliuojamas skaitmeniškai.

Tačiau ir biojutiklių modeliavimas skaitiniais metodais turi trūkumų. Vienas iš jų yra ilga skaitinio eksperimento trukmė. Tai reiškia, kad žinant tik biojutiklio atsaką ir norint sužinoti biojutiklio charakteristikas, reikia atlikti nemažą kiekį ilgai trunkančių skaitinių eksperimentų. Šio magistrinio darbo metu ši problema sprendžiama nustačius biojutiklio, atsižvelgiančio į substrato, produkto, fermento, fermento-produkto kinetiką, charakteristikas, kuomet žinomas biojutiklio atsakas.

Darbo tikslas:
Žinant biojutiklio atsaką, nustatyti biojutiklio, atsižvelgiančio į substrato (S), produkto (P), fermento (E), fermento-produkto (EP) kinetiką, charakteristikas.

Darbo uždaviniai:

- išanalizuoti literatūrinę medžiagą, susijusią su dirbtiniais neuroniniais tinklais, biojutikliais ir jų kompiuteriniu modeliavimu;
- remiantis literatūra, įgyvendinti biojutiklio matematinį modelį, įskaitantį substrato, produkto, fermento, fermento-produkto kinetiką;
- įvertinti įgyvendinto biojutiklio matematinio modelio atsaką, remiantis [36] metodine medžiaga;
- išanalizuoti pradinių sąlygų daromą įtaką biojutiklio atsakui.

Darbo struktūra:

Pirmame skyriuje skaitytojas supažindinamas su biojutikliais, jų veikimo principu, nagrinėjamoje sistemoje vykstančiais procesais, bei skaitiniu tokio biojutiklio modeliavimu.

Antrame skyriuje pateikiamas nagrinėjamo biojutiklio matematinis ir skaitinis modeliai.

Trečiame skyriuje skaitytojas supažindinamas su mašininiu mokymu ir dirbtiniais neuroniniais tinklais.

Ketvirtame skyriuje pateikiamas įgyvendintas biojutiklio modelio, įskaitančio substrato, produkto, fermento, fermento-produkto kinetiką, skaitinis eksperimentas. Aptariamas gautas rezultatas ir šio rezultato verifikavimas.

Penktame skyriuje aptarta sukonstruota duomenų bazė, skirta skaitinių eksperimentų, analogiškų ketvirtame skyriuje aprašytam skaitiniui eksperimentui, duomenų rinkimui, keičiant skaitinio eksperimento pradines sąlygas. Šie duomenys naudojami tolimesnių skyrių medžiagoje.

Šeštame skyriuje atliekamas skaitinio eksperimento biojutiklio atsako priklausomybės tyrimas nuo pradinių sąlygų, keičiant tik vieną biojutiklio charakteristiką. Nagrinėjama galimybė atstatyti varijuojamą biojutiklio charakteristiką, žinant tik biojutiklio atsaką.

Septintame skyriuje atliekamas skaitinio eksperimento biojutiklio atsako priklausomybės tyrimas nuo pradinių sąlygų, keičiant daugiau nei vieną biojutiklio charakteristiką. Nagrinėjama galimybė atstatyti varijuojamas biojutiklio charakteristikas žinant biojutiklio atsaką, pasitelkiant mašininį mokymą, dirbtinius neuroninius tinklus.

Aštuntame skyriuje pateikiamos darbo išvados.

Devintame skyriuje aptariamos ateities tyrimų gairės.

Skyrius 1, skyrius 2, skyrius 4 ir dalis įvado įtraukti iš tiriamojo darbo, išskyrus patobulinimus ir klaidų pataisymus, atsižvelgiant tiek į autoriaus pastebėjimus, tiek į tiriamojo darbo metu pateiktas recenzento pastabas ir rekomendacijas.

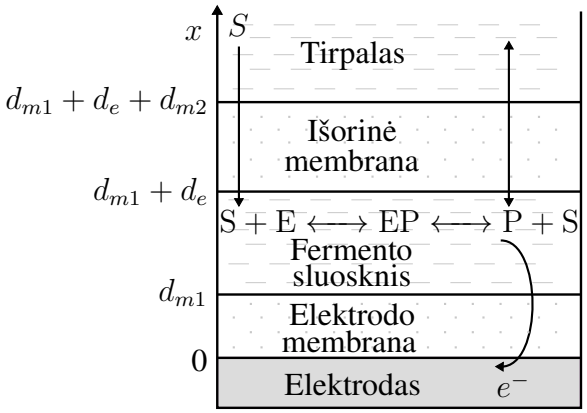
1. Biojutiklių modeliavimas

1.1. Biojutikliai

Biojutikliai (angl.: *biosensors*) – jutiklių klasė, kurioje naudojamos specifinės biologiškai aktyvios medžiagos, pasižyminčios dideliu tikslumu ir jautrumu. Pagrindinės biojutikio dalys yra biologinis atpažinimo elementas ir keitiklis [6]. Biologinis atpažinimo elementas, nustatantis analitę, turi pasižymėti specifiškumu (reaguoti tik į šią analitę). Tai gali būti imobilizuotas baltymas: fermentas, antikūnas ar receptorinis baltymas. Tačiau galimos ir sudėtingesnės sistemos, tokios kaip organelės, ląstelės ar net audinių regionai [46]. Biojutiklyje esantis keitiklis yra įrenginio dalis, kuri paverčia biologinio atpažinimo elemento sukuriamą biocheminį signalą į elektrinį. Ši tarpusavio sąveika yra biojutiklio veikimo principas [30]. Atsižvelgiant į keitiklio tipą, biojutikliai skirstomi į elektrocheminius, mechaninius, optinius, kalorimetrinius ir akustinius [20].

Šiame darbe nagrinėjamas elektrocheminio tipo biojutiklis, kurio mechanizmas paremtas biocheminės reakcijos metu generuojamos elektros srovės nustatymu (1 pav.). Elektrodas panardinamas į pastovios koncentracijos substrato tirpalą. Tuomet substrato molekulės difunduoja pro pusiau pralaidžią išorinę membraną d_{m2} į fermento sluoksnį. Fermento paskirtis yra katalizuoti substrato virtimą į produktą. Substratui prisijungus prie fermento aktyviojo centro vykdoma katalizinė reakcija, kurios metu susidaro elektrochemiškai aktyvus produktas, generuojantis elektrodo registruojamą elektros srovę. Elektrocheminio biojutiklio atveju ši elektros srovė vadinama biojutiklio atsaku. Kadangi biojutiklio atsakas tiesiogiai priklauso nuo vykstamos reakcijos greičio ir trukmės, kiekybiškai įvertinus srovės stiprį nustatoma susidariusio produkto koncentracija.

Fermentinės reakcijos kinetika ir difuzija yra vieni svarbiausių sistemos procesų, kurie ir nulemia bendrą sistemos veikimą bei biojutiklio atsaką.

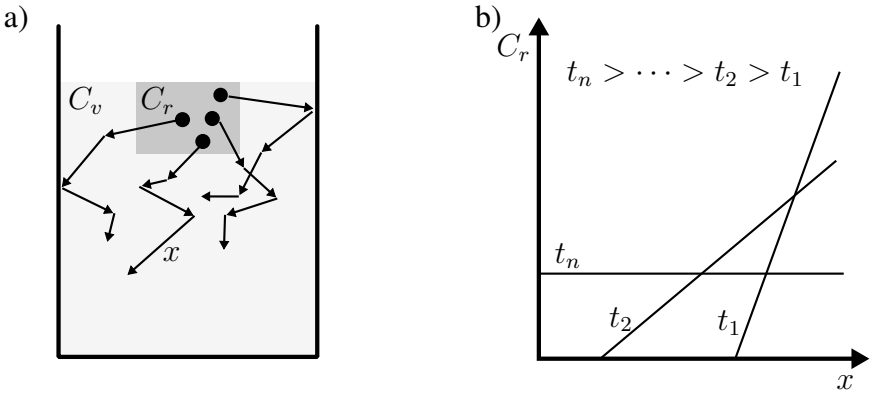


1 pav. Elektrocheminio biojutiklio schema. Substratas pro pusiau pralaidžią membraną difunduoja į fermento sluoksnį, kuriame fermento ir substrato reakcijos metu susidaro elektrochemiškai aktyvus produktas. Čia x – koordinatės kintamasis, d_{m1} – elektrodą gaubiančios membranos storis, d_e – fermento sluoksnio storis, d_{m2} – išorinės membranos storis, S – substratas, E – fermentas, EP – fermento-produkto kompleksas, P – produktas, e^- – elektronas.

1.2. Biojutiklių matematinis modeliavimas

1.2.1. Difuzija

Difuzija – tai masės pernašos procesas, kurio metu dėl spontaninio molekulių judėjimo, skysčių, dujų ir kietųjų kūnų molekulės maišosi tarpusavyje pagal koncentracijos gradientą [15]. Vienas iš difuzijos pavyzdžių yra rašalo lašinimas į vandenį (2a pav.). Iš pradžių, rašalo spalva pasiskleidžia keletą milimetrų, tačiau laikui bėgant visas vandens tūris nusidažo tolygia spalva. Difuzijos sukeltas chaotiškas molekulių judėjimas pagal koncentracijos gradientą, t. y. iš didesnės koncentracijos į mažesnę, ir lemia visišką molekulių susimaišymą (2b pav.) [33].



2 pav. a) Į vandenį įlašinus rašalo, molekulės chaotiškai bejudėdamos susimaišo. Čia x – koordinatės kintamasis, C_v – tirpiklio (vandens) koncentracija, C_r – tirpinio (rašalo) koncentracija. b) Rašalo koncentracijos C_r pasiskirstymas tirpiklio tūryje koordinatės kintamojo x ir laiko kintamojo t atžvilgiu. Laikui einant rašalo koncentracija per visą tirpiklio tūrį tampa tolygi. Paveikslai pagal [41], gauti autoriaus nepriklausomai.

Tokį molekulių maišymąsi pagal koncentracijos gradientą aprašo pirmasis Fiko (angl.: *Fick*) dėsnis:

$$J = -D\frac{\partial C}{\partial x}, \tag{1.1}$$

kur J – medžiagos kiekio srauto tankis (mol $m^{-2}s^{-1}$), C – medžiagos molinė koncentracija (mol m^{-3}), x – koordinatės kintamasis, D – difuzijos konstanta (m^2s^{-1}). Difuzijos koeficientas nusako kaip sparčiai dalelės juda tirpale. Kiekvienai medžiagai difuzijos konstanta skirtinga; kuo difuzijos koeficientas yra didesnis, tuo dalelės juda greičiau ir maišosi tarpusavyje.

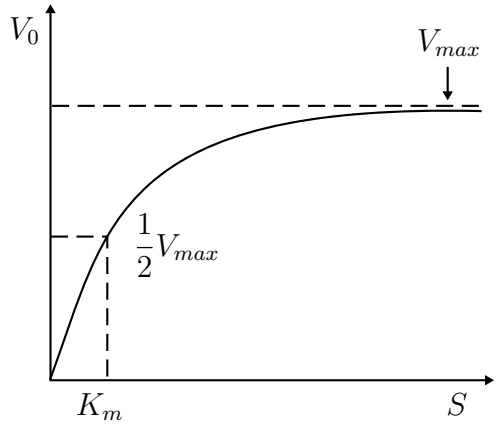
Tačiau pirmasis Fiko dėsnis neįvertina laiko kintamojo. Antrasis Fiko dėsnis, dar vadinamas difuzijos lygtimi, nusako difuzijos sukeltus koncentracijos pokyčius laiko atžvilgiu:

$$\frac{\delta C(x,t)}{\delta t} = D\frac{\delta^2 C(x,t)}{\delta x^2}, \tag{1.2}$$

kur t – laiko kintamasis, x – koordinatės kintamasis, D – difuzijos koeficientas, C – koncentracija. 1.2 lygtis yra ypač svarbi vertinant biojutiklio atsaką per tam tikrą laiko intervalą, nes elektrochemiškai aktyvios dalelės elektrodo paviršių pasiekia būtent difuzijos būdu [40]. Apibrėžus difuzijos lygčių sistemą su tam tikromis kraštinėmis sąlygomis, galima apskaičiuoti eksperimentiniu būdu nustatomus elektros srovės pokyčius, t. y. biojutiklio atsaką.

1.2.2. Fermentinių reakcijų kinetika

Fermentų katalizuojamų reakcijų greitis priklauso nuo įvairių faktorių. Pagrindiniai faktoriai darantys įtaką fermentinės reakcijos greičiui yra temperatūra, pH, reagentų cheminė prigimtis ir substrato koncentracija. Pastarasis veiksnys yra vienas iš svarbiausių [11]. Pradiniu laiko momentu, esant mažoms substrato koncentracijoms, reakcijos greitis tiesiogiai priklauso nuo substrato koncentracijos. Toliau didinant koncentraciją, reakcijos greitis lėtėja ir pasiekiamas nekintamas greitis, kuris vadinamas maksimaliu reakcijos greičiu V_{max}. Toks substrato koncentracijos ir reakcijos greičio santykis vadinamas hiperboliniu ir demonstruoja fermento prisotinimą substratu. Įsotinimo reiškinys lemiamas fiksuoto skaičiaus fermento molekulių, kurių kiekviena turi fiksuotą skaičių substrato surišimo vietų. Didėjant substrato koncentracijai yra pasiekiama riba, kuomet visos fermento surišimo vietos (aktyvieji centrai) yra užimtos, tad, nors kiekvienas fermentas dirba jam būdingu maksimaliu greičiu, reakcijos greitis nedidėja [25].



3 pav. Fermentinės reakcijos greičio priklausomybė nuo substrato koncentracijos. V_0 – pradinis reakcijos greitis, V_{max} – maksimalus reakcijos greitis, K_m – Michaelis-Menten reakcijos konstanta, S – substrato koncentracija. Paveikslas pagal [49], gautas autoriaus nepriklausomai.

L. Michaelis (angl.: *Michaelis*) ir M. Menten (angl.: *Menten*) pasiūlė modelį, paaiškinantį reakcijos greičio ir substrato koncentracijos priklausomybę. Michaelis-Menten fermentinės katalizės reakcijos lygtis:

$$E + S \underset{k_{-1}}{\overset{k_{+1}}{\rightleftharpoons}} EP \underset{k_{-3}}{\overset{k_{+3}}{\rightleftharpoons}} E + P, \tag{1.3}$$

kur E – fermentas, S – substratas, P – produktas, EP – fermento ir produkto kompleksas, k_{+1} – EP susidarymo reakcijos greičio konstanta, k_{+3} – EP skilimo į fermentą ir produktą reakcijos greičio konstanta, k_{-1} – EP disociacijos į fermentą ir substratą reakcijos greičio lygtis, k_{-3} – EP susidarymo iš fermento ir produkto reakcijos greičio konstanta.

Tuomet pradinis reakcijos greitis V_0 išreiškiamas kaip:

$$V_0 = \frac{V_{max} S}{K_m + S}, \tag{1.4}$$

kur V_{max} – maksimalus reakcijos greitis, S – substrato koncentracija, K_M – Michaelio konstanta.

Michaelio konstanta yra tokia substrato koncentracija, kuriai esant reakcijos greitis yra lygus

pusei didžiausio ($\frac{1}{2}V_{max}$). K_M konstanta yra labai naudinga vertinant baltymo afiniškumą skirtingiems substratams [11]. K_M galima išreikšti per reakcijos greičio konstantas [25]:

$$K_M = \frac{k_{p3} + k_{m1}}{k_{p1}}. \tag{1.5}$$

Svarbu paminėti, kad fermento katalizuojamos reakcijos greitis neviršija difuzijos greičio. Difuzija yra limituojantis faktorius, kadangi molekulės judėjimas yra lemiamas difuzijos [3], [25], t. y. molekulės negali tarpusavyje reaguoti greičiau, nei laikas, per kurį priartėja viena prie kitos.

1.3. Biojutiklių skaitinis modeliavimas

Dalinių diferencialinių lygčių aprašančių biojutiklio veikimą analitinis sprendimas turi nemažai trūkumų ir yra komplikuotas, todėl sudėtingose modelinėse biojutiklių sistemose sprendinio radimas neretai vykdomas skaitiniu būdu [6]. Skaičiavimus apsunkina ir tai, kad nuo laiko priklausantys difuzijos procesai yra aprašomi netiesinėmis lygtimis, kuriose reikia įvertinti kraštines sąlygas, nusakančias sistemos būseną sistemos kraštuose, ir pradines sąlygas, nusakančias sistemos pradinę būseną [39]. Matematinių modelių sprendimas skaitiniu būdu palengvina netiesinių sistemų nagrinėjimą, nes pasitelkiama kompiuterio skaičiuojamoji galia.

Šiame darbe fermentinės reakcijos ir difuzijos lygtis sprendžiame skaitiniu būdu, modeliuodami sistemą, turinčią vienmatę geometriją ir pastovius difuzijos parametrus.

1.3.1. Diskretizavimas

Šis metodas paremtas funkcijos verčių pakeitimu vertėmis konkrečiose būsenose. Nagrinėsime tolydžią funkciją C, priklausančią nuo koordinatės kintamojo x ir laiko kintamojo t. Diskretizacijos proceso metu, tolydžiai funkcijai sudaromas tinklelis, t. y. koordinatės kintamojo x ir laiko kintamojo t parametrai parenkami tik konkrečiose būsenose. Tuomet vietoje tolydžios funkcijos $C(x,t)$ sprendžiame diskretizuotą funkciją:

$$C_{i,j} \equiv C(x_i, t_j), \qquad kur \qquad x_i = inx, \qquad t_j = jmt. \tag{1.6}$$

Čia i – koordinatės indeksas, n – konstanta, padalinanti koordinatės intervalą į diskrečius intervalus, j – laiko indeksas, m – konstanta, padalinanti laiko intervalą į diskrečius intervalus, o x_i ir t_j – tinklelio taškai, priklausantys atitinkamiems koordinatės ir laiko tinklelio intervalams [16]. Šiuo nagrinėtu atveju sudaryti tinkleliai yra tolygūs visame sprendinių intervale, tačiau analogiškai galima sudaryti ir netolygius tinklelius.

1.3.1.1. Crank–Nicolson baigtinių skirtumų schema

Sukurta įvairių metodų, leidžiančių išspręsti sudėtingas diferencialines lygtis skaitiškai ir gauti labai geras aproksimacijas ar tikslius sprendinius. Vienas iš tokių aproksimacijos metodų yra Cranko–Nicolsono (angl.: *Crank-Nicolson*) baigtinių skirtumų schema, kuri gali būti išreikštinė arba neišreikštinė [39].

Neišreikštinės schemos sprendinys gaunamas nepriklausomai nuo kitų parametrų t. y. skaičiuojamas viename laiko momente. Išreikštinės schemos atveju, sprendinys priklauso nuo kelių parametrų, todėl reikia spręsti sudėtingesnes lygčių sistemas. Nors toks būdas reikalauja atlikti daugiau skaičiavimų, jis yra gerokai stabilesnis, ir įgalina atlikti skaičiavimus didesniu laiko

žingsniu. Dėl šių priežasčių Cranko–Nicolsono išreikštinis metodas dažnai, kaip ir šiame darbe, naudojamas sprendžiant sistemas, kuriose vyksta difuzija [42].

Cranko–Nicolsono išreikštinė baigtinių skirtumų schema paremta lygčių centriniu verčių suvidurkinimu (4 pav.):

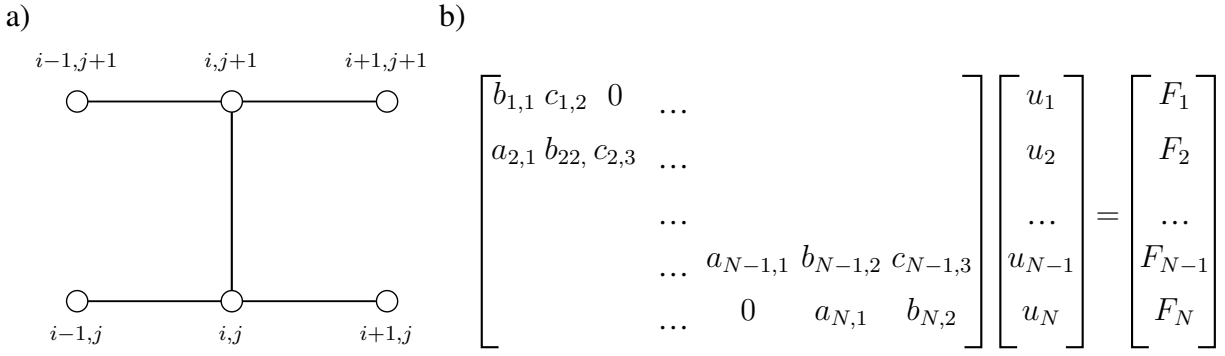
$$\frac{C_i^{j+1} - C_i^j}{\Delta t} = D\frac{(C_{i-1}^{j+1} - 2C_i^{j+1} + C_{i+1}^{j+1}) + (C_{i-1}^j - 2C_i^j + C_{i+1}^j)}{2\Delta x^2}, \tag{1.7}$$

kur D – difuzijos konstanta, x – koordinatės kintamasis, t – laiko kintamasis, i – koordinatės tinklelio indeksas, j – laiko tinklelio indeksas.

Pažymėjus $r = D\Delta t/\Delta x^2$, 1.7 lygtis gali būti užrašyta lakoniškiau:

$$-rC_{i+1}^{j+1} + (2+2r)C_i^{j+1} - rC_{i-1}^{j+1} = rC_{i-1}^j + (2+2r)C_i^j + C_{i+1}^j. \tag{1.8}$$

Visi nariai dešinėje 1.8 lygties pusėje yra žinomi. Svarbu tai, kad 1.8 lygtis sudaro matricos sistemą, turinčią nenulinius narius centrinėje įstrižainėje, o taip pat ir gretimose įstrižainėse (4b pav.). Tokia matrica vadinama tridiagonaline matrica ir nesunkiai išsprendžiama Thomas algoritmu (angl.: *Thomas algorithm*) [39].



4 pav. a) Crank-Nicolson išreikštinė baigtinių skirtumų schema vienos dimensijos atveju. Čia i – koordinatės tinklelio indeksas, j – laiko tinklelio indeksas. b) Tridiagonalinės matricos pavyzdys: ypatingas tiesinių lygčių atvejis, kuomet tiesinių lygčių matricos nenuliniai elementai yra išsidėstę tik centrinėje, ir gretimose įstrižainėse centrinei įstrižainei. Paveikslai pagal [39], gauti autoriaus nepriklausomai.

1.3.1.2. Skirtuminės schemos sprendinio radimas: Thomas algoritmas

Kai kurios matricos turi savybes, kurias galima pasitelkti efektyvesniam sprendinio radimui. Viena iš tokių matricos pavyzdžių yra tridiagonalinė matrica, kurioje tiesinių lygčių matricos nenuliniai elementai yra išsidėstę tik centrinėje, bei gretimose centrinei įstrižainei (4b pav.). Ši matrica gali būti išspręsta tiesioginio ir atvirkštinio pakeitimų būdais, išvengiant operacijų su nuliniais elementais. Toks sprendimo būdas vadinamas Thomas algoritmu.

Pavyzdžiui, 1.8 lygtį užrašius kaip matricos lygtį:

$$\mathbf{AU} = \mathbf{F}, \tag{1.9}$$

kur nežinomieji randami vykdant Thomas algoritmą:

Thomas algoritmas itin palengvina ir pagreitina Crank-Nicolson išreikštinės skirtuminės schemos sprendinio radimą. Todėl šio darbo metu nagrinėjamas skaitinis biojutiklio modelis sprendžiamas Crank-Nicolson išreikštine skirtumine schema ir Thomas algoritmu.

1 algoritmas. Thomas algoritmas

$\beta_1 \leftarrow b_1$

$\gamma_1 \leftarrow F_1/\beta_1$

for $i = 2, \ldots, n$:

$$\beta_i \leftarrow b_i - \left(\frac{a_i c_{i-1}}{\beta_{i-1}}\right)$$

$$\gamma_i \leftarrow \frac{(F_i - a_i\gamma_{i-1})}{\beta_i}$$

$y_n \leftarrow \gamma_n$

for $j = 1, \ldots, n-1$:

$$u_{n-1} \leftarrow \gamma_{n-j} - \left(\frac{c_{n-j}u_{n-j+1}}{\beta_{n-j}}\right)$$

1.3.2. Biojutiklio atsakas

Vykdant eksperimentą *in vitro*, elektrocheminio tipo biojutiklio atsakas nustatomas išmatavus srovės stiprio pokyčius nagrinėjamoje sistemoje. Biojutiklio srovės stipris i laiko momentu t yra proporcingas produkto difuzijos koeficientui ir produkto gradientui elektrodo paviršiuje [6]:

$$i(t) = n_e F D_P(0) \left.\frac{\partial P}{\partial x}\right|_{x=0}, \tag{1.10}$$

kur $i(t)$ – srovės tankis laiko momentu t, F – Faradėjaus konstanta, lygi 96485 C mol^{-1} [46], n_e – elektros krūvį pernešančių elektronų skaičius ties biojutiklio elektrodu, D_P – produkto difuzijos koeficientas elektrodo paviršiuje, P – produkto koncentracija elektrodo paviršiuje.

Biojutiklio atsakas laikui bėgant nusistovi prie tam tikros vertės. Tokia vertė, kuriai esant, biojutiklio atsakas nebekinta ($t \to \infty$), aprašoma empirine formule:

$$I_\infty = \lim_{t\to\infty} i(t), \tag{1.11}$$

kur I_∞ – galutinis biojutiklio atsakas nuostoviojoje būsenoje.

Nagrinėjamo biojutiklio atsakas priklauso nuo sistemoje vykstančio difuzijos proceso ir fermentinės reakcijos kinetikos, todėl, norint sukurti skaitinį tokio biojutiklio modelį, itin svarbu atsižvelgti į šiuos parametrus.

2. Biojutiklio modelis, įskaitantis S, P, E ir EP kinetiką

2.1. Matematinis modelis

Šiame darbe yra nagrinėjamas elektrocheminio tipo biojutiklio modelis, kurio veikimą lemia difuzijos ir fermentinės reakcijos kinetikos procesai [36].

Modeliuojamos biojutiklio sistemos (1 pav., 1.3 lygtis) nežinomieji yra substrato koncentracija S, produkto koncentracija P, fermento koncentracija E. Šias koncentracijas nagrinėjame kaip funkcijas, priklausančias nuo koordinatės kintamojo x ir laiko kintamojo t, o biojutiklio atsakas, proporcingas P koncentracijai, įvertinamas pagal susidariusį produkto kiekį.

In vitro vykdoma biocheminė reakcija priklauso nuo naudojamų cheminių reagentų prigimties ir fermento-substrato tarpusavio afiniškumo. Todėl *in silico* sistemos lygtis apibrėžiame atsižvelgdami į tiesioginės ir atvirkštinės reakcijų greičio konstantas: k_{-1}, k_{+1}, k_{-3}, k_{+3} (žr. 1.2.2 skyrių). Simuliuojamų medžiagų stabilumui įvertinti įvedamas substrato degradacijos santykinis greitis C_S ir produkto degradacijos santykinis greitis C_P.

Modeliuojamoje sistemoje taip pat atsižvelgiama į difuzijos procesus (žr. 1.2.1 skyrių): substrato difundavimą į fermento sluoksnį ir produkto difundavimą prie elektrodo membranos ribos (x = 0). Jiems įvertinti įvedamos substrato difuzijos konstantos skirtingose sistemos srityse $D_{S_{m1}}$, D_{S_e}, $D_{S_{m2}}$ ir produkto difuzijos konstantos skirtingose sistemos srityse $D_{P_{m1}}$, D_{P_e}, $D_{P_{m2}}$.

2.1.1. Pradinės sąlygos

Norint nustatyti galutines S, P, E koncentracijas, modeliavimo pradžioje turi būti apibrėžtos pradinės koncentracijos, kai t = 0:

$$S(x,0) = \begin{cases} 0, & 0 \leqslant x < d_{m1} + d_e + d_{m2}, \\ S_0, & x = d_{m1} + d_e + d_{m2}, \end{cases} \tag{2.1}$$

$$P(x,0) = 0, \qquad 0 \leqslant x \leqslant d_{m1} + d_e + d_{m2}, \tag{2.2}$$

$$E(x,0) = \begin{cases} 0, & 0 \leqslant x < d_{m1}, \\ E_0 & d_{m1} \leqslant x \leqslant d_{m1} + d_e, \\ 0 & d_{m1} + d_e \leqslant x \leqslant d_{m1} + d_e + d_{m2}, \end{cases} \tag{2.3}$$

Pradines vertes apibrėžiame taip, kad substrato koncentracija, pradiniu laiko momentu S_0, būtų pastovi tirpalo ir išorinės membranos riboje ($d_{m1} + d_e + d_{m2}$). Pradiniu laiko momentu produkto nėra, o fermentas lygus fermento koncentracijai pradiniu laiko momentu E_0 ir yra tik fermento sluoksnyje ($d_{m1} \leqslant x \leqslant d_{m1} + d_e$). S_0 ir E_0 parametrai pasirenkami pagal biocheminę prasmę.

2.1.2. Kraštinės sąlygos

Kraštinės sąlygos nurodo modelio ribas (1 pav.). Substratas gali būti aptinkamas tirpalo sluoksnyje ir difunduoti į fermento sluoksnį, tačiau negali patekti į elektrodo sluoksnį. Šiame modelyje išorinės membranos ir tirpalo riboje substrato koncentracija yra pastovi ir lygi S_0:

$$\left.\frac{\partial S}{\partial x}\right|_{x=0} = 0, \qquad S(x = d_{m1} + d_e + d_{m2}, t) = S_0, \qquad t > 0. \tag{2.4}$$

Tuo tarpu, produktas iš fermento sluoksnio, kuriame yra pagaminamas, gali difunduoti į abi puses: tiek į tirpalo sluoksnį, tiek į elektrodo ribą. Kuriamame modelyje, nors produktas gali patekti į elektrodo ribą, darome prielaidą, kad produktas labai greitai sureaguoja, todėl elektrodo riboje produkto koncentraciją prilyginame nuliui:

$$P(x=0,t)=0, \qquad P(x=d_{m1}+d_e+d_{m2},t)=0, \qquad t>0. \tag{2.5}$$

2.1.3. Vyksmo sąlygos

In silico vykdomos biocheminės reakcijos pagrindinių procesų sąlygos yra įvertinamos sudarant reakcijos-difuzijos lygtis:

$$\frac{\partial S}{\partial t}=\frac{\partial}{\partial x}\left(D_S(x)\frac{\partial S}{\partial x}\right)-C_S S-\alpha(x)\left(\left(k_{+1}S+k_{-1}\right)E-k_{-1}E_0\right), \tag{2.6}$$

$$\frac{\partial P}{\partial t}=\frac{\partial}{\partial x}\left(D_P(x)\frac{\partial S}{\partial x}\right)-C_P P-\alpha(x)\left(\left(k_{-3}P+k_{+3}\right)E-k_{-1}E_0\right), \tag{2.7}$$

kai sprendimų sritis: $0<x<d_{m1}+d_e+d_{m2}, \qquad t>0.$ (2.8)

$$\frac{\partial E}{\partial t}=-\alpha(x)\left(\left(k_{+1}S+k_{-3}P+k_{-1}+k_{+3}\right)E-\left(k_{-1}+k_{+3}\right)E_0\right), \tag{2.9}$$

kai sprendimų sritis: $0\leqslant x\leqslant d_{m1}+d_e+d_{m2}, \qquad t>0.$ (2.10)

Čia $D_S(x)$ ir $D_P(x)$ – funkcijos, atitinkamai nusakančios substrato ir produkto difuzijos koeficientų verčių priklausomybę nuo koordinatės kintamojo x skirtingose modeliavimo sistemos srityse:

$$D_S(x)=\begin{cases} D_{S_{m1}}, & 0<x<d_{m1},\\ D_{S_e}, & d_{m1}\leqslant x\leqslant d_{m1}+d_e,\\ D_{S_{m2}}, & d_{m1}+d_e<x<d_{m1}+d_e+d_{m2},\end{cases} \tag{2.11}$$

$$D_P(x)=\begin{cases} D_{P_{m1}}, & 0<x<d_{m1},\\ D_{P_e}, & d_{m1}\leqslant x\leqslant d_{m1}+d_e,\\ D_{P_{m2}}, & d_{m1}+d_e<x<d_{m1}+d_e+d_{m2},\end{cases} \qquad D_P(0)=\begin{cases} D_{P_{m1}}, & d_{m1}\neq 0,\\ D_{P_e}, & d_{m1}=0,\end{cases} \tag{2.12}$$

$$\alpha(x)=\begin{cases} 0, & 0<x<d_{m1},\\ 1, & d_{m1}\leqslant x\leqslant d_{m1}+d_e,\\ 0, & d_{m1}+d_e<x<d_{m1}+d_e+d_{m2},\end{cases} \qquad \alpha(0)=\begin{cases} 0, & d_{m1}\neq 0,\\ 1, & d_{m1}=0,\end{cases} \tag{2.13}$$

Čia $\alpha(x)$ – skirtingų modeliavimo sistemos sričių konstanta.

2.2. Skaitinis modelis

Praeitame skyriuje apibrėžtas matematinis modelis 2.6, 2.7, 2.9 diferencialinėmis lygtimis su pradinėmis sąlygomis 2.1, 2.2, 2.3 ir kraštinėmis sąlygomis 2.4, 2.5 srityje 2.10. Toks matematinis modelis gali būti sprendžiamas skaitiniu būdu.

Skaitiniame modelyje tolygiai diskretizuojame tiek koordinatės kintamąjį x, tiek laiko kintamąjį t. Diferencialinis modelis aproksimuojamas Crank-Nicolson išreikštine baigtinių skirtumų schema (žr. 1.3.1.1 skyrių), kurioje netiesines sistemos dalys sprendžiamos iteratyviniu metodu, o tiesinės – tridiagonalinės matricos algoritmu (žr. 1.3.1.2 skyrių).

2.2.1. Skaitinio eksperimento įrankiai

Nagrinėjamas eksperimento modelis įgyvendintas interpretuojama programvimo kalba PYTHON (versija 3.6.0). Programinio kodo rašymo patogumui naudotas "JetBrains PyCharm Community Edition" (versija 2016.3.2) teksto redaktorius. Parašyta konsolinė aplikacija įrašo į diską biojutiklio atsakus modeliavimo eigoje (sustabdant kuomet pasiekiamas stabdymo kriterijus), o taip pat ir generuoja pagrindinius koncentracijų grafikus.

Rezultatų grafikams generuoti testavimo tikslais panaudota *MatPlotLib* biblioteka, o operacijų su sąrašais patogumui – *Numpy* biblioteka. Galutiniai grafikai, kurie pateikti rezultatų srityje, apdoroti QtiPlot (versija 0.9.9) grafikų redaktoriumi.

2.2.2. Diskretaus tinklelio įvedimas

Koordinatės kintamojo x ir laiko kintamojo t diskretizacijai įvedami tolygūs tinkleliai.

Koordinatės kintamasis padalinamas į N dalių, kur N – natūralusis skaičius, nusakantis į kiek lygių dalių suskirstomas intervalas $[0, d_{m1} + d_e + d_{m2}]$. Tuomet įvedamas parametras h, nusakantis vieno žingsnio dydį metrais:

$$h = \frac{d_{m1} + d_e + d_{m2}}{N}. \tag{2.14}$$

Laiko kintamojo t atveju, pasirenkamas fiksuotas laiko žingsnio dydis τ = 0,001 s, kuris buvo patikrintas eksperimentiniu būdu, lyginant su dešimt kartų didesniu laiko žingsniu (τ = 0,01 s). Šis sprendimas buvo pagrįstas atsižvelgus į eksperimento laiko trukmę ir gauto rezultato kokybiškumo santykį.

2.2.3. Vyksmo algebrinės lygtys

Diferencialinių lygčių sprendimui rasti, sudaroma Crank-Nicolson išreikštinė baigtinių skirtumų schema (žr. 1.3.1.1 skyrių), kuri sudaroma pagal 2.6, 2.7, 2.9 lygtis, esant pradinėms sąlygoms 2.1, 2.2, 2.3, ir kraštinėms sąlygoms 2.4, 2.5 srityje 2.10. Lygčių užrašymui palengvinti, koncentracijos užrašomos kaip dabartinės iteracijos ir sekančios iteracijos parametro vidurkis (žym. vid); analogiškai, vid_{pr} – dabartinės iteracijos ir praeitos iteracijos parametro vidurkis.

Netiesinių algebrinių lygčių sistema suvidurkinto substrato koncentracijos atžvilgiu:

$$a_i S_{i-1}^{vid} - c_i(E_i^{vid}) S_i^{vid} + b_i S_{i+1}^{vid} = -F_i(S_i^k, E_i^{vid}), \quad i = 1, 2, \ldots, N-1, \tag{2.15}$$

$$3S_0^{vid} - 4S_1^{vid} + S_2^{vid} = 0, \quad S_N^{vid} = S_0. \tag{2.16}$$

Čia $a_i = \dfrac{D_S(x_{vid_{pr}})}{h^2}$, $b_i = \dfrac{D_S(x_{vid})}{h^2}$, $c_i = a_i + b_i + \frac{2}{\tau_k} + C_1 + \alpha_i k_{+1} E_i^{vid}$, $F_i(S_i^k, E_i^{vid}) = \frac{2}{\tau_k} S_t^k + \alpha_i k_{-1} E_0 \Big(1 - \dfrac{E_i^{vid}}{E_0}\Big)$.

Netiesinių algebrinių lygčių sistema suvidurkinto produkto koncentracijos atžvilgiu:

$$\tilde{a}_i P_{i-1}^{vid} - \tilde{c}_i(E_i^{vid}) P_i^{vid} + \tilde{b}_i P_{i+1}^{vid} = -G_i(P_i^k, E_i^{vid}), \quad i = 1, 2, \ldots, N-1, \tag{2.17}$$

$$P_0^{vid} = 0, \quad P_N^{vid} = 0. \tag{2.18}$$

Čia $\tilde{a}_i = \dfrac{P_S(x_{vid_{pr}})}{h^2}$, $\tilde{b}_i = \dfrac{P_S(x_{vid})}{h^2}$, $\tilde{c}_i = \tilde{a}_i + \tilde{b}_i + \frac{2}{\tau_k} + C_1 + \alpha_i k_{+1} E_i^{vid}$, $G_i(S_i^k, E_i^{vid}) = \frac{2}{\tau_k} S_t^k + \alpha_i k_{-1} E_0 \Big(1 - \dfrac{E_i^{vid}}{E_0}\Big)$.

Netiesinių algebrinių lygčių sistema suvidurkinto fermento koncentracijos atžvilgiu:

$$E_i^{vid} = E_i^k \frac{\dfrac{2}{\tau_k} + \alpha(k_{-1} + k_{+3}) \dfrac{E_0}{E_i^k}}{\dfrac{2}{\tau_k} + \alpha_i(k_{+1} S_i^{vid} + k_{-3} P_i^{vid} + k_{-1} + k_{+3})}, \qquad i = 0, 1, \ldots, N. \tag{2.19}$$

2.2.4. Pradinės sąlygos

Pradinės sąlygos apibrėžiamos pagal 2.1.1 skyriaus sąlygas, atsižvelgiant į įvestą koordinatės kintamojo tinklelį:

$$S_i^0 = \begin{cases} 0, & i = 0, 1, \ldots, N-1, \\ S_0 & i = N, \end{cases} \tag{2.20}$$

$$P_i^0 = 0, \qquad i = 0, 1, \ldots, N, \tag{2.21}$$

$$E_i^0 = \begin{cases} 0, & 0 \leqslant x_i < d_{m1}, \\ E_0 & d_{m1} \leqslant x_i \leqslant d_{m1} + d_e, \\ 0 & d_{m1} + d_e \leqslant x_i \leqslant d_{m1} + d_e + d_{m2}, \end{cases} \qquad i = 0, 1, \ldots, N. \tag{2.22}$$

2.2.5. Kraštinės sąlygos

Kraštinės sąlygos apibrėžiamos pagal 2.1.2 skyriaus sąlygas kiekvienam laiko kintamojo tinklelio žingsniui:

$$3S_0^{vid} - 4S_1^{vid} + S_2^{vid} = 0, \quad S_N^{vid} = S_0, \quad k = 0, 1, \ldots. \tag{2.23}$$

$$P_0^{vid} = 0, \quad P_N^{vid} = 0, \quad k = 0, 1, \ldots. \tag{2.24}$$

2.2.6. Biojutiklio atsakas

In vitro sąlygomis – biojutiklio atsakas laikui bėgant nusistovi prie tam tikros vertės, tačiau *in silico* sąlygomis – stabdymo kriterijus turi būti pasirenkamas. Biojutiklio atsakas skaičiuojamas empirine formule:

$$I \approx i(t_I), \qquad t_I = \min_{t_k \in \omega_\tau} \left(t_k : \frac{i'(t_k) \cdot 10^3}{i'(t_{did})} < 1 \right), \tag{2.25}$$

kur ω_τ – laiko tinklelio aibė, t_k – laiko vertė, priklausanti laiko tinklelio aibei, $i'(t_k)$, $i'(t_{did})$ – biojutiklio atsako išvestinė atitinkamais laiko momentais t_k, k_{did}. k_{did} yra toks laiko momentas, kuomet biojutiklio atsako išvestinė yra didžiausia, t. y. atsakas auga staigiausiai.

Empiriškai ieškomas toks laiko momentas $t_I = t_k$, kuriam esant biojutiklio atsako augimas yra sumažėjęs 10^3 kartų, lyginant su visos modeliavimo trukmės metu didžiausiu atsako augimu ($i'(t_{did})$).

Augimo kampas (atsako išvestinė) skaičiuojamas formule:

$$i'(i_{t_{k-1}}, i_{t_k}, i_{t_{k+1}}, t_{k-1}, t_k, t_{k+1}) = i_{t_{k-1}} \frac{2t_k - t_k - t_{k+1}}{2h^2} - i_{t_k} \frac{2t_k - t_{k-1} - t_{k+1}}{h^2} + i_{t_{k+1}} \frac{2t_k - t_{k-1} - t_k}{2h^2}. \tag{2.26}$$

2.3. Sprendimas

Nagrinėjamas biojutiklio skaitinis modelis yra sprendžiamas netiesinių lygčių sistema, ieškant trijų pagrindinių nežinomųjų:

$$S_i^{vid}, \qquad P_i^{vid}, \qquad E_i^{vid}, \tag{2.27}$$

kur $i = 0, 1, \ldots, N$.

Tokios netiesinės sistemos tiesioginis sprendimas yra problematiškas, todėl sistema sprendžiama iteraciniu metodu. Pažymėjus E_i^{vid} kaip jau žinomą vertę, lygčių sprendimas palengvinamas, nes likę nežinomi – parametrai S ir P tikslinami pagal tiesinę dalį. O sužinojus S, P randamas E artinys, kuris panaudojamas sekančioje iteracijoje. Ciklas kartojamas, kol pasiekiamas stabdymo kriterijus.

Toliau pateikta taikyto iteracinio metodo žingsnių seka, kur indeksais $sena$ ir $nauja$, žymėsime senos ir naujos iteracijos artinius.

Atliekame žingsnius:

1. Priskiriame pradinį fermento koncentracijų E_i^{sena} artinius, lygius fermento koncentracijai pradiniu laiko momentu E_i^k, kai $k = 0$ ir i = 0, 1, …, N;
2. Išsprendę 2.15, 2.16 lygtis, apskaičiuojame artinius S_i^{nauja};
3. Išsprendę 2.17, 2.18 lygtis, apskaičiuojame artinius P_i^{nauja};
4. Išsprendę 2.19 lygtį, apskaičiuojame artinius E_i^{nauja};
5. Pagal empirinę formulę 2.25 įvertiname biojutiklio atsaką:
   5.1. Jei stabdymo kriterijus tenkinamas, stabdome skaičiavimą ir spausdiname rezultatą.

5.2. Jei stabdymo kriterijus nepatenkinamas, skaičiavimai toliau tęsiami nuo 6 žingsnio.

6. Patikslintus artinius pervadiname senaisiais:

$$S_i^{sena} = S_i^{nauja}, \quad P_i^{sena} = P_i^{nauja}, \quad E_i^{sena} = E_i^{nauja}, \quad i = 0, 1, \ldots, N; \tag{2.28}$$

7. Tęsiame skaičiavimus nuo 2 žingsnio.

3. Mašininis mokymas

Antikos laikotarpiu, kuomet dar net nebuvo kilusi mintis apie kompiuterius, žmonės visais būdais bandė rasti reikšmingus motyvus sukauptuose duomenyse. Garsus graikų astronomas ir matematikas Ptolemėjus (angl.: *Claudius Ptolemy*) pritaikė žvaigždžių judesių stebėjimus formuodamas geocentrinį pasaulio modelį, kuriame planetos juda ne tik apskritimine trajektorija, bet ir mažesniais epiciklais. Taip paaiškindamas planetų atgalinius judesius. Šeštajame amžiuje vokiečių astrologas ir matematikas Kepleris (angl.: *Johannes Kepler*) išanalizavo Koperniko (angl.: *Nicolaus Copernicus*) ir Brahe (angl.: *Tycho Brahe*) surinktus astrologinius duomenis, kad atskleistų anksčiau negirdėtą modelį: planetų skriejimo orbitos yra elipsės, kurių viename židinyje yra Saulė.

Astronominių duomenų analizė, leidžianti išsiaiškinti reikšmingus motyvus ir modelius, davė pradžią matematinių metodų kūrimui: Niutono-Gauso tiesinių lygčių sprendimų metodai (angl.: *Newton-Gauss algorithm*), Niutono visuotinės traukos dėsnis (angl.: *Isaac Newton*), Langražo polinominės interpoliacijos formulės (angl.: *Joseph-Louis Lagrange*), Laplaso (angl.: *Pierre-Simon Laplace*) mažiausio kvadrato glodinimo metodas. XIX a. ir ankstyvo XX a. metu buvo sukurta daug įvairių matematinių metodų duomenų analizei ir reikšmingų motyvų ieškojimui [1].

Skaitmeninių kompiuterių sukūrimas XX a. viduryje leido automatizuoti duomenų analizės metodus. Per pastaruosius pusę amžiaus spartus kompiuterinės galios progresavimas leido įgyvendinti tiesinių algebrinių duomenų analizės metodus, tokius kaip regresija ir principinė komponentinė analizė. Naujų analizės metodų integravimas kartu su vis augančia kompiuterio skaičiuojamąja galia, leido apdoroti vis didesnius kiekius duomenų. Taip palengvinant reikšmingų motyvų ieškojimą ir modelių kūrimą, kadangi neretai tiriamos sistemos, kurių struktūros ir modeliai yra sunkiai apibrėžiami [13].

Klasikinis tokios sudėtingos sistemos pavyzdys būtų kalbos atpažinimas. Šioje sistemoje surenkami duomenys yra girdimi žodžiai, jie analizėje apibrėžiami kaip „įvesties“ komponentas. „Išvestis“, šiuo atveju, būtų tie girdimi žodžiai užrašyti tekstu. Tokia sistema nebūtų sudėtinga, jei reiktų kompiuteriui užrašyti pavienius žodžius, tačiau, kuomet turi būti atsižvelgta į sakinių struktūrą, gramatiką, skyrybą ir kitus elementus, tai pasidaro itin keblu. Konkrečios veiksmų sekos užrašymas tokiam tikslui pasiekti, t. y. algoritmo tokiam modeliui sukurti išvedimas yra sudėtingas. Todėl tokioms užduotims spręsti yra pasitelkiami metodai, gebantys autonominiu būdu išmokti nustatyti ryšius tarp įvesties ir išvesties, t. y. apmokomi mašininio mokymo algoritmai [10].

Mašininis mokymas (angl.: *machine learning*) – tai duomenų analizės metodas, kuris leidžia automatizuoti analitinio modelio kūrimą. Šis metodas yra dirbtinio intelekto atšaka, paremta idėja, jog kompiuterio sistemos gali mokytis iš pateiktų duomenų, nustatyti juose reikšmingus motyvus ir priimti sprendimus su minimalia žmogaus intervencija [4], [47].

Populiariausia mašininio mokymo atšaka yra dirbtiniai neuroniniai tinklai. Šiame darbe nagrinėjamas mašininis mokymas dirbtiniais neuroniniais tinklais ir jų pritaikomumas biojutiklio charakteristikų nustatymui radus biojutiklio atsaką.

3.1. Dirbtiniai neuroniniai tinklai

Skaičiavimai yra vienas iš kelių dalykų, kuriuos skaitmeninis kompiuteris gali atlikti daug geriau už žmogų. Tačiau, nepaisant skaičių, žmogaus smegenys turi daug daugiau privalumų visose kitose srityse. Mes lengvai galime atpažinti veidą, net jei matome jį iš kampo, blogai apšviestame kambaryje, pilname daiktų. Mes lengvai suprantame kalbą, net, jei kalbantysis būtų nepažįsta-

mas ir kalbėtų triukšmingame kambaryje. Nors daug darbo atlikta kompiuterių tyrimuose, jie vis dar negali atlikti tokio aukšto lygio operacijų. Nepaisant visų skirtumų, pati įdomiausia žmogaus smegenų savybė ir pats didžiausias privalumas yra jų gebėjimas mokytis. Nereikalingas joks programavimas, ar programinės įrangos atnaujinimas, vien dėl to, kad sugalvojome, jog norime išmokti važiuoti dviračiu [27].

Žmogaus smegenys sugeba greitai apdoroti didelius, dažnai nenuoseklius, duomenų kiekius ir priimti atitinkamus sprendimus. Pilnai supratus smegenų veikimo ir mokymosi mechanizmą, dirbtinio intelekto, prilygstančiam žmogaus gebėjimams, sukūrimui tereiktų įgyvendinti šį mokymosi mechanizmą kompiuteryje. Nors neuromokslininkams nėra aiškus pilnas smegenų veikimo principas, esminiai smegenys sudarantys elementai yra žinomi.

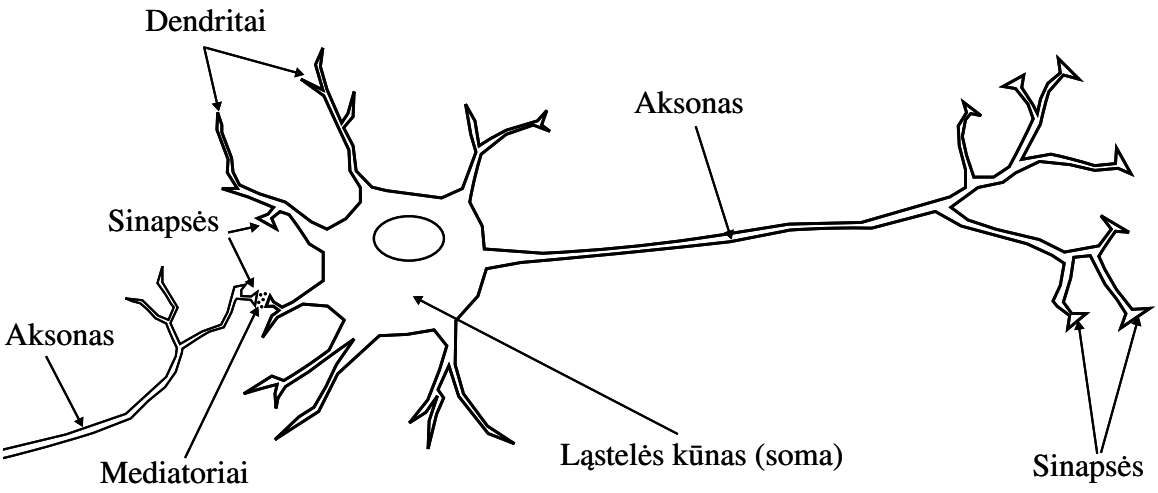
Vienas svarbiausių smegenų komponentų yra nervinės ląstelės (neuronai). Kaip ir kiekvienos gyvų organizmų ląstelės, neuronai komunikuoja tarpusavyje. Mokymąsis neuromokslininkų ir yra apibrėžiamas kaip neuronų tarpusavio komunikacija [21]. Tarpusavyje komunikuojančių neuronų visumą vadiname neuroniniu tinklu (angl.: *neural network*), o biologijos įkvėpti modeliuojami tinklai vadinami dirbtiniais neuroniniais tinklais (angl.: *artificial neural networks*).

Dirbtiniai neuroniniai tinklai yra pakankamai nauji skaičiavimo įrankiai, turintys platų panaudojimą, sprendžiant daugybę sudėtingų realaus pasaulio problemų. Pavyzdžiui, dirbtiniai neuroniniai tinklai taikomi kalbos atpažinime [18], botanikoje [29], finansų srityje [19]. Puikų informacijos apdorojimą nulemia dirbtinių neuroninių tinklų gebėjimas spręsti netiesiškus sąryšius, išlaikyti klaidų ir triukšmo toleranciją, ir atlikti duomenų generalizaciją [8].

3.2. Biologinis neuronas

Nervų sistema – tai labai specializuota organizmo sistema, palaikanti abipusį ryšį su aplinka, integruojanti ir koordinuojanti visas kitas organizmo funkcines sistemas, adaptuojanti organizmą prie kintančios aplinkos ir užtikrinanti jo gyvybingumą. Žmogaus nervų sistema yra skirstoma į centrinę nervų sistemą ir periferinę. Centrinė nervų sistema arba CNS yra toliau skirstoma į galvos ir nugaros smegenis. Galvos smegenys yra atsakingos už trumpalaikę-ilgalaikę atmintį, veiksmų planavimą, judesių kontrolę, kūno suvokimą aplinkoje, regą, klausą, mokymąsi, emocijas. Nugaros smegenys siejamos su raumenų judesiais, refleksais bei priėmimu aplinkos signalų, kurie yra perduodami galvos smegenims. Signalų priėmimas ir perdavimas vyksta per specialias nervų sistemos ląsteles vadinamas neuronais. Neuronas – tai pagrindinis struktūrinis ir funkcinis nervų sistemos vienetas (5 pav.).

Smegenis sudaro tankus 10^9 tarpusavyje sujungtų neuronų tinklas iš kurių kiekvienas vykdo biochemines reakcijas, leidžiančias priimti signalus, juos apdoroti ir perduoti į kitus neuronus. Nors neuronai būna labai įvairūs morfologiškai ir funkciškai, jų sandaros schema yra panaši. Kiekvieną neuroną sudaro trys pagrindinės dalys: soma (ląstelės kūnas), dendritai ir aksonas (5 pav.). Soma yra ląstelės kūnas, kuriame mes rasime ir kitoms ląstelėms būdingas organeles bei branduolį. Skirtingai nuo kitų žmogaus kūno ląstelių, nuo neuronų somos atsišakoja ilgos, plonos ataugos – dendritai ir aksonas. Dendritų skaičius ant somos gali būti įvairus; jie gali būti labai šakoti, tačiau niekados nėra labai nutolę nuo pagrindinės neurono kūno dalies – somos. Dendritų galai baigiasi tarpneuroninėmis jungtimis, kurias vadiname sinapsėmis. Per šias struktūras neuronai priima biocheminius signalus iš kitų neuronų. Neuronuose tą biocheminį signalą perduoda aksonai. Aksonas būna tik vienas ir daug ilgesnis už dendritines ataugas. Aksono galai išsišakoja ir, būtent, per tuos išsišakojimus yra perduodamas biocheminis signalas į kito neurono dendrito sinapses [2], [38].



5 pav. Biologinis neuronas.

Dendritų priimtas biocheminis signalas, t. y. elektrocheminis impulsas, pakeičia jonų balansą neurone, ir taip yra generuojamas naujas elektrocheminis impulsas, kuris sklinda išilgai aksonu. Aksono gale yra neuromediatorių pilnos pūslelės, kurios paveiktos elektrocheminio impulso, išleidžia tuos neuromediatorius į išorę. Ląstelių aplinkoje esantys neuromediatoriai prisijungia prie sinapsių, kurios yra dendritų galuose, ir aktyvuoja naują biocheminį signalą [17].

Biologinio neurono pagrindu yra paremti dirbtiniai neuroniniai tinklai. Nors dabartiniai dirbtiniai neuroniniai tinklai nepasiekė smegenų galimybių, tačiau yra nemažai uždavinių, kurie yra sunkiai suvokiami žmogui, bet lengvai išsprendžiami pasitelkiant tokius neuronų tinklus.

3.3. Dirbtinis neuronas

Kaip ir biologinis neuronas, dirbtiniai neuronai (6 pav.) turi įvestis (dendritus) ir išvestis (aksoną). Neurono įvesties vektorių $\boldsymbol{x}$ gali sudaryti bet koks skaičius n komponenčių: $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$. Tai analogiška biologiniam neuronui, kuris gali turėti įvairų skaičių dendritinių ataugų (įvesčių). Tuomet, varijuojant įvesties komponenčių verte ir kaip šios komponentės yra apdorojamos pačiame neurone, keičiasi išvesties y vertė. Neurono išvestis y gali būti tiek galutinis atsakymas, tiek įvestis į kitą dirbtinį neuroną.

Kaip ir minėta, vektoriaus $\boldsymbol{x}$ komponenčių vertės nėra vienintelis veiksnys darantis įtaką neurono išvesčiai y; įvestys yra transformuojamos neurone. Transformacija atliekama suskaičiavus įvesties vektoriaus $\boldsymbol{x}$ ir svorių vektoriaus $\boldsymbol{w}$ skaliarinę sandaugą $\boldsymbol{x} \cdot \boldsymbol{w}$ ir atimant papildomą slenkstinį parametrą (angl.: *bias*) [9]:

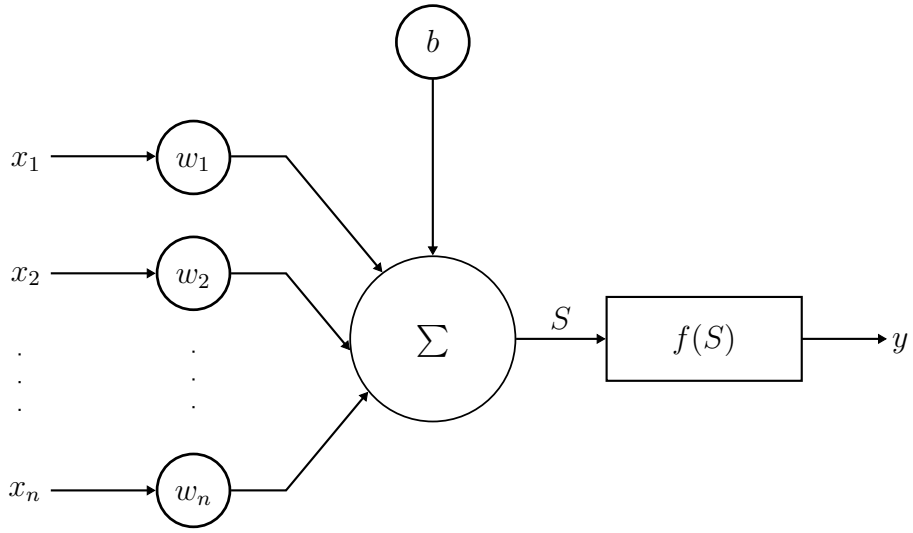
$$S = \boldsymbol{x} \cdot \boldsymbol{w} - b = \sum_{i=1}^{n} x_i w_i - b. \tag{3.1}$$

kur S – signalas, $\boldsymbol{x}$ – įvesties vektorius, $\boldsymbol{w}$ – svorių vektorius b – papildomas slenkstinis parametras.

Tuomet, prieš pateikiant išvestį į kitą neuroną, signalas S papildomai transformuojamas funkcijos (kurios kai kuriais atvejais gali ir nebūti) vadinamos aktyvacijos funkcija (angl.: *activation function*):

$$y = f(S), \tag{3.2}$$

kur y – išvestis, S – signalas, $f(\cdot)$ – aktyvacijos funkcija.



6 pav. Dirbtinis neuronas. Čia x_i – įvestys (i = 1, 2, …, n), w_i – įvesčių svoriai (i = 1, 2, …, n), b – slekstinis parametras, f – aktyvacijos funkcija, y – išvestis.

Yra įrodyta, kad, tinkamai parinkus svorius ir išdėsčius neuronus, galima atlikti universalius skaičiavimus [32]. Tokia dirbtinio neurono struktūra apytiksliai atspindi biologinio neurono struktūrą: jungtys modeliuoja aksonus ir dendritus, jungčių svoriai reprezentuoja sinapses ir jų efektyvumą, o papildomas slenkstinis parametras ir aktyvacijos funkcija procesus – vykstančius neurono ląstelės somoje [24].

3.4. Aktyvacijos funkcijos

Aktyvacijos funkcija yra paskutinis žingsnis transformuojantis įvestį S (6 pav.) ir nulemiantis neurono atsaką. Aktyvacijos funkcijos yra vienas iš reikalavimų, įgalinančių neuroninius tinklus atlikti universalaus funkcijos aproksimatoriaus vaidmenį [23]. Aktyvacijos funkcija reikalinga dėl kelių priežasčių.

Visu pirma, tiek įvestis į neuroną, tiek svoriai ir papildomas slenkstinis parametras gali būti bet kokie skaičiai iš realiosios skaičių aibės. Tai reiškia, kad neuronų signalas S (3.1 lygtis) gali įgyti skaitines vertes intervale $[-\infty, +\infty]$. Remiantis biologinio neurono principu, toks intervalas yra netinkamas, nes, jei ši išvestis yra įvestis į sekantį neuroną, pagal gautą skaitinę vertę pastarajam sunku įvertinti ar neuronas, iš kurio gautas signalas, buvo sužadintas ar ne. Problema sprendžiama pasirenkant aktyvacijos funkciją, pagal kurią nesudėtinga įvertinti neurono sužadinimo lygį, pavyzdžiui, bet kokiam S grąžinantį skaičių intervale $[0, 1]$.

Antra, neurono signalas S (3.1 lygtis) yra apibrėžtas tiesine priklausomybe (7a pav.). Tai reiškia, kad bet kokia tokių neuronų kombinacija taip pat bus tiesinės prigimties, o tai stipriai apriboja mokymosi ir pritaikymo galimybes. Problema sprendžiama pasirinkus netiesinės prigimties aktyvacijos funkciją, t. y. įvedami netiesiniai sąryšiai. Netiesiniai sąryšiai įgalina neuroninį modelį išmokti sudėtingų duomenų, tokių kaip garso, vaizdo, grafikos sąryšius.

Aktyvacijos funkcijos parenkamos pagal atliekamą užduotį. Dažniausiai naudojamos aktyvacijos funkcijos:

- **Slenksčio** funkcija. Slenksčio funkcija grąžina tik dvi vertes, t. y. 1 („taip“), kuomet pasiekiamas slenkstis, ir 0 („ne“) kuomet aktyvacijos slenkstis nėra pasiekiamas. Naudojama binarinėje klasifikacijoje. Tačiau retai naudojama sudėtingesniuose neuroniniuose tinkluose, nes šios funkcijos išvestinė yra lygi 0. Tai reiškia, kad gradientinis nusileidimas negali būti taikomas, ir svoriai, klaidos sklidimo atgal algoritmo būdu, negali būti pakoreguoti.

$$f(S) = \begin{cases} 0, & \text{kai} \quad S < 0, \\ 1, & \text{kai} \quad S \geq 0 \end{cases} \tag{3.3}$$

- **Sigmoidė** funkcija. Sigmoidė funkcija grąžina vertes intervale $[0, 1]$. Ši aktyvacijos funkcija dažnai taikoma klasifikacijos uždaviniuose, ypač paskutiniame giliojo neuroninio tinklo sluoksnyje, kuomet statistiškai reikia spėti įvesties panašumą į tam tikrą klasę. Tuomet išvestis 0 reiškia visišką nepanašumą, o 1 reiškia 100 % atitikimą.

$$f(S) = \frac{1}{1 + e^{-S}} \tag{3.4}$$

- **Hiperbolinė tangento** funkcija. Hiperbolinė tangento funkcija grąžina vertes intervale $[-1, 1]$. Pasižymi simetrija ties 0 ir didesniu gradientu nei sigmoidė funkcija [28].

$$f(S) = tanh(S) = \frac{2}{1 + e^{-2S}} - 1 \tag{3.5}$$

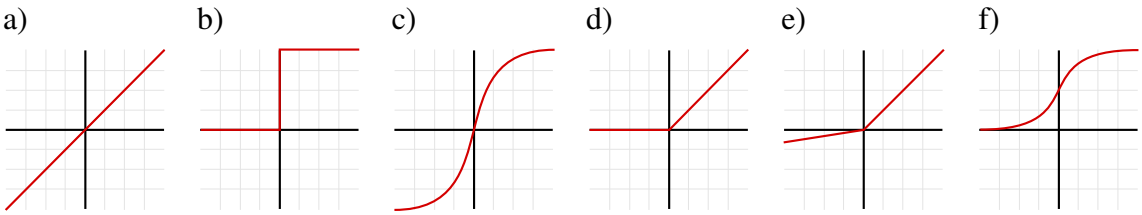
- **ReLu** (angl.: *rectified linear unit*) funkcija. ReLu funkcijos intervalas $[0, +\infty]$. ReLu aktyvacijos funkcija turi keletą pranašumų prieš sigmoidę ir tanh aktyvacijos funkcijas. Tai lemia platų ReLu panaudojimą. Neuroninis tinklas su ReLu aktyvacijos funkcijomis greičiau konverguoja, taip pat pasižymi mažesne "dingstančio gradiento" (angl.: *vanishing gradient*) tikimybe [26]. Tačiau ReLu funkcija turi neigiamą savybę – kad gradientas gali tapti 0. Tokiu atveju sakoma, kad neuronas „miršta“ t. y. nustoja reaguoti į įvesties ir paklaidos pokyčius.

$$f(S) = \begin{cases} 0, & kai \quad S < 0, \\ S, & kai \quad S \geq 0 \end{cases} \tag{3.6}$$

- **Parametrinė ReLu** (angl.: *parametric rectified linear unit*) funkcija. Tai ReLu variacija, siekianti išvengti gradiento tapimo 0. Šis efektas pasiekiamas horizontaliąją ReLu funkcijos

sritį padauginus iš daugiklio a, pavyzdžiui, $a = 0.001$ ir sukuriant šios srities verčių pasvirimą (7 pav.). Yra ir kitų ReLu variacijų, pavyzdžiui, ELu atveju horizontalioji sritis dauginama iš netiesinio daugiklio.

$$f(S) = \begin{cases} aS, & kai \quad S < 0, \\ S, & kai \quad S \geq 0 \end{cases} \tag{3.7}$$



7 pav. Grafinė aktyvavimo funkcijų reprezentacija: a) tiesinė, b) slenksčio, c) Tanh, d) ReLu, e) parametrinė ReLu, f) sigmoidė.

3.5. Netekties funkcija

Visi mašininio mokymo algoritmai remiasi į funkciją, vadinamą netekties funkcija (angl.: *loss function*). Netekties funkcija skaitiškai įvertina neatitikimą tarp gaunamų rezultatų (išvesties) ir rezultatų, kurių tikimasi. Algoritmo optimizacijos eigoje ieškoma tokių algoritmų parametrų, su kuriais netekties funkcijos vertė yra mažiausia.

Išskiriamos dvi pagrindinės netekties funkcijų kategorijos: klasifikacijos ir regresijos. Šiame darbe paminėsime tik pagrindines regresijos netekties funkcijas:

- **Vidutinė kvadratinė paklaida**:

$$E(x, y) = \frac{1}{n} \sum_{i=1}^{n} (x_i - y_i)^2, \tag{3.8}$$

kur n – duomenų taškų skaičius, x_i – tikimasi vertė, y_i – algoritmo suskaičiuota vertė.

- **Vidutinė absoliutinė paklaida**:

$$E(x, y) = \frac{1}{n} \sum_{i=1}^{n} |x_i - y_i| , \tag{3.9}$$

kur n – duomenų taškų skaičius, x_i – tikimasi vertė, y_i – algoritmo suskaičiuota vertė.

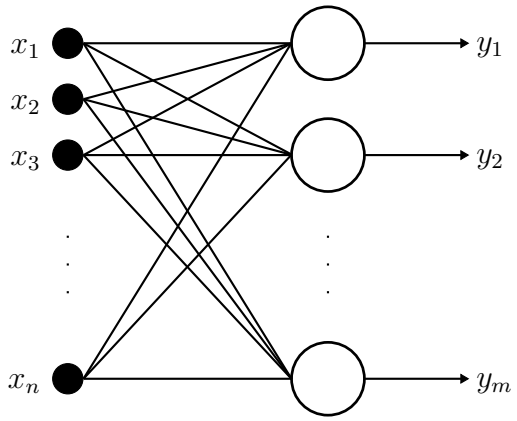
3.6. Vienasluoksniai perceptronai

Neuroniniai tinklai, sudaryti iš vieno ar daugiau tiesioginio sklidimo (angl.: *feed-forward*) neuronų (3.3 skyrius), yra vadinami perceptronais [44], [31]. Vienas paprasčiausių perceptronų

yra perceptronas turintis p neuronų, kurių kiekvienas sujungtas su n įvesčių (8 pav.). Tokiame perceptrone kiekviena išvestis y_i (i = 1, …, p) gali būti paskaičiuota pagal funkciją [12]:

$$y_i = f(a_i) = f\big(\sum_{k=1}^{n} w_{ik}x_k\big), \qquad i = 1, \ldots, p, \tag{3.10}$$

kur y_i – išvesties vektoriaus $\boldsymbol{y}$ i-toji komponentė, x_k – įvesties vektoriaus $\boldsymbol{x}$ k-toji komponentė, $f(\cdot)$ – aktyvavimo funkcija, w_{ik} – svorių vektoriaus $\boldsymbol{w}$ i, k-toji komponentė (įvesties x_k į y_i išvestį).



8 pav. Vienasluoksnis tiesioginio sklidimo neuroninis tinklas. x_i – įvestis (i = 1, . . . , n), n – įvesčių skaičius, y_j – išvestis (j = 1, . . . , m), m – išvesčių skaičius.

Pažymėsime duomenų rinkinį su kuriuo vyksta apmokymas, sudarytas iš m mėginių, $\boldsymbol{t} = (t_1, x_2, \ldots, t_m)$, o tokio rinkinio teisingus atsakymus t^q (q = 1, …, m).

Tuomet, norint suskaičiuoti y^q vertes, kurios artimiausios t^q vertėms, reikia bandyti įvairias svorių vektoriaus $\boldsymbol{w}$ komponenčių verčių konfigūracijas tol, kol y^q priartės prie t^q.

Svorių komponenčių verčių keitimas turi vykti tikslingai. Dėl šios priežasties yra įvesta klaidos matavimo funkcija – netektis (3.5 skyrius). Netekties nuo svorių vektoriaus funkciją pažymėsime $E(\boldsymbol{w})$.

Jei netekties funkcija $E(\boldsymbol{w})$ yra diferencijuojama, tai reiškia, kad galima nustatyti kaip reikia keisti svorius, kad sumažinti netektį, t. y. galima pasitelkti gradientinio optimizavimo algoritmus. Viena tokių netekties funkcijų yra kvadratinių paklaidų sumos funkcija:

$$E(\boldsymbol{w}) = \frac{1}{2}\sum_{q=1}^{m}\sum_{i=1}^{p}(y_i^q - t_i^q)^2 = \sum_{q=1}^{m} E^q(\boldsymbol{w}) \tag{3.11}$$

Čia y_i^q – neuroninio tinklo q-tosios duomenų komponentės išvestis iš i-tojo neurono, t_i^q – neuroninio tinklo q-tosios duomenų komponentės išvestis iš i-tojo neurono rezultatas, kurį norima gauti.

Pradinės vektoriaus $\boldsymbol{w}$ komponentės w_{ik} yra parenkamos atsitiktine tvarka normaliojo skirstinio ribose. Tuomet iteraciniu būdu priešinga gradiento kryptimi yra keičiamos svorių vertės:

$$w_{ik}(u+1) = w_{ik}(u) + \Delta w_{ik}(u) \tag{3.12}$$

$$\Delta w_{ik}(u) = -\eta \frac{\partial E(u)}{\partial w_{ik}} = -\eta \sum_{q=1}^{m} \frac{\partial E^q(u)}{\partial w_{ik}} = \sum_{q=1}^{m} \Delta w_{ik}^q(u) \tag{3.13}$$

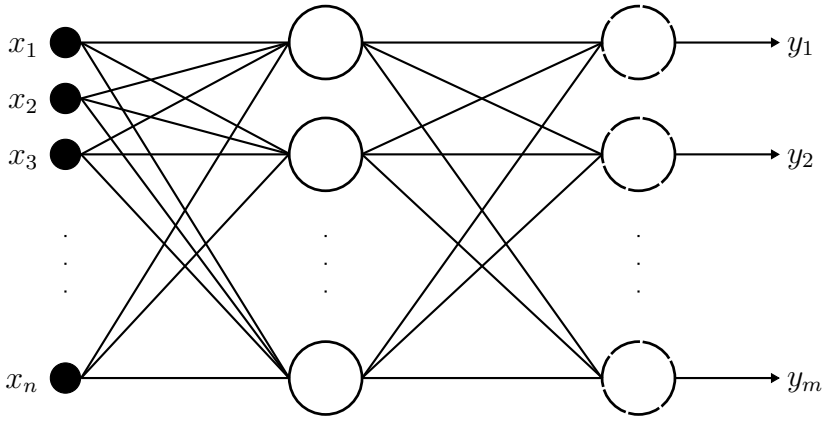
$$\Delta w_{ik}^q(u) = -\eta \frac{\partial E^q(u)}{\partial w_{ik}} \tag{3.14}$$

Čia η – mokymosi greitis (angl.: *learning rate*), u – iteracijos ciklo numeris, dar vadinamas epocha (angl.: *epoch*). Mokymosi greitis apibrėžia likutinės klaidos (angl. *residual error*) kiekį, propoguojamą į svorius.

1957 m. Frankas Rosenblatas (angl.: *Frank Rosenblatt*) sukonstravo mechaninį vienasluoksnį perceptroną ir pademonstravo jo gebėjimą klasifikuoti geometrines figūras: trikampius, keturkampių, apskritimus [43]. Nors tai buvo didelis žingsnis mašininio mokymo srityje, tokia neuroninio tinklo struktūra sugebėjo klasifikuoti tik tiesiškai atskiriamas objektų klases. 1969 m. Marvinas Minskis (angl.: *Marvin Lee Minsky*) ir Seimūras Papertas (angl.: *Seymour Papert*) parodė, kad toks perceptronas negali išmokti netiesinių sąryšių, pavyzdžiui: išmokti atlikti sudėties moduliu du operacijos, nes duomenų taškai gali būti atskirti tik tiese [37].

Šis architektūrinis apribojimas gali būti išspręstas išplečiant vieno sluoksnio architektūrą pridedant papildomų neuronų sluoksnių, t. y. konstruojant daugiasluoksnius perceptronus.

3.7. Daugiasluoksniai perceptronai



9 pav. Daugiasluoksnis neuroninis tinklas. x_i – įvestis (i = 1, ..., n), n – įvesčių skaičius, y_j – išvestis (j = 1, ..., m), m – išvesčių skaičius.

Neuroniniai tinklai, sudaryti iš daugiau nei vieno dirbtinių neuronų sluoksnio, sudarant ryšius viena kryptimi iš įvesčių į išvestis (angl.: *feed-forward*), vadinami daugiasluoksniais perceptronais (angl.: *multilayer perceptrons*; 9 pav.).

Tokiam daugiasluoksniam neuroniniam tinklui turinčiam L sluoksnių, kur l = 0, 1, ..., L, l-tajame sluoksnyje i-tojo neurono išvestis paskaičiuojama pagal 3.10 lygtį. Tuomet kiekvienas praeito sluoksnio ($l-1$) išėjimas y_k atitinka l sluoksnio įvestį x_k:

$$y_i = f_i(a_i) = f_i\big(\sum_{k=1}^{n(l-1)} w_{ik} y_k\big), \qquad i = 1, \ldots, n_l, \tag{3.15}$$

kur a_i – įvestis į i-tąjį neuroną, $f_i(\cdot)$ – i-tojo neurono aktyvacijos funkcija, $n(l-1)$ – neuronų skaičius $l-1$ sluoksnyje, kai l = 0 – įvesties sluoksnis, o $l = L$ – išvesties sluoksnis.

Išvesties neuronui pateikus klaidingą atsakymą, neįmanoma pasakyti, kokiu santykiu neuronai paslėptame sluoksnyje prisidėjo prie klaidingo atsakymo, todėl yra nežinoma kaip svoriai turi būti pakoreguoti. Daugiasluoksniai tiesioginio sklidimo neuroniniai tinklai negali būti apmokyti mažiausio vidutinio kvadrato radimo būdu (3.6 skyrius).

Praktikoje, rasti tinkamus svorius rankiniu būdu yra labai sudėtinga, šis uždavinys priskiriamas nedeterministinio polinominio laiko (NP, angl.: *nondeterministic polynomial time*) uždavinių klasei [14].

Daugiasluoksnis neuroninis tinklas gali išmokti gerų svorių kombinaciją algoritmo, vadinamo klaidos sklidimo atgal algoritmu, metu. Pirmą kartą šis metodas pademonstruotas sprendžiant 3.6 skyriuje paminėtą sudėties moduliu 2 uždavinį [45].

3.8. Klaidos sklidimo atgal algoritmas

Klaidos sklidimo atgal algoritmas (angl.: *backpropagation learning algorithm*) realizuoja gradientinio nusileidimo mokymo strategiją daugiasluoksniam tiesioginio sklidimo neuroniniam tinklui.

Daugiausluoksnis perceptronas mokomas kiekvienos epochos metu atliekant kelis žingsnius:

1. **Tiesioginio perdavimo** metodu apskaičiuojamos tinklo išėjimo sluoksnio vertės.
2. **Klaidos sklidimo atgal** metodu koreguojami visi tinklo svoriai, pradedant išėjimo sluoksniu ir pabaigiant įėjimo sluoksniu.

Pirmo algoritmo žingsnio metu iš mokymo duomenų paeiliui patenka į kiekvieną neuroną, kuriame atliekama duomenų transformacija. Rezultatas perduodamas į sekantį sluoksnį ir procedūra kartojama tol, kol pasiekiamas paskutinis sluoksnis, kuriame suskaičiuojamas tinklo išėjimo vektorius.

Antro žingsnio metu, pradedant išėjimo sluoksniu, yra suskaičiuojamas gautų verčių nuo rezultatų, kurių tikimasi, netektis. Jei $E^q(\boldsymbol{w})$ yra nelygi nuliui, tuomet gradiento nusileidimo būdu yra propoguojama svorių korekcija. Kiekvieno neurono, sujungto su k įėjimu $l-1$ sluoksnyje ir i išėjimu l sluoksnyje, svorio pokytis paskaičiuojamas lygtimi:

$$\Delta w_{ik}^q = -\eta \delta_i^q y_k^q \qquad \begin{matrix} i \in \text{sluoksniui } l \\ k \in \text{sluoksniui } l-1 \end{matrix}, \tag{3.16}$$

kur η – mokymo greitis, δ_i^q – lokalus gradientas, dar vadinamas delta, y_k^q – neuroninio tinklo q-tosios duomenų komponentės išvestis iš k-tojo neurono.

$$\delta_i^q = \begin{cases} \delta_i^q = f'(a_i^q)(y_i^q - d_i^q), & i \in \text{išvesties sluoksniui } L, \\ \delta_i^q = f'(a_i^q) \sum_{j=1}^{n(l+1)} w_{ji} \delta_j^q, & \begin{matrix} i \in \text{sluoksniui } l < L \\ j \in \text{sluoksniui } l+1 \end{matrix} \end{cases}. \tag{3.17}$$

4. Biojutiklio modelio, įskaitančio S, P, E ir EP kinetiką, skaitinis eksperimentas

Skaitiniu būdu sprendžiamas matematinis įgyvendinto amperometrinio biojutiklio modelis, šiuo atveju, yra vadinamas skaitiniu eksperimentu. Šio eksperimento metu yra nustatomas biojutiklio atsakas pasirinktose pradinėse modeliuojamos sistemos sąlygose.

Skaitinis eksperimentas įgyvendintas *Python* (versija 3.6) programavimo kalba. Panaudotos *Python* bibliotekos:

- *NumPy* biblioteka skirta kompaktiškesniam ir lakoniškesniam darbui su duomenų masyvais.
- *MatPlotLib* biblioteka skirta skaitinių eksperimentų duomenų grafinei vizualizacijai.

4.1. Skaitinio eksperimento sąlygos

Skaitinio eksperimento metu parinktos pradinės sąlygos, turinčios biocheminę prasmę, t. y. jos galimos *in vivo* arba *in vitro* eksperimentuose, ir keli *in silico* eksperimentui reikalingi parametrai [36]:

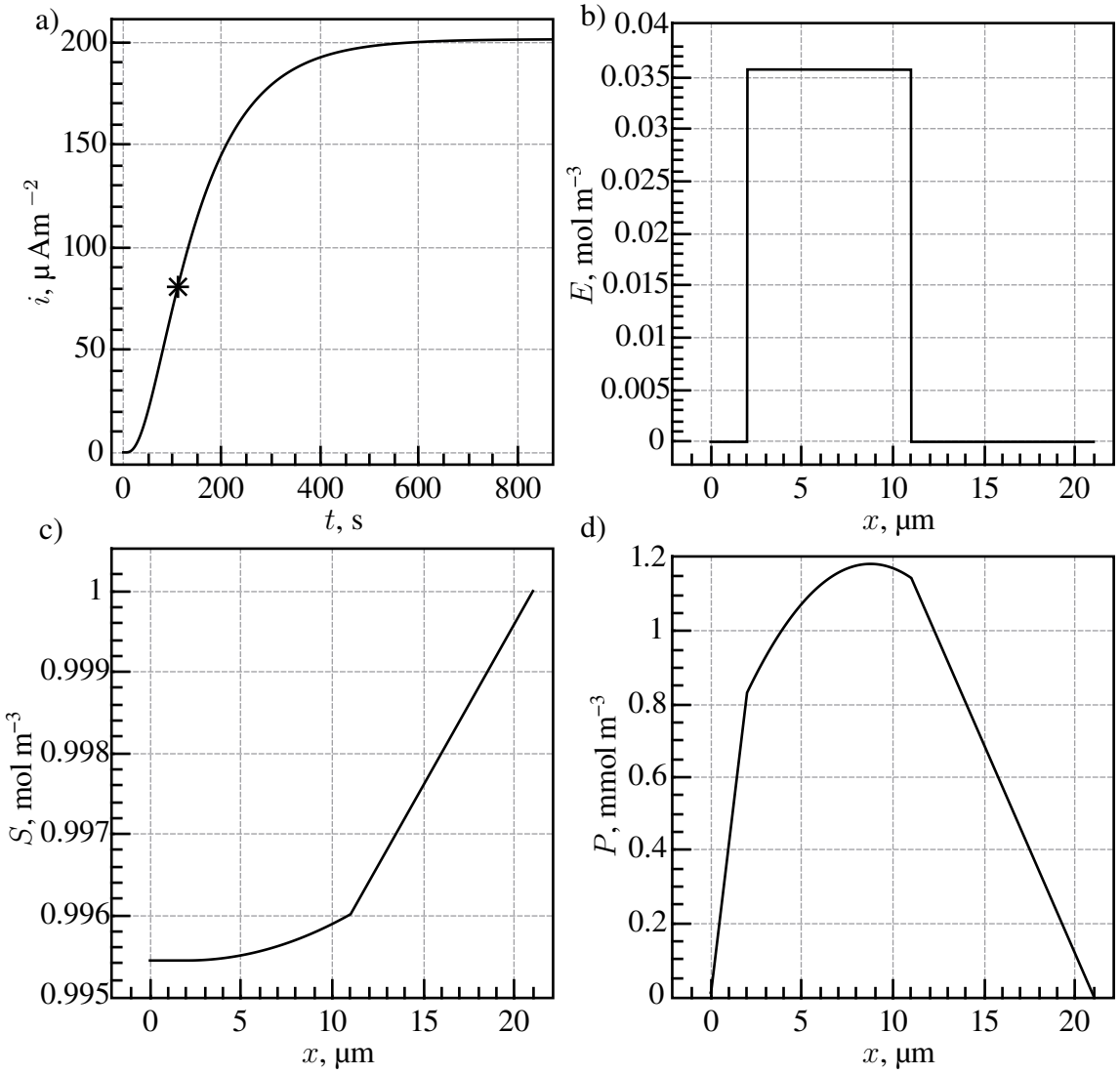
N = 1000, h = 0,021 µm, τ = 0,001 s, d_{m1} = 2 µm, d_e = 9 µm, d_{m2} = 10 µm, C_S = 0 s^{-1}, C_P = 0 s^{-1}, S_0 = 1 mol L, E_0 = 0,19 mol m^{-3}, k_{+1} = 0,0153846 $m^3mol^{-1}s^{-1}$, k_{-1} = 0,00153846 s^{-1}, k_{+3} = 0,002 $m^3mol^{-1}s^{-1}$, k_{-3} = 0,002 s^{-1}, n_e = 1, $D_{S_{m1}}$ = 6 µm$^2s^{-1}$, $D_{P_{m1}}$ = 5 µm$^2s^{-1}$, D_{S_e} = 22 µm$^2s^{-1}$, D_{P_e} = 20 µm$^2s^{-1}$, $D_{S_{m2}}$ = 7µm$^2s^{-1}$, $D_{P_{m2}}$ = 6µm$^2s^{-1}$.

Čia N – koordinatės kintamojo padalinimų skaičius, h – koordinatės kintamojo žingsnis, τ – laiko kintamojo žingsnis, d_{m1} – elektrodą gaubiančios membranos storis, d_e – fermento sluoksnio storis, d_{m2} – išorinės membranos storis, C_S – substrato degradacijos koeficientas, C_P – produkto degradacijos koeficientas, S_0 – pradinė substrato koncentracija, E_0 – pradinė fermento koncentracija, k_{+1}, k_{-1}, k_{+3}, k_{-3} – reakcijos intensyvumą apibrėžiantys koeficientai, n_e – ties biojutiklio elektrodu elektros krūvį pernešančių elektronų skaičius, $D_{S_{m1}}$ – substrato difuzijos koeficientas elektrodą gaubiančioje membranoje, $D_{P_{m1}}$ – produkto difuzijos koeficientas elektrodą gaubiančioje membranoje, D_{S_e} – substrato difuzijos koeficientas fermente, D_{P_e} – produkto difuzijos koeficientas fermente, $D_{S_{m2}}$ – substrato difuzijos koeficientas išorinėje membranoje, $D_{P_{m2}}$ – produkto difuzijos koeficientas išorinėje membranoje.

4.2. Skaitinis eksperimentas

Įvykdžius planuotą skaitinį eksperimentą, su 4.1 skyriuje aprašytomis pradinėmis sąlygomis, gautas biojutiklio atsakas I = 201,112 µA m^{-2}. Skaitinis eksperimentas sustabdytas pagal 2.2.6 skyriuje aprašytą kriterijų, t. y. biojutiklio atsako I augimui sumažėjus 10^3 kartų, lyginant su visos modeliavimo trukmės metu didžiausiu atsako augimu t_{did}. Šio eksperimento metu nustatytas intensyviausias biojutiklio srovės tankio augimas laiko momentu t_{did} = 80,68 s (10a pav.).

Gautas biojutiklio atsakas ir rezultatai yra identiški [36] tyrimo metodinėje medžiagoje pateiktiems rezultatams, išskyrus 10a pav. grafiko rezultatus. Nors buvo gautas teisingas biojutiklio atsakas I = 201,112 µA m^{-2}, jis buvo pasiektas per dvigubai didesnį laiko intervalą, todėl atitinkamai t_{did} vertė paklaidos ribose taip pat yra dvigubai didesnė. Tai galėjo atsitikti šio darbo metu padarius biojutiklio įgyvendinime klaidą, lemiančią tik tokį šalutinį efektą, nedarantį, autoriaus nuomone, įtakos galutinio rezultato korektiškumui; arba [36] tyrimo metodinėje medžiagoje padaryta spausdinimo klaida.



10 pav. a) Biojutiklio atsako $i(t)$ priklausomybė nuo laiko t. Žvaigždute pažymėtas laiko momentas t_{did} = 80,68 s. b) Laisvo fermento koncentracijos priklausomybė nuo koordinatės kintamojo x, kai $i(t) = I$. c) Substrato koncentracijos priklausomybė nuo koordinatės kintamojo x, kai $i(t) = I$. d) Produkto koncentracijos priklausomybė nuo koordinatės kintamojo x, kai $i(t) = I$.

4.3. Skaitinio eksperimento testavimas

Įgyvendinto skaitinio biojutiklio modelio pagal [36] tyrimo metodinę medžiagą korektiškumas ir rezultatų teisingumas įvertintas šiais būdais:

- Kadangi *in silico* modelis yra paremtas *in vitro* vykdomomis biocheminėmis reakcijomis, gautos vertės turi būti validžios biocheminiame kontekste:
  - S, P, E vertės, o taip pat ir biojutiklio atsakas, negali būti neigiami;
  - laisvo fermento koncentracija $E(x,t)$ negali viršyti pradinėmis sąlygomis pasirinktos fermento koncentracijos E_0;
  - substrato koncentracija $S(x,t)$ negali viršyti pradinėmis sąlygomis pasirinktos substrato koncentracijos S_0;
  - biojutiklio atsakas turi nuosekliai didėti ir artėti prie nuostoviosios būsenos.
- Tikrinama, ar išlaikomos modelio apibrėžtos kraštinės sąlygos. Tai buvo daroma vizualiai vertinant skaitinio eksperimento metu generuotus rezultatų grafikus.
- Lyginami [36] tyrimo metodinėje medžiagoje gautas galutinio biojutiklio atsako rezultatas ir, gauta vertė, kuomet biojutiklio atsakas augo staigiausiai su įgyvendintu biojutiklio skaitiniu modeliu.
- Lyginami [36] tyrimo metodinėje medžiagoje gauti atsako ir koncentracijų grafikai su įgyvendinto biojutiklio skaitinio modelio gautais grafikais.

5. Biojutiklio modelio, įskaitančio S, P, E ir EP kinetiką, skaitinių eksperimentų duomenų bazė

Įgyvendinto biojutiklio pradinių sąlygų analizei ir šių sąlygų atstatymo tyrimui atlikti, žinant biojutiklio atsaką, reikalingas atitinkamas skaitinių eksperimentų rinkinys. Tai yra, skaitiniai eksperimentai atlikti esant įvairioms pradinėms sąlygoms. Vykdant tokias analizes, susikaupia dideli duomenų kiekiai, kurie turi būti saugomi duomenų bazėje. Tai daroma didelio duomenų kiekio integralumui užtikrinti. Dėl šios priežasties vienas iš iškeltų magistrinio darbo uždavinių yra duomenų bazės sukūrimas, kaupiančios atliktų skaitinių eksperimentų pradines sąlygas (žr. 4.1 skyrių), bei šių eksperimentų metu nustatytą biojutiklio atsaką ir kitus tyrimui reikalingus parametrus.

5.1. Skaitinių eksperimentų duomenų rinkinys

Patogiam duomenų saugojimui, integralumo užtikrinimui ir greitoms skaitymo operacijoms pasirinkta *MySQL* reliacinė duomenų bazė. Į duomenų bazę įrašomos tiek skaitinio eksperimento pradinės sąlygos, tiek eksperimento metu nusistovėjęs biojutiklio atsakas.

Siekiant pagreitinti skaitinių eksperimentų rinkinio sudarymo greitį, pasirinkta duomenų bazės architektūra, leidžianti lygiagrečiai vykdyti skaitinius eksperimentus (11 pav.). Tai pasiekiama sudarius bendrą skaitinių eksperimentų užduočių eilę.

Skaitinių eksperimentų eilė kontroliuojama trijų pagrindinių veiksmų: naujo skaitinio eksperimento sukūrimas, skaitinio eksperimento vykdymas ir užbaigimas.

- **Skaitinio eksperimento sukūrimas**. Į bendrą skaitinių eksperimentų eilę įrašomas naujas skaitinis eksperimentas. Pažymima, kad šis eksperimentas yra dar neįvykdytas. Kiekvienoje naujoje skaitinio eksperimento užduotyje įrašomos: pradinės sąlygos, varijuojamo parametro kitimo sritis, bei kitimo žingsnis, kuris, priklausomai nuo tiriamos charakteristikos, gali būti skaičiuojamas kaip tiesinis žingsnis arba logaritminis.

- **Skaitinio eksperimento vykdymas ir užbaigimas**. FIFO (angl.: *first in first out*) principu kiekvienas skaitinio eksperimento vykdymo procesas prisiskiria skaitinio eksperimento užduotį ir ją pradeda vykdyti iš karto įrašydamas rezultatus į duomenų bazę. Pažymima, kad eksperimentas yra vykdomas. Skaitinio eksperimento vykdymo procesas, baigęs skaitinio eksperimento užduotį, įrašo vykdymo trukmę ir nustatytą biojutiklio atsaką, ir pažymi, kad eksperimentas yra įvykdytas.

Keletas papildomų tokios duomenų bazės architektūros pranašumų yra tai, kad vykdymo procesai be perstojo atlieka skaičiavimus, t. y. nereikia laukti vykdymo pabaigos, nes nauja užduotis priskiriama automatiškai. Nesibaigus visiems skaitiniams eksperimentams galima nagrinėti turimą duomenų sritį.

Paraleliai leidžiamų skaitinių eksperimentų vykdymo procesų kiekis pasirinktas atsižvelgiant į turimo asmeninio kompiuterių klasterio skaičiuojamąją galią.

Atliktų skaitinių eksperimentų rinkinį sudaro 770 tūkst. duomenų taškų. Šie duomenys panaudoti 6 ir 7 skyrių medžiagoje, todėl atskirai duomenys neaptariami. Prieinami Vilniaus Universiteto superkompiuterio skaičiavimo resursai nebuvo pasitelkti.

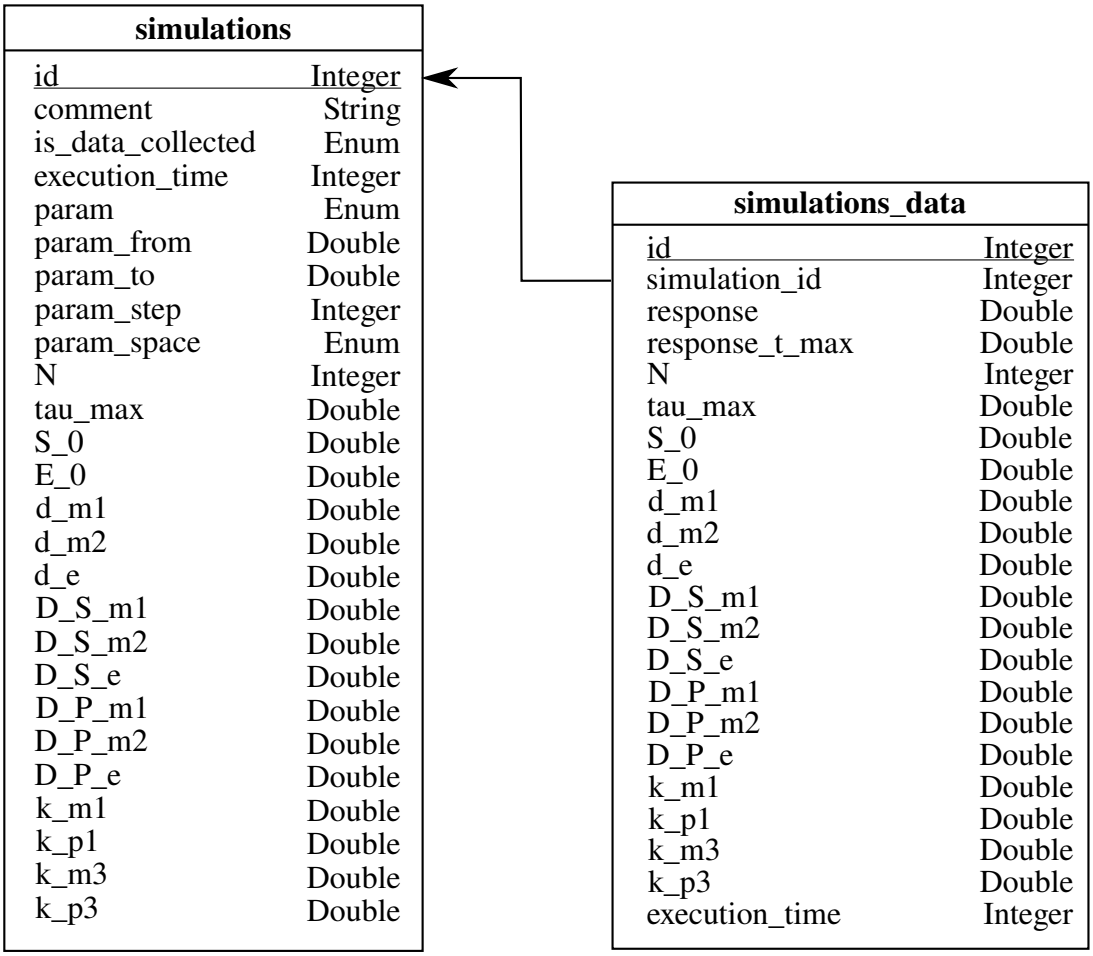
5.2. Skaitinių eksperimentų trukmės optimizavimas

Įgyvendinto matematinio biojutiklio modelio skaitiniam eksperimentui atlikti reikalingos ne tik biocheminės charakteristikos, bet ir *in silico* parametrai. N ir τ_{max} modelio algoritmo parametrai susiję tik su skaitinio modelio įgyvendinimu ir kontroliuoja erdvės ir laiko koordinačių kintamuosius. τ_{max} kontroliuoja tinklo laiko kintamojo t žingsnį, o N – koordinatės kintamojo x dalių skaičių. N ir τ_{max} parametrai daro įtaką skaitinio eksperimento vykdymo tikslumui ir trukmei: mažinant τ_{max} ir/arba didinant N, didėja modeliavimo trukmė ir tikslumas (ir priešingai).

Tinkamas N ir τ_{max} charakteristikų verčių parinkimas yra svarbus skaitinio eksperimento vykdymo trukmės ir skaičiavimo tikslumo optimizacijai. Kadangi, norint atlikti įgyvendinto biojutiklio modelio charakteristikų analizę, reikia atlikti nemažą kiekį skaitinių eksperimentų ir surinkti duomenų su įvairiais modelio parametrų deriniais, tai skaitiniai eksperimentai turi vykti pakankamu greičiu ir tikslumu. Eksperimentiniu būdu nustatyta, kad N = 100 ir τ_{max} = 1,0 s yra optimalios skaičiavimo greičio ir skaitinio modeliavimo rezultato tikslumo vertės (1 lentelė). Jos pasirinktos pagal biojutiklio atsako I vertę. Kadangi šie eksperimentai vykdyti esant 4.1 skyriuje aprašytoms pradinėmis sąlygomis (išskyrus N ir τ_{max}), tai biojutiklio atsakas I turi būti kuo artimesnis 201,112 μA m^{-2} vertei.

Modeliavimo trukmė (val.)	N	τ_{max}	I (μA m^{-2})	t_{did} (s)
0,1025	1000	0,1	201,112	80,9
0,0138	1000	1,0	201,112	83,0
0,0136	100	0,1	200,998	81,2
0,0014	**100**	**1,0**	**200,999**	**83,0**
0,0010	10	1,0	332,290	81,0

1 lentelė. Modeliavimo trukmės priklausomybė nuo koordinatės kintamojo dalių skaičiaus N ir tinklo laiko kintamojo žingsnio τ_{max} verčių. I – biojutiklio atsakas, t_{did} – laiko momentas, kuomet biojutiklio atsakas auga intensyviausiai. Skaitinio eksperimento trukmė ir tikslumas didėja mažinant τ_{max} ir/arba didinant N. Paryškintai pažymėtos N = 100 ir τ_{max} = 1,0 vertės pasirinktos tolimesnių eksperimentų vykdymui.



11 pav. UML bazės reliacinė schema. Lentelėje „simulations“ įrašomas skaitinių eksperimentų uždavinys. Atlikus skaitinį eksperimentą kiekviename uždavinio žingsnyje, į lentelę „simulations_data“ įrašoma kiekvieno skaitinio eksperimento pradinės sąlygos, gautas nusistovėjęs biojutiklio atsakas bei kiti parametrai. Vienas „simulations_data“ įrašas atitinka vieną atliktą skaitinį eksperimentą.

6. Biojutiklio atsako I priklausomybė nuo pradinių sąlygų

Biojutiklio atsakas I esant konkrečioms pradinėms sąlygoms randamas įvykdant vieną skaitinį eksperimentą, tačiau atvirkštinis radimas – sudėtingesnis, nes, nežinant biojutiklio atsako I kitimo tendencijų, gali tekti atlikti daugybę ilgai trunkančių skaitinių eksperimentų. Todėl nagrinėjame biojutiklio modelio atsako priklausomybę nuo įvairių pradinių sąlygų ir galimybę nustatyti keičiamą charakteristiką žinant tik biojutiklio atsaką.

Analizės metu naudojamas skaitinių eksperimentų duomenų rinkinys iš sudarytos biojutiklio atsako I priklausomybės nuo įvairių pradinių sąlygų skaitinių eksperimentų duomenų bazės.

Tyrimui naudotos ne tik anksčiau minėtos *NumPy* ir *MatPlotLib* bibliotekos (žr. 4 skyrių), bet ir papildomos *Python* bibliotekos:

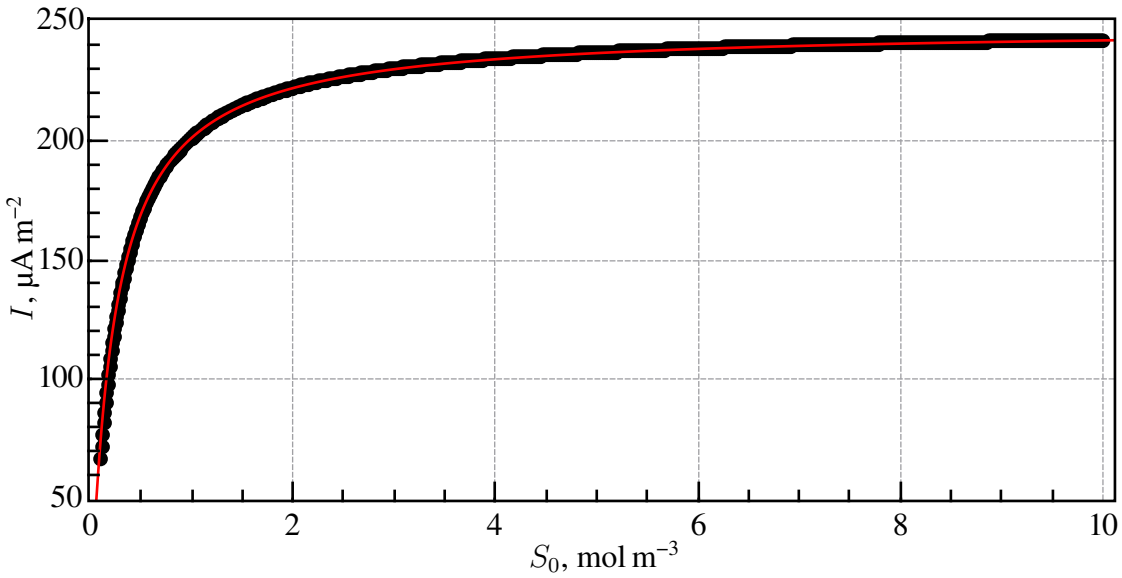
- *MySQL* biblioteka susisiekimui su MySQL duomenų baze (žr. 5 skyrių) Python aplinkoje.
- *Pandas* biblioteka lakoniškam darbui su duomenimis iš MySQL duomenų bazės.

6.1. Biojutiklio atsako I ir pradinės substrato koncentracijos S_0 sąryšis

Biojutiklio atsako I priklausomybės nuo substrato pradinės koncentracijos S_0 įvertinimui, atliktas skaitinis modeliavimas keičiant substrato koncentraciją intervale nuo 0,1 mol m^{-3} iki 10 mol m^{-3} žingsniu 0,01 mol m^{-3} (12 pav.). Gauta kreivė atkartoja [36] tyrime gautus rezultatus. Minėtame tyrime biojutiklio atsako priklausomybė nuo substrato koncentracijos aproksimuota formule:

$$I(S_0) \approx \frac{1}{\frac{K}{S_0} + M}, \tag{6.1}$$

kur M, K – koeficientai, nustatomi glodinant (angl. *fitting*) kreivę, S_0 – pradinė substrato koncentracija. 12 pav. raudona linija atitinka 6.1 lygtį, kai K = 942,37599 mol m^3 A^{-1}, M = 4042,97 m^2 A^{-1}, nustatyta mažiausio kvadrato glodinimo metodu.



12 pav. Biojutiklio atsako I priklausomybė nuo substrato koncentracijos S_0 intervale nuo 0,1 iki 10,0 mol m^{-3}. Juodi apskritimai – skaitinio modeliavimo metu gautos priklausomybės. Raudona linija – aproksimacija 6.2 lygtimi.

Šiame darbe daroma įžvalga, kad 6.1 lygtis gali būti užrašyta kitokia forma, pažymėjus $K = K_M M$. K_M priskiriamas pagal biocheminę prasmę (1.5 lygtis):

$$I(S_0) \approx \frac{1}{\frac{K_M M}{S_0} + M}, \tag{6.2}$$

kur K_M – Michaelis-Menten reakcijos greičio konstanta, M – koeficientas, nustatomas glodinant kreivę, S_0 – substrato koncentracija, kai $K_M = \frac{k_{p3} + k_{m1}}{k_{p1}}$ = 0,23 mol m^{-3}, M = 4042,97 m^2 A^{-1}. Taip apskaičiuota K_M = 0,23 mol m^{-3} vertė yra panaši į apskaičiuojamą K_M = 0,233 mol m^{-3} vertę glodinant kreivę. Tokiu atveju, išvengiama glodinamo K koeficiento įvedimo.

Nustačius K_M pagal biocheminę prasmę ir glodinus M 6.2 lygtį, pradinės substrato koncentracijos priklausomybė nuo biojutiklio atsako išreiškiama formule:

$$S_0(I) \approx \frac{I K_M M}{1 - IM}. \tag{6.3}$$

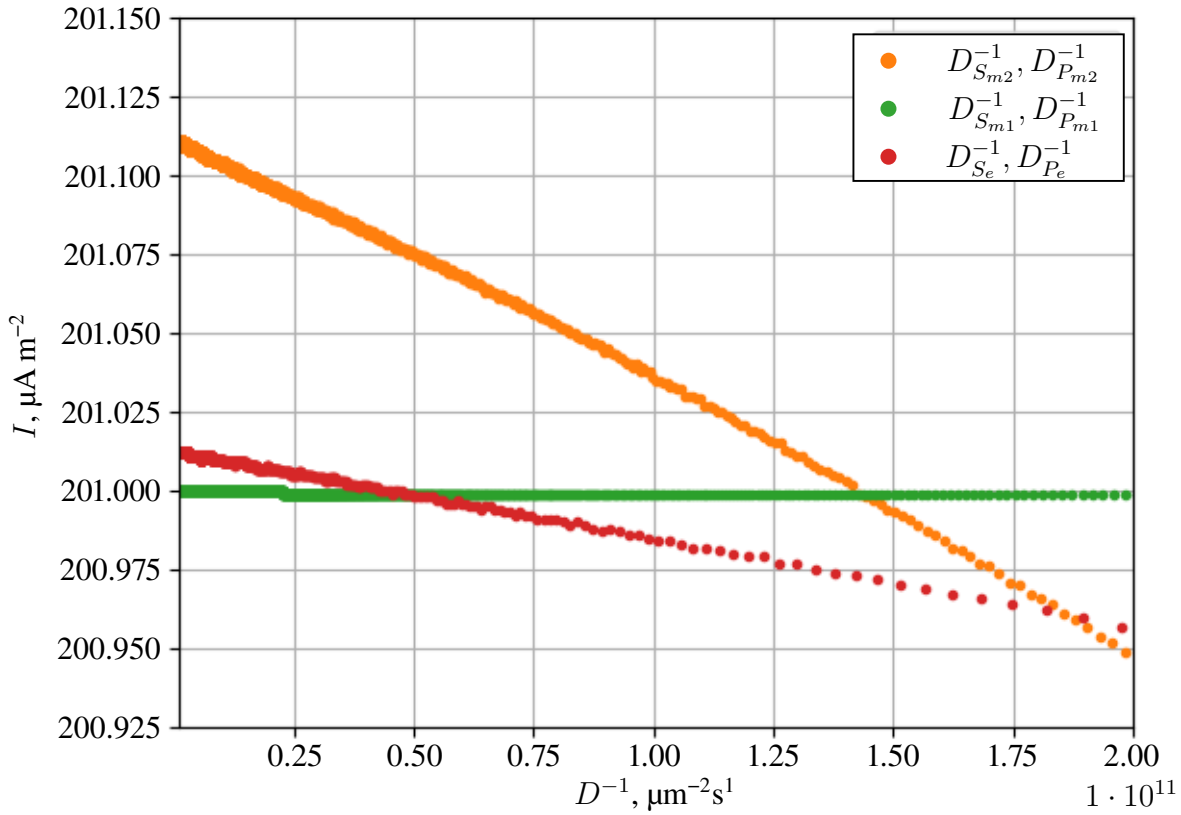
Čia S_0 – pradinė substrato koncentracija, I – biojutiklio atsakas, K_M – Michaelis-Menten reakcijos greičio konstanta, M glodintas koeficientas lygus 4042,97 m^2 A^{-1}. Su minėta M verte 6.3 lygtis tinka I nustatymui esant įvairioms S_0 koncentracijoms, kuomet kitos charakteristikos atitinka 2.1.1 skyriuje aprašytas pradines sąlygas.

6.2. Biojutiklio atsako I ir difuzijos konstantų $D_{S_{m2}}$, $D_{S_{m1}}$, D_{S_e}, $D_{P_{m2}}$, $D_{P_{m1}}$, $D_{P_{me}}$ sąryšis

Biologinėse sistemose difuzijos konstantos paprastai svyruoja nuo 10^{-12} m^{2}s^{-1} iki 10^{-9} m^{2}s^{-1}. Kadangi nagrinėjama modelinė biocheminė sistema, paremta *in vivo* ir *in vitro* eksperimentų charakteristikomis, yra tikslinga tirti biojutiklio atsako priklausomybę nuo difuzijų konstantų *in vivo* biologinių sistemų difuzijos verčių intervale [36].

Vykdyti skaitiniai eksperimentai parodė, kad mažėjant difuzijos koeficientui, biojutiklio atsakas silpnėja. Tokie *in silico* pastebėjimai neprieštarauja *in vitro* vykdomiems eksperimentams, nes molekulių tarpusavio maišymosi intensyvumas lemia galinčių reaguoti tarpusavyje molekulių susidūrimo tikimybę. O tai reiškia, kad elektrochemiškai aktyvaus produkto susidarymo kiekiui mažėjant, biojutiklio atsakas taip pat mažėja.

Difuzijos koeficientai didžiausią įtaką daro eksperimento eigai, kaip biojutiklio atsakas yra pasiekiamas, bet ne biojutiklio atsako galutinei skaitinei vertei [36]. Iš 13 pav. matyti, kad intervale nuo $1 \cdot 10^{-9}$ m^{2}s^{-1} iki $5 \cdot 10^{-12}$ m^{2}s^{-1} difuzijos koeficientai kinta mažiau nei per 0,1 μm^{-2}s^1. Dėl nedidelio poveikio biojutiklio atsakui šiame darbe difuzijos koeficientai nebuvo plačiau nagrinėjami.



13 pav. Atsako priklausomybė nuo difuzijos konstantų intervale nuo $1 \cdot 10^{-9}$ m^{2}s^{-1} iki $5 \cdot 10^{-12}$ m^{2}s^{-1}. I – biojutiklio atsakas, D – difuzijos koeficientas. Dėl geresnio vizualumo abscisių ašyje pateikiamos atvirkštinės difuzijos konstantų vertės.

6.3. Biojutiklio atsako I ir reakcijos greičio konstantų k_{m1}, k_{m3}, k_{p1}, k_{p3} sąryšis

Nagrinėjamos modelinės sistemos tiesioginių reakcijų greičio konstantos k_{p1} ir k_{p3} didina biojutiklio atsaką, o atvirkštinės k_{m1} ir k_{m3} mažina (14 pav.). Pagal 14 paveikslą galime matyti, jog: didžiausią įtaką reakcijos greičiui daro k_{p3} reakcijos greičio konstanta; biojutiklio atsakas nuo k_{m1}, k_{p1}, k_{p3} priklauso netiesiškai, o nuo k_{m3} priklauso tiesiškai. Todėl biojutiklio atsako priklausomybė nuo k_{m1}, k_{p1}, k_{p3} reakcijos greičio konstantų aproksimuotos hiperboline formule:

$$I(k) = \frac{1}{\frac{C_1}{k + C_2}}, \tag{6.4}$$

kur k – nagrinėjama reakcijos greičio konstanta, o C_1, C_2 – glodinamos konstantos. O priklausomybė nuo k_{m3} reakcijos greičio konstantos aproksimuota tiesine formule (2 lentelė):

$$I(k) = C_1 k + C_2, \tag{6.5}$$

kur k – nagrinėjama reakcijos greičio konstanta, o C_1, C_2 – glodinamos konstantos.

2 lentelėje pateikiamos glodinimo mažiausio kvadrato metodu apskaičiuotos 6.5 ir 6.4 lygčių konstantų vertės.

k	C_1	C_2	glodinimo formulė
k_{p3}	8,69	546,84	hiperbolinė
k_{p1}	14,74	4039,5	hiperbolinė
k_{m3}	-20,66	200,76	tiesinė
k_{m1}	263173,59	4571,81	hiperbolinė

2 lentelė. Greičio konstantų vertės tiesinei (6.5 lygtis) ir hiperbolinei (6.4 lygtis) glodinimo formulėms. Čia C_1, C_2 – glodintų kreivių lygčių konstantos.

Remiantis 6.4 lygtimi išvedama lygtis, apibrėžianti greičio konstantų k_{m1}, k_{p1}, k_{p3} priklausomybę nuo biojutiklio atsako:

$$k(I) = IC_1 - C_2, \tag{6.6}$$

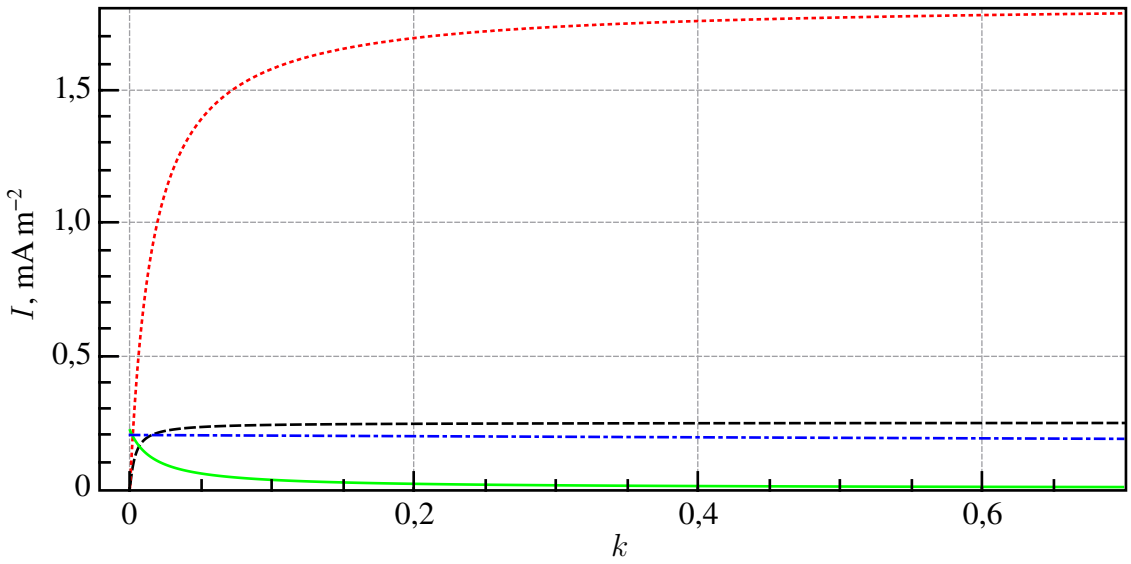
kur I – biojutiklio atsakas (A m^{-2}), C_1, C_2 – glodintos hiperbolinės lygties konstantos.

O iš 6.5 lygties greičio konstantos priklausomybę k_{m3} nuo biojutiklio atsako:

$$k(I) = \frac{I - C_2}{C_1}, \tag{6.7}$$

kur I – biojutiklio atsakas (A m^{-2}), C_1, C_2 – glodintos tiesinės lygties konstantos.

Su atitinkamomis C_1 ir C_2 vertėmis (žr. 2 lentelę) 6.6 ir 6.7 lygtys tinka I nustatymui esant įvairioms pasirinktoms reakcijos greičio konstantų vertėms, kuomet kitos charakteristikos atitinka 2.1.1 skyriuje aprašytas pradines sąlygas.



14 pav. Biojutiklio atsako priklausomybė nuo reakcijos greičio konstantų. Čia I – biojutiklio atsakas, raudona taškinė kreivė – reakcijos greičio konstanta k_{p3}, juoda brūkšninė kreivė – reakcijos greičio konstanta k_{p1}, mėlyna brūkšninė-taškuota tiesė – reakcijos greičio konstanta k_{m3}, žalia ištisinė kreivė – reakcijos greičio konstanta k_{m1}.

7. Biojutiklio atsako I priklausomybė nuo reakcijos greičio konstantų k_{m1}, k_{p1}, k_{p3}

Nežinant biojutiklio atsako I kitimo tendencijų, gali tekti atlikti daugybę ilgai trunkančių skaitinių eksperimentų. Šią problemą sprendžiame 6 skyriuje, kuomet parodome, kad pageidautinas biojutiklio atsakas I gali būti surastas fiksuojant visas pradines sąlygas išskyrus vieną charakteristiką, kuri gali būti išskaičiuota pagal 6 skyriuje apibrėžtas formules. Tačiau toks problemos sprendimas *in vitro* sistemose turi labai ribotą pritaikymą, nes prisirišama prie specifinių sistemos pradinių sąlygų. Dėl šios priežasties nagrinėjame galimybę nustatyti daugiau nei vieną varijuojamą charakteristiką. Kadangi reakcijos greičio konstantos daro didžiausią įtaką biojutiklio atsako I vertei, tai nagrinėjame būtent šių charakteristikų kitimus.

Be 4 skyriaus įžangoje jau paminėtų *NumPy* ir *MatPlotLib* ir 6 skyriaus įžangoje paminėtų *MySQL* ir *Pandas* bibliotekų, šiame skyriuje naudotos *Python* bibliotekos:

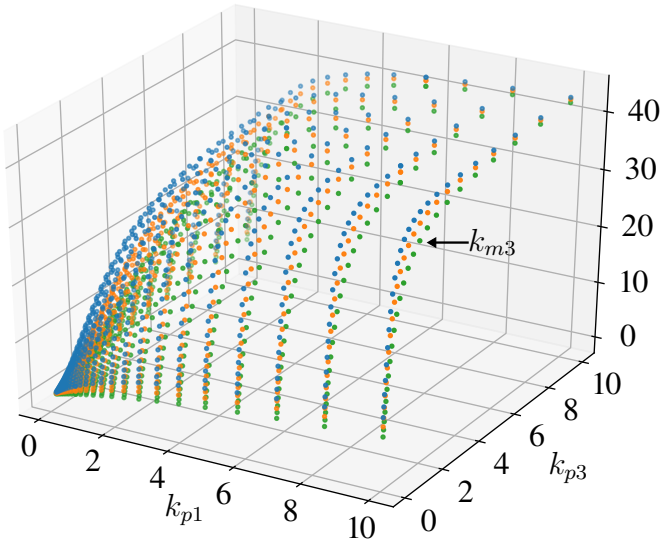
- *TensorFlow* biblioteka daugiasluoksnio neuroninio tinklo sudarymui ir apmokymui.
- *Tensorboard* biblioteka daugiasluoksnio neuroninio tinklo mokymo eigos ir netekties funkcijos monitoringui.

7.1. Biojutiklio atsako I priklausomybės nuo reakcijos greičio konstantų k_{m1}, k_{p1}, k_{p3} duomenų rinkinys

Biojutiklio atsako I priklausomybės nuo kelių reakcijos greičio konstantų k_{m1}, k_{p1}, k_{p3} analizei atlikti sudarytas duomenų rinkinys. Šiame duomenų rinkinyje k_{m1}, k_{p1}, k_{p3} charakteristikos keičiamos ne atsitiktinai, o pasirenkamos vertės iš intervalo srities nuo $1 \cdot 10^{-5}$ iki $1 \cdot 10^{1}$, aprėpiantį platų biojutiklio atsako I intervalą. Remiantis 6.3 skyriaus tyrimu, kuomet pastebėta, kad didėjant reakcijos greičiui mažėja jo daroma įtaka biojutiklio atsako kitimo intensyvumui, pasirinktas logaritminis, o ne tiesinis, žingsnis. Kiekviename intervalo žingsnyje analogiškai yra skaičiuojama biojutiklio atsako priklausomybė kitoms reakcijos greičio konstantoms. Tokio duomenų rinkinio statistinės duomenų savybės pateiktos 3 lentelėje, o duomenų sklaidos grafikas (angl.: *scatter plot*) 15 paveiksle.

statistinis įvertis	I	k_{p1}	k_{p3}	k_{m1}
mažiausia vertė	$1{,}23569 \cdot 10^{-12}$	$1{,}0 \cdot 10^{-5}$	$1{,}0 \cdot 10^{-5}$	$1{,}0 \cdot 10^{-5}$
apatinis kvartilis	$1{,}79735 \cdot 10^{-6}$	0,000288	0,000288	0,000288
mediana	$2{,}85484 \cdot 10^{-5}$	0,010000	0,010000	0,010000
viršutinis kvartilis	$7{,}25249 \cdot 10^{-4}$	0,347169	0,347169	0,347169
didžiausia vertė	$4{,}34142 \cdot 10^{-2}$	10,0	10,0	10,0

3 lentelė. Biojutiklio atsako I priklausomybės nuo reakcijos greičio konstantų k_{m1}, k_{p1}, k_{p3} duomenų rinkinio statistinė analizė.



15 pav. Biojutiklio atsako I priklausomybė nuo k_{m1}, k_{p1}, k_{p3} reakcijos greičio konstantų. Aiškesnei duomenų sklaidos vizualizacijai, paveiksle k_{p3} pateikiamas be intervalo $[0, 1]$ srities, o priklausomybė nuo k_{m3} atvaizduojama trijose vertėse skirtingomis spalvomis. Žalia spalva – k_{p3} = 10, oranžine – k_{p3} = 0,01, mėlyna – $k_{p3} = 1{,}0 \cdot 10^{-5}$.

7.2. Biojutiklio atsako I priklausomybės nuo reakcijos greičio konstantų k_{m1}, k_{p1}, k_{p3} duomenų rinkinio analizė

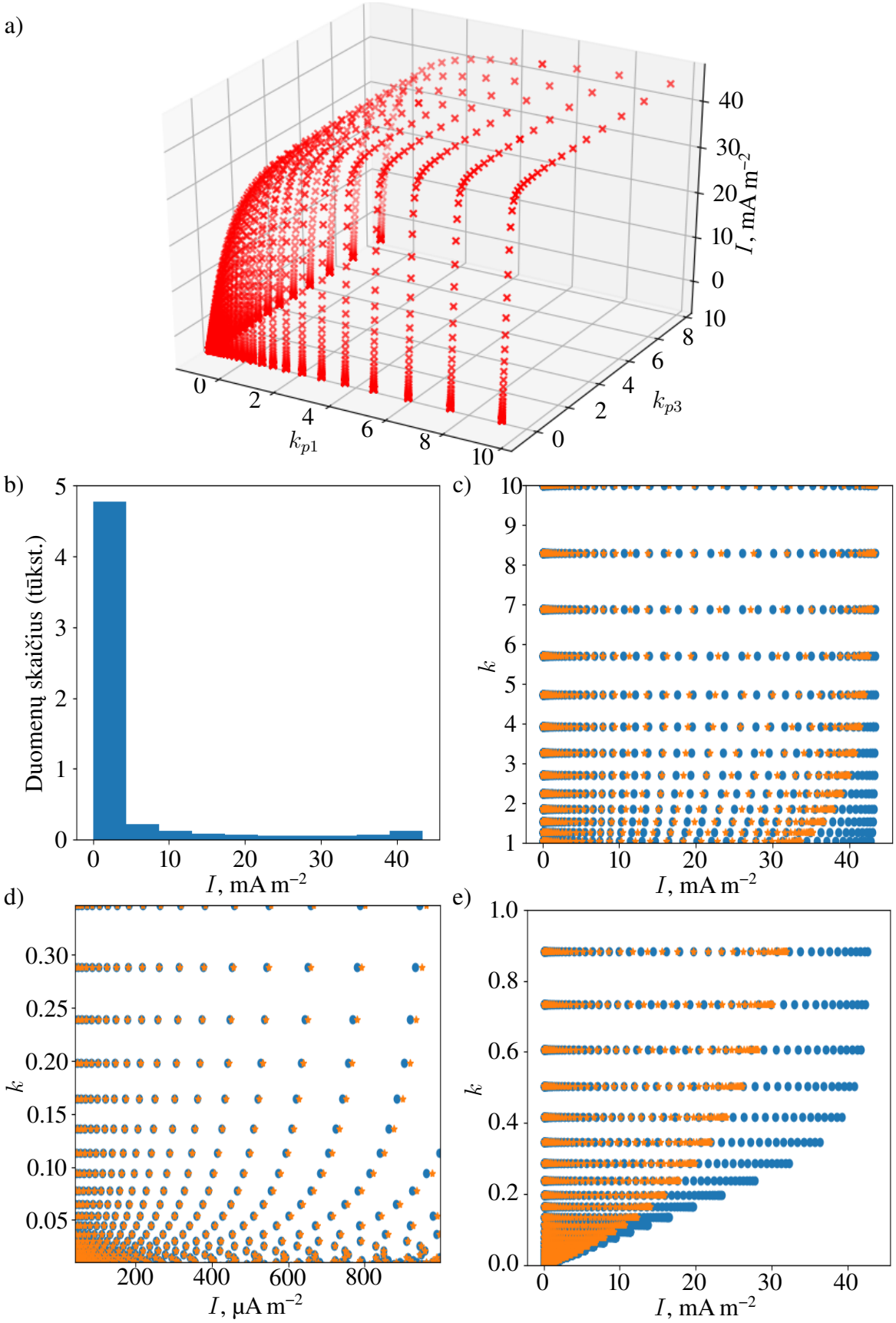
Bendra biojutiklio atsako I kitimo tendencija matoma iš 3 lentelės statistinių duomenų ir 15 paveikslo. Tai yra, didėjant greičio konstantų vertėms, didėja ir biojutiklio atsako vertė. Siekiant detaliau išnagrinėti duomenų rinkinį, fiksuojama viena iš reakcijos greičio konstantų. Šios analizės atveju nutarta fiksuoti atvirkštinę reakcijos greičio konstantą k_{m1} ir nagrinėti tiesioginės reakcijos greičio konstantų k_{p1}, k_{p3} kitimus.

16a paveikslas atvaizduoja reakcijos greičio konstantų k_{p1}, k_{p3} verčių priklausomybę nuo biojutiklio atsako I, esant fiksuotai reakcijos greičio konstantos k_{m1} vertei, t. y. k_{m1} = 0,0145. Iš 16a, 16c, bei 16e paveikslų galima spręsti, kad didėjant reakcijos greičio konstantoms k_{p1}, k_{p3}, biojutiklio atsako I didėjimo intensyvumas mažėja. Tokią tendenciją taip pat galima stebėti ir anksčiau nagrinėtu atveju, t. y. 6.3 skyriuje 14 paveiksle, kuomet didėjant reakcijos greičio konstantoms priklausomybė tampa beveik tiesiška.

Iš paminėtų pastebėjimų galima daryti išvadą, kad tokiame duomenų rinkinyje yra atvejų, kuomet panaši biojutiklio atsako I vertė gali būti aprašyta ne vienos, o kelių reakcijos greičio konstantų kombinacijų. Dėl šios priežasties, siekiant daugiau sužinoti apie nagrinėjamą modelinę sistemą, yra tiriami keli greičio konstantų atstatymo atvejai:

- 7.3 skyriuje nagrinėjama 16d paveiksle pavaizduota biojutiklio atsako I priklausomybės nuo dviejų reakcijos greičio konstantų duomenų sritis. Šioje duomenų srityje gilusis neuroninis tinklas apmokomas nuspėti vienos reakcijos greičio konstantos vertę, kuomet tinklui kaip įvestis paduodamos kitos reakcijos greičio konstantos bei biojutiklio atsako vertės.

- 7.4 skyriuje nagrinėjamame 16a paveiksle pavaizduota biojutiklio atsako I priklausomybės sritis nuo dviejų reakcijos greičio konstantų duomenų. Šioje duomenų srityje sprendžiamas regresijos uždavinys, kuomet glodinimo būdu bandoma atrasti formulių, aprašančių panašią į tiriamų duomenų topologiją, koeficientus.



16 pav. a) Reakcijos greičio konstantų k_{p1}, k_{p3} priklausomybė nuo biojutiklio atsako I, esant fiksuotai reakcijos greičio konstantos $k_{m1} = 0{,}0145$ vertei. b) Biojutiklio atsako I verčių pasiskirstymo histograma. c) Biojutiklio atsako I priklausomybė (intervale $[1, 10]$) nuo reakcijos greičio konstantų k_{p1} ir k_{p3}. d) Biojutiklio atsako I priklausomybė nuo reakcijos greičio konstantų k_{p1} ir k_{p3} apatinio ir viršutinio reakcijos greičio konstantų ir biojutiklio atsako kvartilių srityje. e) Biojutiklio atsako I priklausomybė (intervale $[0, 1]$) nuo reakcijos greičio konstantų k_{p1} ir k_{p3}. Čia oranžinės žvaigždutės – k_{p1}, mėlyni apskritimai – k_{p3}.

7.3. Reakcijos greičio konstantų k_{p1}, k_{p3} nustatymas duomenų rinkinio poaibyje

16a paveiksle pavaizduotame duomenų rinkinyje yra atvejų, kuomet panaši biojutiklio atsako I vertė gali būti aprašyta ne vienos, o kelių reakcijos greičio konstantų kombinacijų. Dėl šios priežasties nagrinėjamas 16d paveiksle pavaizduoto duomenų rinkinio poaibis, t. y. apatinio ir viršutinio reakcijos greičio konstantų ir biojutiklio atsako kvartilių sritis. Šioje duomenų srityje gilusis neuroninis tinklas apmokomas nuspėti vienos reakcijos greičio konstantos vertę (išvestis), tinklui žinant kitos reakcijos greičio konstantos bei biojutiklio atsako vertes (įvestys). Modeliuojami du variantai: kuomet išvestis yra k_{p1}, o įvestys – I, k_{p3} ir kuomet išvestis – k_{p3}, o įvestys – I, k_{p1}.

Kadangi nagrinėjami duomenys yra plačiame verčių intervale, prieš paduodant duomenis mokymui, jie transformuoti į intervalą, atitinkantį normalinės distribucijos verčių sritį:

$$d_i = \frac{d_i - d_{vid}}{d_{std}}, \tag{7.1}$$

kur d – duomenų rinkinio I, k_{p1} arba k_{p3} duomenų eilutė, o d_{vid} – šios duomenų eilutės vidurkis, o d_{std} – šios duomenų eilutės standartinis nuokrypis.

Mokymo eigoje gilusis neuroninis tinklas gali išmokti duomenų padavimo į neuroninį tinklą artefaktų, todėl siekiant to išvengti – duomenų eilutės išmaišytos atsitiktine tvarka. Tuomet duomenų eilutės padalintos į du duomenų rinkinius: pirmą, skirtą giliojo neuroninio tinklo apmokymui, sudarytą iš 80 % visų duomenų, ir antrą, skirtą apmokyto modelio testavimui, sudarytą iš likusių 20 % duomenų. Taip pat duomenų eilutės maišomos prieš paduodant duomenų pavyzdžius (angl.: *batch*) į neuroninį tinklą kiekvienoje mokymo iteracijoje.

Išbandytos įvairios giliojo neuroninio tinklo struktūrų kombinacijos, skirtingas sluoksnių ir neuronų kiekis, skirtingos aktyvacijos funkcijos, bei mokymo greičio, epochų ir duomenų pavyzdžių skaičiaus epochoje hiperparametrai (angl.: *hyperparameters*), tačiau nei viena iš kombinacijų nedavė tenkinamų rezultatų.

Viena iš nagrinėtų giliojo neuroninio tinklo struktūrų pavaizduota 17a paveiksle. Toks tiesioginio sklidimo neuroninis tinklas, sudarytas iš trijų sluoksnių, mokytas nuspėti vienos reakcijos greičio konstantos vertę (išvestis), tinklui žinant kitos reakcijos greičio konstantos bei biojutiklio atsako vertes (įvestys). Pirmąjį sluoksnį sudaro aštuoni neuronai, aktyvuojami eksponentinės tiesinės (ELu) aktyvacijos funkcijos. Antrą (paslėptą) sluoksnį sudaro šešiolika neuronų, aktyvuojamų hiperbolinės tangento (Tanh) funkcijos. O trečią (išvesties) sluoksnį sudaro vienas neuronas. Tokio neuroninio tinklo rezultatai pateikti 4 lentelėje, bei 17 paveiksle.

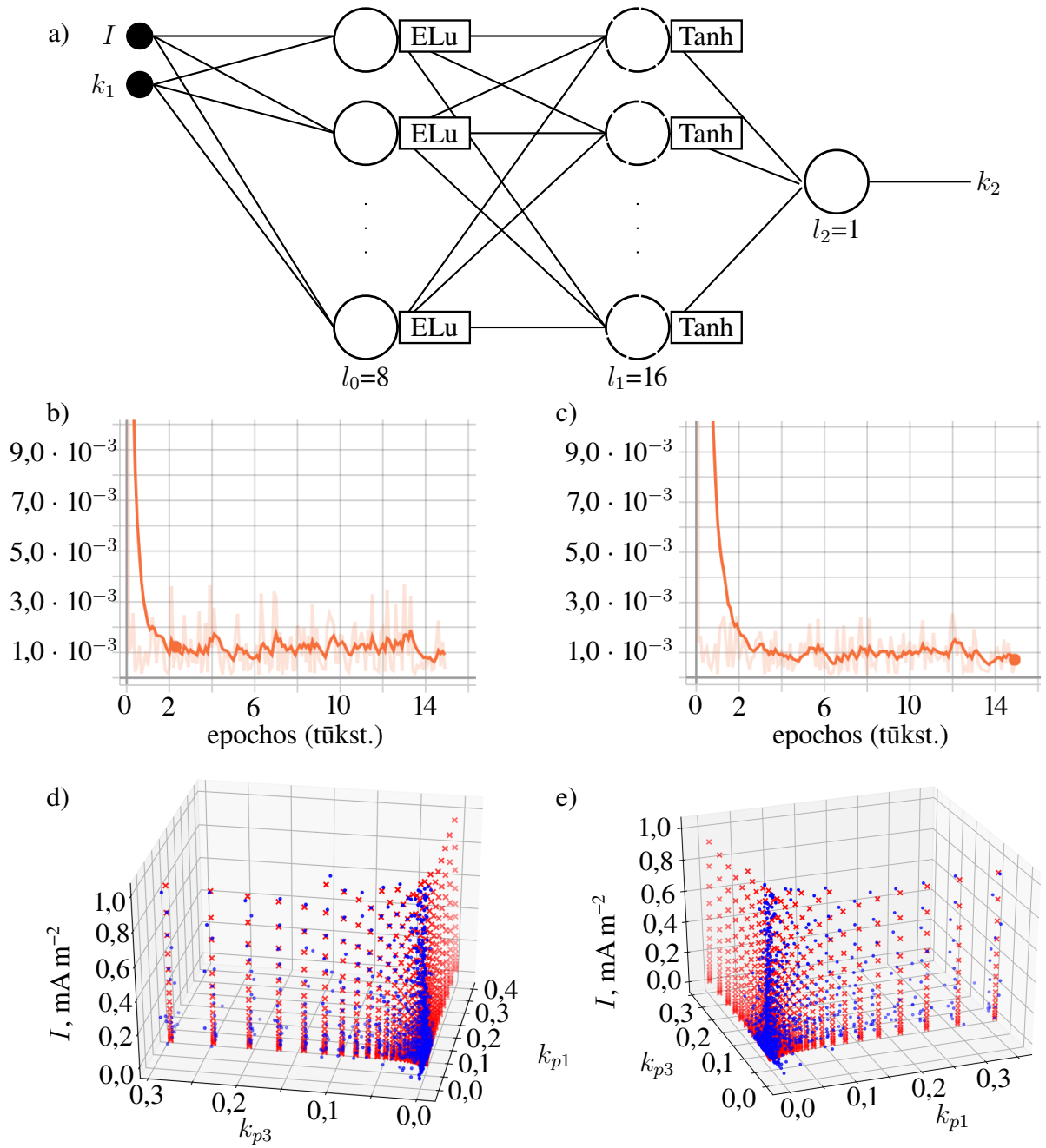
Įvestys	Išvestis	Žingsnis	Epochos (tūkst.)	Netektis (mokymo)	Netektis (testavimo)
I, k_{p3}	k_{p1}	$1{,}0 \cdot 10^{-4}$	15	$9{,}07 \cdot 10^{-4}$	$1{,}14 \cdot 10^{-3}$
I, k_{p1}	k_{p3}	$1{,}0 \cdot 10^{-4}$	15	$7{,}13 \cdot 10^{-4}$	$1{,}21 \cdot 10^{-3}$

4 lentelė. Neuroninio tinklo netekties rezultatai. Rezultatai pateikti paskutinėje mokymo epochoje ir testavimo duomenims: 1. įvestys yra I, k_{p3}, o išvestis k_{p1}; 2. įvestys I, k_{p1}, o išvestis k_{p3}.

Iš 17d ir 17e paveikslų matyti, kad sukonstruotas neuroninis tinklas blogai spėja išvesties vertes abiem reakcijos greičio konstantų išvesčių atvejais, ypač išvesčiai didėjant, nors testavimo netektys yra 10^{-3} eilės.

Didinant neuroninio tinklo talpą, t. y. jį darant sudėtingesniu ir galinčiu apibrėžti daugiau netiesinių sąryšių, išvesčių spėjimo tikslumas nepadidėjo. Iš šių pastebėjimų galima daryti išvadą, kad didelis duomenų tankis mažose reakcijos greičio konstantų vertėse, ir faktas, kad yra panašių biojutiklio atsako I verčių, kurios gali būti aprašytos ne vienos, o kelių reakcijos greičio konstantų kombinacijų, daro neigiamą poveikį. Šis neigiamas poveikis lemia prastą spėjimo tikslumą neuroniniam tinklui bandant atrasti svorių kombinaciją, aprašančią didesnes išvesties vertes.

Nors pasirinktas duomenų surinkimo žingsnis lėmė tai, kad buvo ne tik išvengta skaitinių eksperimentų kartojimo lėtai kintančio biojutiklio atsako srityse, tačiau ir surinkta daug duomenų mėginių srityje, kuomet biojutiklio atsakas intensyviai kinta. Tačiau nepaisant šių teigiamų savybių, stebimas didelis spėjimų kiekis mažose reakcijos greičio konstantose. Tokį stebimą efektą gali lemti būtent tai, kad nagrinėtas rinkinys sudarytas iš logaritminiu žingsniu surinktų duomenų mėginių. Kadangi neuroninis tinklas skaičiuoja neatitikimą tarp išvesties, kurios tikimasi, ir gautos išvesties, tai reiškia, kad esant netolygiam duomenų tankiui, duomenų sritys taip pat turi netolygų indėlį netekties skaičiavimui, ko pasekoje prasčiau išmokstamos sritys su mažesniu duomenų tankiu. Todėl ateities tyrimuose šį duomenų rinkinį patartina normalizuoti.



17 pav. a) Tiesioginio sklidimo neuroninis tinklas, sudarytas iš trijų sluoksnių, mokytas nuspėti vienos reakcijos greičio konstantos vertę (išvestis), tinklui žinant kitos reakcijos greičio konstantos bei biojutiklio atsako vertes (įvestys). Pirmąjį sluoksnį sudaro aštuoni neuronai, aktyvuojami eksponentinės tiesinės (ELu) aktyvacijos funkcijos. Antrą (paslėptą) sluoksnį sudaro šešiolika neuronų, aktyvuojamų hiperbolinės tangento (Tanh) funkcijos. O trečią (išvesties) sluoksnį sudaro vienas neuronas. b) k_{p1} išvesties netekties kitimas epochų atžvilgiu. c) k_{p3} išvesties netekties kitimas epochų atžvilgiu. d) Raudoni taškai – reakcijos greičio konstantų k_{p1}, k_{p3} priklausomybė nuo biojutiklio atsako I, esant fiksuotai reakcijos greičio konstantos k_{m1} = 0,0145 vertei; mėlyni taškai – k_{p1} spėjimas. e) Raudoni taškai – reakcijos greičio konstantų k_{p1}, k_{p3} priklausomybė nuo biojutiklio atsako I, esant fiksuotai reakcijos greičio konstantos k_{m1} = 0,0145 vertei; mėlyni taškai – k_{p3} spėjimas.

7.4. Duomenų rinkinio glodinimas

Priešingai, nei 7.3 skyriuje aprašyto bandymo konstruoti neuroninius tinklus, kurie gebėtų savarankiškai išmokti duomenų savybes, šiame skyriuje nagrinėjamas atvejis, kuomet konstruojamas ne neuronų tinklas, o vienas neuronas, apibrėžtas konkrečia formule, turinčią į nagrinėjamus duomenis panašią topologiją (16a pav.). Radus tokios formulės koeficientus, atvirkštinės paieškos būdu būtų galima nustatyti rinkinį greičio reakcijos konstantų, atitinkančių konkrečią biojutiklio atsako I vertę.

Pateikiama keletas formulių, autoriaus nuomone, turinčių panašumų į turimo duomenų rinkinio topologiją (16a pav.):

$$f(x,y) = \frac{(x-1)}{x} + \frac{y-1}{y} - \frac{xy-1}{xy}, \tag{7.2}$$

$$g(x,y) = x + y + log(x) + log(y). \tag{7.3}$$

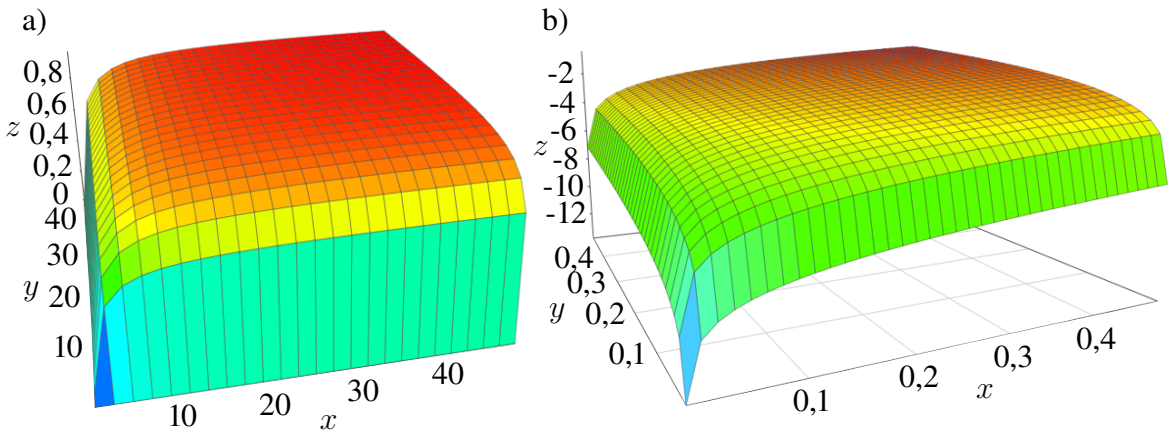
$f(x,y)$ formulės generuojamas paviršius pateiktas 18a paveiksle, o $g(x,y)$ formulės generuojamas paviršius pateiktas 18b paveiksle. Kadangi 7.4 ir 7.5 lygtys operuoja kitose skaičių srityse nei nagrinėjami duomenys, tai šios formulės papildytos koeficientais. Pažymėsime šias formules $f'(k_{p1}, k_{p3})$ ir $g'(k_{p1}, k_{p3})$. Tuomet $f'(k_{p1}, k_{p3})$:

$$f'(k_{p1}, k_{p3}) = b_0 + \frac{(w_1 k_{p1} - w_2)}{w_3 k_{p1}} + \frac{w_4 k_{p3} - w_5}{w_6 k_{p3}} - \frac{w_7 k_{p1} k_{p3} - w_8}{w_9 k_{p1} k_{p3}}, \tag{7.4}$$

kur b_0 – slenksčio konstanta, $w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8, w_9$ – svorių konstantos, o $g'(k_{p1}, k_{p3})$:

$$g'(k_{p1}, k_{p3}) = b_0 + w_1 k_{p1} + w_2 k_{p3} + w_3 log(w_4 k_{p1}) + w_5 log(w_6 k_{p3}), \tag{7.5}$$

kur b_0 – slenksčio konstanta, $w_1, w_2, w_3, w_4, w_5, w_6$ – svorių konstantos.



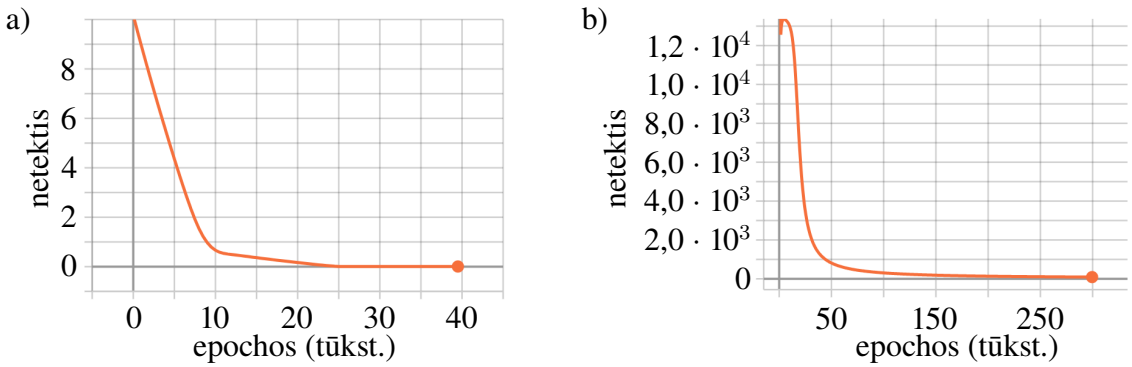
18 pav. a) 7.2 formulės generuojamas paviršius. b) 7.3 formulės generuojamas paviršius.

Prieš apmokymą, duomenų eilutės padalintos į du duomenų rinkinius: pirmą, skirtą giliojo neuroninio tinklo apmokymui, sudarytą iš 80 % visų duomenų, ir antrą, skirtą apmokyto modelio testavimui, sudarytą iš likusių 20 % duomenų.

Eksperimentiniu būdu nustatyta, kad mažiausio kvadrato netektis greičiausiai konvergavo parinkus pradinius svorių ir slenkstinį koeficientus lygius vienetui. Netekties skaitinės vertės skirtingose epochose pateiktos 19 paveiksle, o galutinės netekties vertės mokymo ir testavimo duomenų rinkiniams – 5 lentelėje.

Iš funkcijų kreivių glodinimo matyti, kad geriausius rezultatus parodė funkcijos $g'(k_{p1}, k_{p3})$ (žr. 7.4 lygtį) glodinimas, kuomet testavimo duomenims, iš trijų eksperimentų, netekties vidurkis lygus $6.73 \cdot 10^{-3}$. Tačiau toks rezultatas nėra pakankamas tikslių reakcijos greičio konstantų k_{p1} ir k_{p3} nustatymui, todėl atvirkštinė šių charakteristikų paieška iš 7.4 lygties su išmoktais svoriais nebuvo atlikta. Autoriaus nuomone, nors vizualiai nagrinėtos funkcijos buvo panašios į duomenų rinkinį, tačiau iš tikro jo neatitiko, todėl reikalingas papildomas tyrimas funkcijos, pilnai atitinkančios nagrinėjamus duomenis, radimui.

Sėkmingai priglodinus paviršių prie duomenų rinkinio, galima būtų daugiau nei vieną



19 pav. a) Funkcijos $g'(k_{p1}, k_{p3})$ (žr. 7.4 lygtį) netekties priklausomybė nuo epochos. b) Funkcijos $f'(k_{p1}, k_{p3})$ (žr. 7.5 lygtį) netekties priklausomybė nuo epochos.

Glodinimo funkcija	Žingsnis	Epochos (tūkst.)	Netektis (mokymo)	Netektis (testavimo)
$f'(k_{p1}, k_{p3})$	0,001	300	79,93	71,64
$g'(k_{p1}, k_{p3})$	0,0001	40	$6{,}76 \cdot 10^{-3}$	$6{,}73 \cdot 10^{-3}$

5 lentelė. Funkcijų $f'(k_{p1}, k_{p3})$ ir $g'(k_{p1}, k_{p3})$ (žr. 7.4, 7.5 lygtis) glodinimo duomenų rinkiniui (žr. 16a pav.) rezultatai.

8. IŠVADOS

- Šiame darbe įgyvendintas biojutiklio skaitinis modelis, įskaitantis S, P, E ir EP kinetiką pagal [36] metodinę medžiagą, yra tinkamas apskaičiuoti elektrocheminio biojutiklio, kurio mechanizmas paremtas difuzijos ir fermentinių reakcijų kinetikos procesais, rezultatus.
- Žinant biojutiklio atsaką I, galima nustatyti vieną iš S_0, $D_{S_{m2}}$, $D_{S_{m1}}$, D_{S_e}, $D_{P_{m2}}$, $D_{P_{m1}}$, D_{P_e}, k_{m1}, k_{m3}, k_{p1}, k_{p3} pradinių sąlygų, jei kitos yra fiksuojamos.
- Yra ne viena k_{m1}, k_{p1}, k_{p3} reakcijos greičio konstantų kombinacija, apibrėžianti panašią biojutiklio atsako I vertę.
- Didėjant reakcijos greičio konstantoms k_{m1}, k_{p1}, k_{p3} ir substrato pradinei koncentracijai S_0, biojutiklio atsako I kitimo intensyvumas mažėja.

9. Ateities tyrimų gairės

Šio magistrinio darbo metu atliktas tyrimas yra tik mažas žingsnis modelinių biojutiklių sistemų nagrinėjime. Siekiant patobulinti ir praplėsti biojutiklio modelio, įskaitančio S, P, E ir EP kinetiką, jis gali būti pratęstas keliomis kryptimis:

- Sumažinti įgyvendinto biojutiklio skaitinio modelio atsako skaičiavimo trukmę jį optimizuojant ir išlygiagretinant.
- Praplėsti įgyvendintą biojutiklio matematinį modelį papildoma dimensija, pereinant nuo siūlo iki dviejų, ar trijų dimensijų modelio.
- Praplėsti įgyvendintą biojutiklio matematinį modelį papildomais biocheminiais parametrais, tokiais kaip pH ar temperatūra.
- Praplėsti sudarytą skaitinių eksperimentų duomenų rinkinį.
- Tęsti biojutiklio atsako priklausomybės nuo pradinių sąlygų analizę.
- Tęsti pradinių sąlygų nustatymo analize žinant biojutiklio atsaką.

LITERATŪROS ŠALTINIAI

[1] *Landmark Writings in Western Mathematics 1640-1940*. Elsevier Science, feb 2005.

[2] Ajith Abraham. Artificial neural networks. *handbook of measuring system design*, 2005.

[3] Robert A. Alberty and Gordon G. Hammes. Application of the theory of diffusion-controlled reactions to enzyme kinetics. *The Journal of Physical Chemistry*, 62(2):154–159, feb 1958.

[4] Ethem Alpaydin. *Introduction to machine learning*. MIT press, 2009.

[5] Fractal Analysis. Development cost of a biosensor, 2016. https://www.barnardhealth.us/fractal-analysis/development-cost-of-a-biosensor.html. Aplankyta: 2016-05-15.

[6] R Baronas, F Ivanauskas, and J Kulys. Modelling dynamics of amperometric biosensors in batch and flow injection analysis. *Journal of mathematical chemistry*, 32(2):225–237, 2002.

[7] Romas Baronas, Feliksas Ivanauskas, and Juozas Kulys. *Mathematical Modeling of Biosensors: An Introduction for Chemists and Mathematicians (Springer Series on Chemical Sensors and Biosensors)*. Springer, 2010 edition, 12 2009.

[8] Imad A Basheer and Maha Hajmeer. Artificial neural networks: fundamentals, computing, design, and application. *Journal of microbiological methods*, 43(1):3–31, 2000.

[9] D Richard Baughman and Yih An Liu. *Neural networks in bioprocessing and chemical engineering*. Academic press, 2014.

[10] Yoshua Bengio et al. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

[11] Jeremy M Berg, John L Tymoczko, Lubert Stryer, et al. Biochemistry, 2002.

[12] Michael R Berthold and David J Hand. *Intelligent data analysis: an introduction*. Springer, 2007.

[13] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. *Nature*, 549(7671):195, 2017.

[14] Avrim Blum. Rank-r decision trees are a subclass of r-decision lists. *Information Processing Letters*, 42(4):183–185, 1992.

[15] Encyclopaedia Britannica et al. Encyclopædia britannica. *Chicago: Common Law*, 2009.

[16] Chi-Tsong Chen. *Linear system theory and design*. Oxford University Press, Inc., 1998.

[17] Ivan Nunes da Silva. *Artificial Neural Networks: A Practical Course*. Springer, aug 2016.

[18] Li Deng, Geoffrey Hinton, and Brian Kingsbury. New types of deep neural network learning for speech recognition and related applications: An overview. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8599–8603. IEEE, 2013.

[19] Matthew Francis Dixon, Diego Klabjan, and Jin Hoon Bang. Classification-based financial markets prediction using deep neural networks. *SSRN Electronic Journal*, 2016.

[20] Adrián Fernández Gavela, Daniel Grajales García, Jhonattan Ramirez, and Laura Lechuga. Last advances in silicon-based optical biosensors. *Sensors*, 16(3):285, feb 2016.

[21] Carla Hannaford. *Smart moves: Why learning is not all in your head.* ERIC, 1995.

[22] William R. Heineman and William B. Jensen. Leland c. clark jr. (1918–2005). *Biosensors and Bioelectronics*, 21(8):1403–1404, feb 2006.

[23] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, jan 1989.

[24] Anil K Jain, Jianchang Mao, and K Moidin Mohiuddin. Artificial neural networks: A tutorial. *Computer*, 29(3):31–44, 1996.

[25] Jurgis Kadziauskas. Biochemijos pagrindai. *Vilniaus universiteto leidykla*, 2008.

[26] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, may 2017.

[27] Anders Krogh. What are artificial neural networks? *Nature biotechnology*, 26(2):195, 2008.

[28] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 2012.

[29] Sue Han Lee, Chee Seng Chan, Paul Wilkin, and Paolo Remagnino. Deep-plant: Plant identification with convolutional neural networks. In *Image Processing (ICIP), 2015 IEEE International Conference on*, pages 452–456. IEEE, 2015.

[30] CR Lowe. An introduction to the concepts and technology of biosensors. *Biosensors*, 1(1):3–16, 1985.

[31] C. Van Der Malsburg. Frank rosenblatt: Principles of neurodynamics: Perceptrons and the theory of brain mechanisms. In *Brain Theory*, pages 245–248. Springer Berlin Heidelberg, 1986.

[32] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[33] Helmut Mehrer and Nicolaas A Stolwijk. Heroes and highlights in the history of diffusion. *Diffusion Fundamentals*, 11(1):1–32, 2009.

[34] Parikha Mehrotra. Biosensors and their applications – a review. *Journal of Oral Biology and Craniofacial Research*, 6(2):153–159, may 2016.

[35] Tadas Meškauskas, Feliksas Ivanauskas, and Valdas Laurinavicius. Degradation of substrate and/or product: mathematical modeling of biosensor action. *Journal of Mathematical Chemistry*, 51(9):2491–2502, jul 2013.

[36] Tadas Meškauskas. *Netiesinio modeliavimo metodai. Metodinė medžiaga studentų projektams.* Vilniaus universitetas, Matematikos ir informatikos fakultetas, 2016.

[37] ML Minsky and SA Papert. Perceptrons: An introduction to computational geometry, 1969.

[38] Charles R Noback, Norman L Strominger, Robert J Demarest, and David A Ruggiero. *The human nervous system: structure and function*. Number 744. Springer Science & Business Media, 2005.

[39] William H Press. *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press, 2007.

[40] Jaime Punter-Villagrasa, Jordi Colomer-Farrarons, Francisco J del Campo, and Pere Miribel. *Amperometric and Impedance Monitoring Systems for Biomedical Applications*, volume 4. Springer, 2017.

[41] P. Ravindran. Properties of materials, 2014. http://folk.uio.no/ravi/cutn/pmat/4.diffusion+ficks.pdf. Aplankyta: 2016-05-25.

[42] H-G Roos. Thomas, jw: Numerical partial differential equations. finite difference methods. new york etc., springer-verlag 1995. xx, 437 pp., dm 78,-. isbn 0-387-97999-9 (texts in applied mathematics 22). *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, 77(5):386–386, 1997.

[43] Frank Rosenblatt. *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

[44] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[45] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[46] FW Scheller, R Hintsche, D Pfeiffer, F Schubert, K Riedel, and R Kindervater. Biosensors: fundamentals, applications and trends. *Sensors and Actuators B: Chemical*, 4(1-2):197–206, 1991.

[47] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[48] Aby Thyparambil, Ingrid Bazin, and Anthony Guiseppi-Elie. Molecular modeling and simulation tools in the development of peptide-based biosensors for mycotoxin detection: Example of ochratoxin. *Toxins*, 9(12):395, dec 2017.

[49] Tuition Tube. The Michaelis-Menten equation in biochemistry, 2016. https://tuitiontube.com/the-michaelis-menten-equation/. Aplankyta: 2016-05-14.

[50] Anthony Turner, etc., Isao Karube, and George S. Wilson, editors. *Biosensors: Fundamentals and Applications*. Oxford University Press, 8 1987.

[51] S. Vigneshvar, C. C. Sudhakumari, Balasubramanian Senthilkumaran, and Hridayesh Prakash. Recent advances in biosensor technology for potential applications – an overview. *Frontiers in Bioengineering and Biotechnology*, 4, feb 2016.

PRIEDAI

Dokumentą sudaro priedas A. Tai įgyvendintas biojutiklio skaitinis modelis PYTHON programavimo kalba.

A. Įgyvendintas biojutiklio, įskaitančio S, P, E ir EP kinetiką, skaitinis modelis

```
# -*- coding: utf-8 -*-

import sys
import numpy as np
import matplotlib.pyplot as plt

""" Kompiuterinis biojutiklių modeliavimas (Computer Modeling of Biosensors)

"Skaitinis algoritmo implementacija modeliui, įskaitančiam S, P, E ir EP kinetiką" (difunduojančio produkto į tirpalą atvejis)
remiantis Prof. dr. T. Meškauso mokymo priemonės "Netiesinio modeliavimo metodai – metodinė medžiaga projektams" teorine medžiaga.

Darbą atliko: Danielius Dvareckas
Vadodas: Prof. dr. Tadas Meškauskas
"""

# Pagalbiniai kintamieji


masininis_tikslumas = sys.float_info.epsilon


# Biocheminiai kitamieji (pagal metodinės medžiagos 30 puslapį)


N = 1000  # dalių intervalo [0, d_m1 + d_e + d_m2] skaičius
d_m1 = 2e-06  # elektrodą gaubianti membrana (nuo 0e-6 iki 10e-6 m)
d_e = 9e-06  # fermento sluoksnio storio membrana (nuo 3e-6 iki 5e-6 m ar nuo 5e-6 iki 10e-6 m)
d_m2 = 10e-06  # išorinės membranos storis (nuo 5e-6 iki 10e-6 m)
h = 0.021e-06  # h = (d_m1 + d_e + d_m2) / N – pastovaus žingsnio tinklas
E_0 = 0.19  # fermento koncentracija fermento sluoksnyje biojutiklio modeliavimo trukmės pradžioje
S_0 = 1.0  # pastovi substrato koncentracija tirpale, į kurį panardintas biojutiklis
D_S_m2 = 7e-12  # substrato difuzijos koeficientas išorinėje membranoje
D_S_m1 = 6e-12  # substrato difuzijos koeficientas elektrodą gaubančioje membranoje
D_S_e = 22e-12  # substrato difuzijos koeficientas fermente
D_P_m2 = 6e-12  # produkto difuzijos koeficientas išorinėje membranoje
D_P_m1 = 5e-12  # produkto difuzijos koeficientas elektrodą gaubančioje membranoje
D_P_e = 20e-12  # produkto difuzijos koeficientas fermente
k_m1 = 0.00153846  # fermento–produkto komplekso susidarymo greičio konstanta
```

```
39 k_p1 = 0.0153846  # fermento–produkto komplekso susidarymo atvirkštinė
       greičio konstanta
40 k_m3 = 0.002  # produkto susidarymo greičio konstanta
41 k_p3 = 0.002  # produkto susidarymo atvirkštinė greičio konstanta
42 C_1 = 0  # substrato nuostolių santykinis greitis
43 C_2 = 0  # fermento nuostolių santykinis greitis
44 tau_max = 0.001  # laiko žingsnis sekundėmis
45
46 # Pagalbinės funkcijos
47
48
49 def klaida(zinute):
50     """
51     Klaidos spausdinimo funkcija, kai pasiekiamas neleistinas rėžis ar
           kintamojo vertė
52     :param zinute: klaidos tekstas
53     :return:
54     """
55     sys.exit(zinute)
56
57
58 # Koordinatės tinklas
59
60
61 def x(i):
62     """
63     Koordinatės indekso pavertimas į koordinatę metrais
64     :param i: koordinatės indeksas
65     :return: koordinatė
66     """
67     x_i = i * h
68     if not (0 <= x_i <= d_m1 + d_e + d_m2):
69         klaida("x vertė nepatenka į intervalą: 0 <= x <= d_m1 + d_e +
              d_m2")
70     return x_i
71
72
73 def x_mh(i):
74     """
75     Dabartinės ir prieš tai buvusios koordinatės vidurkio skaičiavimas
76     :param i: koordinatės indeksas
77     :return: koordinatė
78     """
79     if not (1 <= i <= N - 1):
80         klaida("i nepatenka į intervalą: 1 <= i <= N - 1")
81     return (x(i - 1) + x(i)) / 2.0
82
83
84 def x_ph(i):
85     """
86     Dabartinės ir sekančios koordinatės vidurkio skaičiavimas
87     :param i: koordinatės indeksas
88     :return: koordinatė
89     """
90     if not (1 <= i <= N - 1):
```

```
91          klaida("i nepatenka į intervalą: 1 <= i <= N - 1")
92      return (x(i) + x(i + 1)) / 2.0
93
94
95  # Laiko tinklas
96
97
98  def t(k):
99      """
100     Laiko indekso pavertimas į laiką sekundėmis
101     :param k: laiko indeksas
102     :return: laikas
103     """
104     return round(k * tau_max, 4)
105
106
107 def tau(k):
108     """
109     Laiko indekso pavertimas į laiką žingsnį
110     :param k: laiko indeksas
111     :return: laiko žingsnis
112     """
113     return tau_max
114
115
116 # Fermento koncentracija
117
118
119 def E_pradinis(i):
120     """
121     Fermento koncentracija fermento sluoksnyje biojutiklio modeliavimo
            trukmės pradžioje
122     :param i: koordinatės kintamasis
123     :return: fermento koncentracija
124     """
125     x_i = x(i)
126     if 0 <= x_i < d_m1:
127         return 0
128     elif d_m1 <= x_i <= d_m1 + d_e:
129         return E_0
130     elif d_m1 + d_e < x_i <= d_m1 + d_e + d_m2:
131         return 0
132     klaida("E netinkamas intervalas")
133
134
135 def E_pradinis_masyvas():
136     """
137     Pradinio fermento koncentracijų masyvo sudarymas
138     :return:  pradinis E masyvas
139     """
140     E_masyvas = np.zeros(N + 1)  # 0...N
141     for i in range(0, N + 1):  # 0...N
142         E_masyvas[i] = E_pradinis(i)
143     return E_masyvas
144
```

```
145
146 def E_new(tau_k, alpha_i, S_new_i, P_new_i, E_k_i):
147     """
148     Patikslinto fermento artinio skaičiavimas
149     :param tau_k: laiko žigsnis
150     :param alpha_i: alpha parametras
151     :param S_new_i: patikslintas S
152     :param P_new_i: patikslintas P
153     :param E_k_i: praeitas E
154     :return: E artinys
155     """
156     if (E_k_i == 0):
157         return 0
158     return (E_k_i * (2 / tau_k + alpha_i * (k_m1 + k_p3) * (E_0 /
            E_k_i))) \
159             / (2 / tau_k + alpha_i * (k_p1 * S_new_i + k_m3 * P_new_i +
                    k_m1 + k_p3))
160
161
162 def E_artiniu_masyvas(k, S_new_i, P_new_i, E_k_i):
163     """
164     E artinių masyvo skaičiavimas
165     :param k: laiko indeksas
166     :param S_new_i: S artinys
167     :param P_new_i: P artinys
168     :param E_k_i: praeitas E
169     :return: E artinių masyvas
170     """
171     E_new_i = np.zeros(N + 1)
172     for i in range(0, N + 1):  # 0...N
173         E_new_i[i] = E_new(tau(k), alpha(i), S_new_i[i], P_new_i[i],
                E_k_i[i])
174     return E_new_i
175
176
177 def E_dabartinis(E_k_i, E_kph_i):
178     """
179     E dabartinio skaičiavimas
180     :param E_k_i: E praeitas
181     :param E_kph_i: E suvidurkintas
182     :return: E dabartinis
183     """
184     E_k_i = [2 * m for m in E_kph_i] - E_k_i
185     return E_k_i
186
187
188 # Substrato procedūros
189
190
191 def S_pradinis(i):
192     """
193     Substrato koncentracija tirpale į kurį panardintas biojutiklis
            modeliavimo trukmės pradžioje
194     :param i: koordinatės kintamasis
195     :return: S pradinis
```

```
196     """
197     if 0 <= i <= N - 1:
198         return 0
199     elif i == N:
200         return S_0
201     klaida("S netinkamas intervalas")
202
203
204 def S_pradinis_masyvas():
205     """
206     S pradinio masyvo skaičiavimas
207     :return: S pradinio masyvas
208     """
209     S_masyvas = np.zeros(N + 1)  # 0...N
210     for i in range(0, N + 1):  # 0...N
211         S_masyvas[i] = S_pradinis(i)
212     return S_masyvas
213
214
215 def S_artiniu_masyvas(k, S_k_i, E_old_i):
216     """
217     S artiniu masyvo skaičiavimas
218     :param k: laiko indeksas
219     :param S_k_i: S dabartinio masyvas
220     :param E_old_i: E praeito masyvas
221     :return: S artinių masyvas
222     """
223     [F_S_arr, a_S_arr, b_S_arr, c_S_arr] = [np.zeros(N + 1), np.zeros(
            N + 1), np.zeros(N + 1), np.zeros(N + 1)]  # 0..N
224     for i in range(1, N):  # 1...N-1
225         F_S_arr[i] = F(k, i, S_k_i[i], E_old_i[i])
226         a_S_arr[i] = a_S(i)
227         b_S_arr[i] = b_S(i)
228         c_S_arr[i] = c_S(k, i, E_old_i[i])
229     A_arr = TDM(np.zeros(N + 1), a_S_arr, b_S_arr, c_S_arr, 1, 0)
230     B_arr = TDM(np.zeros(N + 1), a_S_arr, b_S_arr, c_S_arr, 0, 1)
231     Y_arr = TDM(F_S_arr, a_S_arr, b_S_arr, c_S_arr, 0, 0)
232     for i in range(0, N + 1):  # 0..N
233         if abs(A_arr[i]) < masininis_tikslumas:
234             A_arr[i] = 0
235         if abs(B_arr[i]) < masininis_tikslumas:
236             B_arr[i] = 0
237         if abs(Y_arr[i]) < masininis_tikslumas:
238             Y_arr[i] = 0
239     S_new_i = np.zeros(N + 1)  # 0..N
240     S_new_i[0] = (4 * (S_0 * B_arr[1] + Y_arr[1]) - S_0 * B_arr[2] -
            Y_arr[2]) / (3 - 4 * A_arr[1] + A_arr[2])
241     if abs(S_new_i[0]) < masininis_tikslumas:
242         S_new_i[0] = 0
243     S_new_i[N] = S_0
244     for i in range(0, N + 1):  # 0..N
245         S_new_i[i] = S_new_i[0] * A_arr[i] + S_new_i[N] * B_arr[i] +
                Y_arr[i]
246         if (S_new_i[i] < masininis_tikslumas):
247             S_new_i[i] = 0
```

```
248     return S_new_i
249
250
251 def S_dabartinis(S_k_i, S_kph_i):
252     """
253     S dabartinio masyvo skaičiavimas
254     :param S_k_i: S praeitas masyvas
255     :param S_kph_i: S suvidurkintas masyvas
256     :return: S dabartinis masyvas
257     """
258     S_k_i = [2 * m for m in S_kph_i] - S_k_i
259     return S_k_i
260
261
262 def a_S(i):
263     """
264     a koeficiento substratui skaičiavimas
265     :param i: koordinatės indeksas
266     :return: a koeficientas substratui
267     """
268     return D_S(x_mh(i)) / (h ** 2)
269
270
271 def b_S(i):
272     """
273     b koeficiento substratui skaičiavimas
274     :param i: koordinatės indeksas
275     :return: b koeficientas substratui
276     """
277     return D_S(x_ph(i)) / (h ** 2)
278
279
280 def c_S(k, i, E):
281     """
282     c koeficiento substratui skaičiavimas
283     :param k: laiko indeksas
284     :param i: koordinatės indeksas
285     :param E: fermento koncentracija
286     :return: c koeficientas substratui
287     """
288     return a_S(i) + b_S(i) + (2 / tau(k)) + C_1 + alpha(i) * k_p1 * E
289
290
291 def D_S(x):
292     """
293     Substrato difuzijos koeficiento skaičiavimas
294     :param x: koordinatė
295     :return: S difuzijos koeficientas
296     """
297     if 0 < x < d_m1:
298         return D_S_m1
299     elif d_m1 <= x <= d_m1 + d_e:
300         return D_S_e
301     elif d_m1 + d_e < x < d_m1 + d_e + d_m2:
302         return D_S_m2
```

```
303     klaida("D_S␣Pasiekta␣neįmanoma␣kodo␣vieta")
304
305
306 def F(k, i, S_k_i, E_k_i):
307     """
308     F koeficiento substratui skaičiavimas
309     :param k: laiko indeksas
310     :param i: koordinatės indeksas
311     :param S_k_i: S masyvas
312     :param E_k_i: E masyvas
313     :return: F koeficiento vertė
314     """
315     F_val = ((2 * S_k_i) / tau(k)) + (alpha(i) * k_m1 * E_0) * (1 - (
            E_k_i / E_0))
316     if F_val < masininis_tikslumas:
317         F_val = 0
318     return F_val
319
320
321 # Produkto procedūros
322
323
324 def P_pradinis(i):
325     """
326     Produkto koncentracija pradiniu laiko momentu
327     :param i: koordinatės kintamasis
328     :return: P pradinis
329     """
330     if 0 <= i <= N:
331         return 0
332     klaida("P␣netinkamas␣intervalas")
333
334
335 def P_pradinis_masyvas():
336     """
337     P pradinio masyvo konstravimas
338     :return: P pradinio masyvas
339     """
340     P_masyvas = np.zeros(N + 1)  # 0...N
341     for i in range(0, N + 1):  # 0...N
342         P_masyvas[i] = P_pradinis(i)
343     return P_masyvas
344
345
346 def P_artiniu_masyvas(k, P_k_i, E_old_i):
347     """
348     P artinių masyvo konstravimas
349     :param k: laiko indeksas
350     :param P_k_i: P masyvas
351     :param E_old_i: E masyvas
352     :return:P artinių masyvas
353     """
354     G_P_arr = np.zeros(N + 1)
355     a_P_arr = np.zeros(N + 1)
356     b_P_arr = np.zeros(N + 1)
```

```
357     c_P_arr = np.zeros(N + 1)
358     for i in range(1, N):  # 1...N-1
359         G_P_arr[i] = G(k, i, P_k_i[i], E_old_i[i])
360         a_P_arr[i] = a_P(i)
361         b_P_arr[i] = b_P(i)
362         c_P_arr[i] = c_P(k, i, E_old_i[i])
363     P_new_i = TDM(G_P_arr, a_P_arr, b_P_arr, c_P_arr, 0, 0)
364     return P_new_i
365
366
367 def P_dabartinis(P_k_i, P_kph_i):
368     """
369     P dabartinio skaičiavimas
370     :param P_k_i: P
371     :param P_kph_i: P suvidurkintas
372     :return: P dabartinis
373     """
374     P_k_i = [2 * m for m in P_kph_i] - P_k_i
375     return P_k_i
376
377
378 def a_P(i):
379     """
380     a koeficiento P skaičiavimas
381     :param i: koordinatės indeksas
382     :return: a koeficientas P
383     """
384     return D_P(x_mh(i)) / h ** 2
385
386
387 def b_P(i):
388     """
389     b koeficiento skaičiavimas P
390     :param i: koordinatės indeksas
391     :return: b koeficientas P
392     """
393     return D_P(x_ph(i)) / h ** 2
394
395
396 def c_P(k, i, E):
397     """
398     c koeficiento skaičiavimas
399     :param k: laiko indeksas
400     :param i: koordinatės indeksas
401     :param E: fermento koncentracija
402     :return: c koeficientas P
403     """
404     return a_P(i) + b_P(i) + (2 / tau(k)) + C_2 + alpha(i) * k_m3 * E
405
406
407 def D_P(x):
408     """
409     Produkto difuzijos koeficiento skaičiavimas
410     :param x: koordinatė
411     :return: P difuzijos koeficientas
```

```
412     """
413     if x == 0:
414         if d_m1 != 0:
415             return D_P_m1
416         else:
417             return D_P_e
418     elif 0 < x < d_m1:
419         return D_P_m1
420     elif d_m1 <= x <= d_m1 + d_e:
421         return D_P_e
422     elif d_m1 + d_e < x < d_m1 + d_e + d_m2:
423         return D_P_m2
424     klaida("D_S Pasiekta neįmanoma kodo vieta")
425
426
427 def G(k, i, P_k_i, E_k_i):
428     """
429     G koeficiento skaičiavimas produktui
430     :param k: laiko indeksas
431     :param i: koordinatės indeksas
432     :param P_k_i: P koncentracija
433     :param E_k_i: E koncentracija
434     :return: G koeficientas P
435     """
436     G_val = ((2 * P_k_i) / tau(k)) + (alpha(i) * k_p3 * E_0) * (1 - (
            E_k_i / E_0))
437     if G_val < masininis_tikslumas:
438         G_val = 0
439     return G_val
440
441
442 # Alpha
443
444
445 def alpha(i):
446     """
447     Atskirų biojutiklio sluoksnių konstanta
448     :param i: koordinatės indeksas
449     :return: konstantos vertė
450     """
451     x_i = x(i)
452     if x_i == 0:
453         if d_m1 != 0:
454             return 0
455         else:
456             return 1
457     elif 0 < x_i < d_m1:
458         return 0
459     elif d_m1 <= x_i <= d_m1 + d_e:
460         return 1
461     elif d_m1 + d_e < x_i <= d_m1 + d_e + d_m2:
462         return 0
463     klaida(F"alpha pasiekta neįmanoma kodo vieta")
464
465
```

```
466 def TDM(F_arr, a_arr, b_arr, c_arr, gamma1, gamma2):
467     """
468     Tridiagonalinės matricos algoritmas (Thomas algoritmas)
469     :param F_arr: F koeficientų masyvas
470     :param a_arr: a koeficientų masyvas
471     :param b_arr: b koeficientų masyvas
472     :param c_arr: c koeficientų masyvas
473     :param gamma1: gamma 1
474     :param gamma2: gamma 2
475     :return:  sprendinių masyvas
476     """
477     [y, alpha, beta] = [[0]*(N+1), [0]*(N+1), [0]*(N+1)]
478     [alpha[1], beta[1]] = [0, gamma1]
479     for j in range(1, N):
480         alpha[j+1] = b_arr[j]/(c_arr[j]-a_arr[j]*alpha[j])
481         beta[j+1] = (F_arr[j]+a_arr[j]*beta[j])/(c_arr[j]-a_arr[j]*
                alpha[j])
482     y[N] = gamma2
483     for i in range(N, 0, -1):
484         y[i-1] = alpha[i]*y[i] + beta[i]
485     y[0] = y[1]
486     return np.asarray(y)
487
488
489 # Programa
490
491
492 def spausdink_iteracijos_antraste(k, T):
493     """
494     Iteracijos antraštės spausdinimas
495     :param k: laiko indeksas
496     :param T: laikas
497     :return:
498     """
499     print(F"### t({k})={t(k)} <= {T}")
500
501
502 def tikrink_artinius(S_k_i, P_k_i, E_k_i, S_new_i, P_new_i, E_new_i):
503     """
504     Artinių tikrinimas
505     :param S_k_i: S
506     :param P_k_i: P
507     :param E_k_i: E
508     :param S_new_i: S artinys
509     :param P_new_i: P artinys
510     :param E_new_i: E artinys
511     :return:
512     """
513     for i in range(0, N + 1):
514         if not (0 <= S_new_i[i] <= S_0):
515             klaida("S_new_i netenkina sąlygų")
516         if not(0 <= P_new_i[i]):
517             klaida("P_new_i netenkina sąlygų")
518         if not(0 <= E_new_i[i] <= E_0):
```

```
519             klaida(F"E_new_i netenkina sąlygų 0 <= E_new_i[i] and
                    E_new_i[i] <= E_0: 0 <= {E_new_i[i]} and {E_new_i[i]}
                    <= {E_0}")
520     [S_k_i_max, S_new_i_max] = [S_k_i[0], S_new_i[0]]
521     [P_k_i_max, P_new_i_max] = [P_k_i[0], P_new_i[0]]
522     [E_k_i_max, E_new_i_max] = [E_k_i[0], E_new_i[0]]
523     for i in range(0, N):
524         if S_k_i_max < S_k_i[i]:
525             S_k_i_max = S_k_i[i]
526         if S_new_i_max < S_new_i[i]:
527             S_new_i_max = S_new_i[i]
528     for i in range(0, N + 1):
529         if P_k_i_max < P_k_i[i]:
530             P_k_i_max = P_k_i[i]
531         if P_new_i_max < P_new_i[i]:
532             P_new_i_max = P_new_i[i]
533         if E_k_i_max < E_k_i[i]:
534             E_k_i_max = E_k_i[i]
535         if E_new_i_max < E_new_i[i]:
536             E_new_i_max = E_new_i[i]
537     if not (S_k_i_max < S_new_i_max):
538         klaida("netenkina sąlygų: S_k_i_max < S_new_i_max")
539     if not (P_k_i_max <= P_new_i_max):
540         klaida("netenkina sąlygų: P_k_i_max <= P_new_i_max")
541     if not (E_k_i_max >= E_new_i_max > masininis_tikslumas * 100):
542         klaida("netenkina sąlygų: (E_k_i_max >= E_new_i_max >
                masininis_tikslumas * 100)")
543
544
545 def ivertink_progresus(k, S_new_i, P_new_i, E_new_i, S_k_i, P_k_i,
        S_old_i, P_old_i, E_old_i):
546     """
547     Progresu vertinimas
548     :param k: laiko indeksas
549     :param S_new_i: S artinys
550     :param P_new_i: P artinys
551     :param E_new_i: E artinys
552     :param S_k_i: S
553     :param P_k_i: P
554     :param S_old_i: S praeitas
555     :param P_old_i: P praeitas
556     :param E_old_i: E praeitas
557     :return: Progresų ir netikčių masyvas
558     """
559     S_netiktis = np.zeros(N)
560     P_netiktis = np.zeros(N)
561     for i in range(1, N):  # 1..N-1
562         S_netiktis[i] = (abs(a_S(i) * S_new_i[i - 1] - c_S(k, i,
                E_new_i[i]) * S_new_i[i] + b_S(i) * S_new_i[i + 1] + F(k, i
                , S_k_i[i], E_new_i[i]))) \
563                         / (abs(a_S(i)) + abs(b_S(i)) + abs(c_S(k, i,
                            E_new_i[i])))
564         P_netiktis[i] = (abs(a_P(i) * P_new_i[i - 1] - c_P(k, i,
                E_new_i[i]) * P_new_i[i] + b_P(i) * P_new_i[i + 1] + G(k, i
                , P_k_i[i], E_new_i[i]))) \
```

```
565                         / (abs(a_P(i)) + abs(b_P(i)) + abs(c_P(k, i,
                                E_new_i[i])))
566     S_progresas = abs(S_new_i - S_old_i)
567     P_progresas = abs(P_new_i - P_old_i)
568     E_progresas = abs(E_new_i - E_old_i)
569     S_progresas_max = S_progresas[0]
570     P_progresas_max = P_progresas[0]
571     E_progresas_max = E_progresas[0]
572     S_netiktis_max = S_netiktis[1]
573     P_netiktis_max = P_netiktis[1]
574     for i in range(0, N + 1):
575         if (S_progresas_max < S_progresas[i]):
576             S_progresas_max = S_progresas[i]
577         if (P_progresas_max < P_progresas[i]):
578             P_progresas_max = P_progresas[i]
579         if (E_progresas_max < E_progresas[i]):
580             E_progresas_max = E_progresas[i]
581     for i in range(1, N):
582         if (S_netiktis_max < S_netiktis[i]):
583             S_netiktis_max = S_netiktis[i]
584         if (P_netiktis_max < P_netiktis[i]):
585             P_netiktis_max = P_netiktis[i]
586     return [S_progresas_max, P_progresas_max, E_progresas_max,
            S_netiktis_max, P_netiktis_max]
587
588
589 def tenkina_stabdymo_kriterijai(sigma, S_new_i, P_new_i, E_new_i,
        S_progresas_max, P_progresas_max, E_progresas_max, S_netiktis_max,
        P_netiktis_max):
590     """
591     Stabdymo kriterijų tikrinimas
592     :param sigma: sigma
593     :param S_new_i: S artinys
594     :param P_new_i: P artinys
595     :param E_new_i: E artinys
596     :param S_progresas_max: S didžiausias progresas
597     :param P_progresas_max: P didžiausias progresas
598     :param E_progresas_max: E didžiausias progresas
599     :param S_netiktis_max: S didžiausia netiktis
600     :param P_netiktis_max: P didžiausia netiktis
601     :return: bool vertė stabdyti ar ne
602     """
603     [S_new_i_max, P_new_i_max, E_new_i_max] = [S_new_i[0], P_new_i[0],
            E_new_i[0]]
604     for i in range(0, N + 1):
605         if (S_new_i_max < S_new_i[i]): S_new_i_max = S_new_i[i]
606         if (P_new_i_max < P_new_i[i]): P_new_i_max = P_new_i[i]
607         if (E_new_i_max < E_new_i[i]): E_new_i_max = E_new_i[i]
608     if not (E_new_i_max > 0 and S_new_i_max > 0):
609         klaida("netenkinama␣sąlyga:␣E_new_i_max␣>␣0␣arba␣S_new_i_max␣>
                ␣0")
610     if (S_progresas_max < sigma * S_new_i_max
611         and P_progresas_max <= sigma * P_new_i_max
612         and E_progresas_max < sigma * E_new_i_max
613         and S_netiktis_max < sigma * S_new_i_max
```

```
614             and P_netiktis_max <= sigma * P_new_i_max):
615             return True
616         else:
617             return False
618 
619 def kph_vertes():
620     """
621     S, P, E suvidurkintų verčių skaičiavimas
622     :return:  [S_vid, P_vid, E_vid]
623     """
624     S_kph_i = (S_k_i + S_new_i) / 2
625     P_kph_i = (P_k_i + P_new_i) / 2
626     E_kph_i = (E_k_i + E_new_i) / 2
627     return [S_kph_i, P_kph_i, E_kph_i]
628 
629 def i_atsakas(P_k_i):
630     """
631     Atsako skaičiavimas
632     :param P_k_i: P
633     :return: atsakas
634     """
635     n_e = 1
636     Farad = 96485.3365
637     return ((n_e * Farad * D_P(0))*(4*P_k_i[1]-P_k_i[2])) / (2*h)
638 
639 def i_isvestine(i_t_km1, i_t_k, i_t_kp1, t_km1, t_k, t_kp1, h):
640     """
641     Atsako duomenu taškų išvestinės skaičiavimas
642     :param i_t_km1: atsakas 1
643     :param i_t_k: atsakas 2
644     :param i_t_kp1: atsakas 3
645     :param t_km1: laiko momentas 1
646     :param t_k: laiko momentas 2
647     :param t_kp1: laiko momentas 3
648     :param h: koordinatės žingsnis
649     :return: atsako išvestinė taške 2
650     """
651     a = i_t_km1 * (2*t_k - t_k - t_kp1) / (2*h**2)
652     b = i_t_k * (2*t_k - t_km1 - t_kp1) / (h**2)
653     c = i_t_kp1 * (2*t_k - t_km1 - t_k) / (2*h**2)
654     return a - b + c
655 
656 
657 sigma = masininis_tikslumas * 100  # S_new_i, P_new_i, E_new_i
        santykinės paklaidos tikslumo parametras
658 T = 1000  # Maksimali modeliavimo trukmė (sekundės)
659 k = 0  # Laiko kintamojo pradinė vertė
660 vykdymo_pabaiga = False
661 augimo_kampas_max_t = 0
662 augimo_kampas_max = 0
663 augimo_kampas = 0
664 atsakas = []
665 atsakas_t = []
666 atsako_santykinis_tikslumas = 1e-3
667 X_asis = np.linspace(0, x(N), N + 1)
```

```
668 dir = "outas"
669 while t(k) <= T and not vykdymo_pabaiga:
670     # spausdink_iteracijos_antraste(k, T)
671     if k == 0:
672         S_k_i = S_pradinis_masyvas()
673         P_k_i = P_pradinis_masyvas()
674         E_k_i = E_pradinis_masyvas()
675         E_old_i = E_k_i
676     else:
677         S_k_i = S_dabartinis(S_k_i, S_kph_i)
678         P_k_i = P_dabartinis(P_k_i, P_kph_i)
679         E_k_i = E_dabartinis(E_k_i, E_kph_i)
680     S_new_i = S_artiniu_masyvas(k, S_k_i, E_old_i)
681     P_new_i = P_artiniu_masyvas(k, P_k_i, E_old_i)
682     E_new_i = E_artiniu_masyvas(k, S_new_i, P_new_i, E_k_i)
683     tikrink_artinius(S_k_i, P_k_i, E_k_i, S_new_i, P_new_i, E_new_i)
684     if k != 0:
685         [S_progresas_max, P_progresas_max, E_progresas_max,
              S_netiktis_max, P_netiktis_max] = ivertink_progresus(k,
              S_new_i, P_new_i, E_new_i, S_k_i, P_k_i, S_old_i, P_old_i,
              E_old_i)
686         if (tenkina_stabdymo_kriterijai(sigma, S_new_i, P_new_i,
              E_new_i, S_progresas_max, P_progresas_max, E_progresas_max,
               S_netiktis_max, P_netiktis_max)):
687             print("pabaiga")
688             break
689     [S_old_i, P_old_i, E_old_i] = [S_new_i, P_new_i, E_new_i]
690     [S_kph_i, P_kph_i, E_kph_i] = kph_vertes()  # 2.5.1
691
692     atsakas.append(i_atsakas(P_k_i))
693     atsakas_t.append(t(k))
694     if k >= 3: # yra bent 3 elementai: 0, 1, 2.
695         augimo_kampas = i_isvestine(atsakas[k-2], atsakas[k-1],
              atsakas[k], t(k-2), t(k-1), t(k), h)
696         if augimo_kampas_max < augimo_kampas:
697             augimo_kampas_max_t = t(k)
698             augimo_kampas_max = augimo_kampas
699         if (augimo_kampas_max > 0):
700             atsako_santykis = augimo_kampas/augimo_kampas_max
701             if (atsako_santykis < atsako_santykinis_tikslumas):
702                 vykdymo_pabaiga = True
703
704     # Duomenų išsaugojimas
705     if (k % 100 == 0 and augimo_kampas_max > 0) or (vykdymo_pabaiga):
706         filename = F"t_{t(k)}_i_{atsakas[k]}_tiMax_{
              augimo_kampas_max_t}_as_{atsako_santykis}"
707         fig = plt.figure()
708         fig.set_size_inches(30, 5)
709         fig.subplots_adjust(bottom=0.07, left=0.045, top=0.975, right
              =0.975)
710         plt.subplot(1, 3, 1)
711         plt.plot(X_asis, S_new_i)
712         plt.subplot(1, 3, 2)
713         plt.plot(X_asis, P_new_i)
714         plt.subplot(1, 3, 3)
```

```
715             plt.plot(X_asis, E_new_i)
716             plt.savefig(F"{dir}/{filename}.png")
717             plt.close('all')
718             with open(F"{dir}/{filename}.csv", "a") as output:
719                 for i in range(0, N + 1):
720                     output.write(F"{X_asis[i]},{S_new_i[i]},{P_new_i[i]},{
                        E_new_i[i]}\n")
721             fig = plt.figure()
722             plt.plot(atsakas_t, atsakas)
723             plt.savefig(F"{dir}/atsakas.png")
724             plt.close('all')
725         with open(F"{dir}/atsakas.csv", "a") as output:
726             output.write(F"{atsakas_t[k]},{atsakas[k]}\n")
727
728         k += 1 # Sekantis laiko momentas
729
730 print(F"Vykdymo pabaiga. Atsakas {atsakas[k-1]}, t_max = {
        augimo_kampas_max_t}")
```