KAUNO TECHNOLOGIJOS UNIVERSITETAS

ELEKTROS IR ELEKTRONIKOS FAKULTETAS

Deividas Baranskas

MODELIAVIMO APLINKOS, BIOTECHNOLOGINIŲ PROCESŲ ANALIZEI IR TYRIMAMS, SUKŪRIMAS

Baigiamasis magistro projektas

Vadovas
Prof. Rimvydas Simutis

KAUNAS, 2017

KAUNO TECHNOLOGIJOS UNIVERSITETAS

ELEKTROS IR ELEKTRONIKOS FAKULTETAS
AUTOMATIKOS KATEDRA

MODELIAVIMO APLINKOS, BIOTECHNOLOGINIŲ PROCESŲ ANALIZEI IR TYRIMAMS, SUKŪRIMAS

Baigiamasis magistro projektas

Studijų programos pavadinimas (kodas 621H66001)

Vadovas
Prof. Rimvydas Simutis

Recenzentas
Doc. Renaldas Urniežius

Projektą atliko
Deividas Baranskas

KAUNAS, 2017

KAUNO TECHNOLOGIJOS UNIVERSITETAS

Elektros ir elektronikos fakultetas

(Fakultetas)

Deividas Baranskas

(Studento vardas, pavardė)

621H66001

(Studijų programos pavadinimas, kodas)

Modeliavimo aplinkos, biotechnologinių procesų analizei ir tyrimams, sukūrimas

AKADEMINIO SĄŽININGUMO DEKLARACIJA

20 ___ m. _____ ____ d.

Kaunas

Patvirtinu, kad mano **Deivido Baransko** baigiamasis projektas tema „Modeliavimo aplinkos, biotechnologinių procesų analizei ir tyrimams, sukūrimas“ yra parašytas visiškai savarankiškai, o visi pateikti duomenys ar tyrimų rezultatai yra teisingi ir gauti sąžiningai. Šiame darbe nei viena dalis nėra plagijuota nuo jokių spausdintinių ar internetinių šaltinių, visos kitų šaltinių tiesioginės ir netiesioginės citatos nurodytos literatūros nuorodose. Įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs.

Aš suprantu, kad išaiškėjus nesąžiningumo faktui, man bus taikomos nuobaudos, remiantis Kauno technologijos universitete galiojančia tvarka.

(vardą ir pavardę įrašyti ranka)	*(parašas)*

Baranskas, Deividas. Modeliavimo aplinkos, biotechnologinių procesų analizei ir tyrimams, sukūrimas. *Valdymo systemų magistro* baigiamasis projektas / vadovas prof. Rimvydas Simutis; Kauno technologijos universitetas, Elektros ir elektronikos fakultetas, automatikos katedra.

Mokslo kryptis ir sritis: Elektros ir elektronikos inžinerija, Technologiniai mokslai

Reikšminiai žodžiai: *mielių auginimo modelis, E. coli bakterijų auginimo modelis, Žinduolių ląstelių auginimo modelis, grafinė vartotojo sąsaja.*

Kaunas, 2017. 49 p.

SANTRAUKA

Šiame darbe atlikta literatūros analizė apie svarbių biotechnologinių procesų (mielių auginimo, E.coli bakterijų ir žinduolių ląstelių kultivavimo) technologinius ypatumus ir šių procesų matematinio modeliavimo principus Pateikta mielių klasifikacija, jų augimo fazės, matematinis modelis. Išanalizuoti E. coli bakterijų kultivavimo aspektai, bakterijų biologija bei auginimo matematinis modelis. Aprašyti žinduolių ląstelių auginimo biotechnologiniai procesai, ir šių procesų matematinis modeliavimas. Pateikta bioreaktorių klasifikacija. Visų trijų kultūrų kultivavimo procesų matematiniai modeliai realizuoti *Matlab/Simulink* aplinkoje. Kiekvienam minėtam modeliui realizuoti sukurta vartotojui patogi grafinė sąsaja. Sukurtos schemos svarbių kultivavimo proceso kintamųjų reguliavimui, kurios leidžia pačiam vartotojui modeliuoti įvairius kultivavimo režimus. Mielių auginimo proceso modelis panaudotas reguliuojant ištirpusio deguonies koncentraciją terpėje, keičiant masės perdavimo iš dujinės į skystąją fazę koeficientą (manipuliuojamas oro srautas ir maišyklės apsukos). E. coli auginimo proceso modelis panaudotas iliustruoti gliukozės koncentracijos reaktoriuje reguliavimo principą, keičiant gliukozės pamaitinimo srautą. Žinduolių auginimo proceso modelis panaudotas glutamino koncentracijos terpėje reguliavimui, keičiant glutamino pamaitinimo srautą. Modeliavimo rezultatai pateikiami esant įvairioms kultivavimo sąlygoms, kurias gali pasirinkti vartotojas. Darbo pabaigoje trumpai pristatyti darbo rezultatai ir išvados.

Baranskas, Deividas. Development of Modeling Environment for Bioprecesses Analysis and Investigation: *Master's* thesis in *Control Technologies degree* / supervisor assoc. prof. Rimvydas Simutis. Kaunas University of Technology, Faculty of Electrical and Electronics Engineering, department of automation

Research area and field: Electrical and Electronics Engineering, Technological Sciences

Key words: *modeling of growing yeast, modeling of growing E.coli bacteria, modeling of growing mammalian cells culture, graphical modeling environment*

Kaunas, 2017. 49 p.

SUMMARY

In this work there is conducted analysis of the literature about technological peculiarities of important biotechnological processes (yeast growing, E.coli bacteria and mammalian cells culture) and mathematical modelling principles of these processes. It is presented yeast classification, phases of their growth and the mathematical model. It is analysed aspects of E.coli bacteria culture, biology of bacteria and mathematical model of growth. It is described biotechnological processes of mammalian cells growing and mathematical modelling of these processes. It is presented classification of the bioreactors. Mathematical models of culturing all three cultures are implemented in the Matlab/Simulink environment. For the realization of each above-mentioned model it is created a user-friendly graphical interface. It is made schemes for the regulation of important culture process variables, which allows modelling various culture regimes for the user oneself. Model of yeast growing process is used for regulating dissolved oxygen concentration in the medium, changing the coefficient of the mass transfer from gaseous to the liquid (by manipulating the air flow and the stirrer speed). E.coli growing process model is used to illustrate the control principle of glucose concentration in the reactor while changing glucose feeding stream. Process model of growing mammalian cells is used for regulation of glutamine concentration in the medium while changing glutamine feeding stream. Results of modelling are presented under various culture conditions which can be chosen by the user. At the end of work there are presented results of the work and conclusions.

TURINYS

SANTRUMPŲ IR ŽENKLŲ AIŠKINIMO ŽODYNAS

DNR – deoksiribonukleorūgštis;

FDA (angl. *Food and Drug Administration*) – JAV maisto ir vaistų administracija;

CD3 (OKT3) – baltymas, kuris yra skiriamasis limfocitų bruožas;

CHO (angl. *chinese hamster ovary*) – kiniško žiurkėno kiaušidės;

GUI (angl. *Graphical User Interfaces*) – grafinė vartotojo sąsaja.

ĮVADAS

Biotechnologija, tai mokslo kryptis, kuri sujungia biologijos ir technologijų galimybes. Kitaip tariant tai biologinių procesų naudojimas medicinoje, pramonėje, žemės ūkyje. Daugėjant žmonių skaičiui ir tobulėjant technologijoms, efektyvus biologinių procesų panaudojimas tampa vis svarbesniu klausimu, nes jų nauda suteikia nemažai privalumų: maisto produkto kokybės gerinimas, ligų išvengimas bei gydymas, sveikatos apsauga. Biotechnologijų klestėjimas sukuria naujų darbo vietų [1].

Didėjant konkurencijai, bioprocesų tinkamas optimizavimas sumažina gaunamos produkcijos kainą, jos nukrypinus nuo norimų rezultatų bei visiškai atitinka saugumo keliamus reikalavimus, padidina proceso kokybę. Norint pasiekti šiuos rodiklius būtina sukurti gerą proceso matematinį modelį. Naudojant proceso matematinį modelį galima tinkamai parinkti kultivavimo režimus, keisti pradines sąlygas, matyti grafines kintamųjų priklausomybes. Matematinio modelio pagalba gauti rezultatai gali būti panaudojami optimizuojant realius procesus. Toks būdas leidžia parinkti optimalius medžiagų kiekius, be didelių išlaidų ir pakankamai greitai atlikti modelio savybių ir elgesio tyrimą įvairiuose situacijose [2].

Toliau išsamiau nagrinėjami trys svarbūs šiuolaikinei biotechnologijai procesai:

- Mielių auginimo procesas. Jis naudojamas kulinarijoje, alkoholinių gėrimų fermentavimui, medicinoje;
- E. coli bakterijų auginimo procesas. E. Coli bakterijos naudojamos rekombinantinių baltymų ir vaistų gamyboje. Paprastai šios bakterijos nepavojingos žmogui, tačiau dalis jų gali sukelti tokias ligas kaip viduriavimą, šlapimo ir kvėpavimo takų infekcijas, todėl jų kultivavimo procesai turi būti vedami saugiai [23].
- Žinduolių ląstelių auginimo procesas. Šis procesas daugiausiai taikomi įvairių biomedicininių produktų gamybai. Išgavus baltymus iš šių ląstelių jas galima panaudoti įvairioms ligoms gydyti.

Darbo tikslas – sukurti vartotojui draugišką aplinką tipiniams biotechnologiniams procesams modeliuoti.

Darbo uždaviniai:

- išanalizuoti tipinius kultivavimo procesus: kepimo mielių, E. coli bakterijų, žinduolių ląstelių;
- parinkti tipinius modelius jų elgsenos aprašymui;
- modelius realizuoti *Simulink* aplinkoje – sukuriant patogią vartotojo sąsają;
- atlikti įvairių procesų režimų modeliavimą ir valdymo sistemų testavimą.

1. MIELIŲ AUGINIMO PROCESAS

1.1. Mielių klasifikacija

Mielės buvo pradėtos naudoti daug ankščiau nei buvo ištirtos 1680 metais. Šiais laikais jau turbūt retas kuris nebūtų vienaip ar kitaip susidūręs su mielėmis, kai jos yra taip dažnai yra vartojamos. Plačiausiai jos naudojamos kulinarijoje įvairioms tešloms kildinti, alkoholiniams gėrimams fermentuoti, o pastaruoju metu vis dažniau vartojamas kaip maisto papildas.

Mielės, kaip augalai bei gyvūnai yra eukariotiniai organizmai, kurie priklauso grybų karalystei. Mielių baltymuose yra visos amino rūgštys reikalingos gyvūno bei žmogaus organizmui. Jose ypač gausu B grupės vitaminų, E, PP fermentų ir kitų vertingų medžiagų, didinančių organizmo atsparumą infekcijai bei gerinančių virškinimą [3].

Paprastai mielės skirstomos į aktyvias ir neaktyvias. Neaktyvios neišskiria anglies dioksido. Jos būna džiovintos ir naudojamos kaip skonio ir aromato komponentai ar maisto papildai. Aktyvios mielės naudojamos fermentacijai.

Pagal rūgimą mielės skirstomos į žemutinio ir aukštutinio rūgimo mieles. Aukštutinio rūgimo mielės išsiskiria ląstelių mažumu bei silpnu gebėjimu flokuliuoti. Jos geriausiai dauginasi esant 25...27 ºC temperatūrai. Žemutinio rūgimo mielės, priešingai nei aukštutinio, pasižymi daug didesniu dydžiu bei geresnėmis flokuliacinėmis savybėmis. Jos priklauso šalto rūgimo mielėms ir aktyviai dauginasi 5...10 ºC temperatūroje [4].

Mielės dauginasi mitozės arba pumpuravimo būdu. Jos gali augti ne tik esant deguoniui bet ir tada, kai jo nėra. Toks mielių dauginamasis yra anaerobinis procesas, kur deguonis ir cukrus dėl oksidacinės medžiagų apykaitos paverčiamas anglies dioksidu ir veiksmingam mielių ląstelių augimui reikalinga laisvąja energija. Paprastai mielių geram augimui reikia didelio kiekio deguonies iš oro [5].

Technologinis mielių auginimo procesas susideda iš trijų etapų:

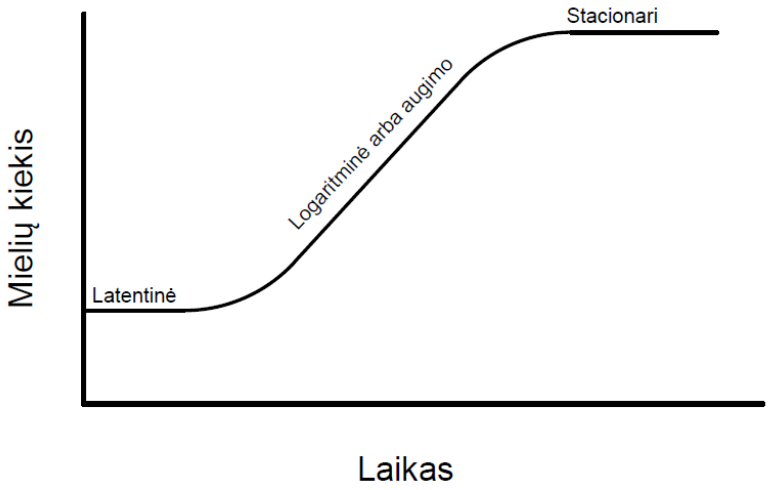
1. švarios mielių kultūros augimo;
2. natūraliai švarios mielių kultūros augimo;
3. prekinių mielių auginimo [6].

1.2. Mielių augimo fazės

Mielių augimo fazes galima suskirstyti į tris pagrindines grupes:

- latentinė fazė (angl. *lag phase*);
- logaritminė arba augimo fazė (angl. *log phase*);
- stacionari fazė (angl. *stationary phase*).

Mielių augimas vyksta pagal 1.1 paveikslėlyje pateiktą charakteristiką.



1.1 pav. Mielių augimo fazės

Latentinės fazės metu mielės aklimatizuoja misą ir paruošia atgaminti ir vartoti didelį kiekį cukraus. Kaip ir su visais etapais taip ir su šiuo ciklo (fazės) laikas nėra tiksliai žinomas. Ne visos mielės atitinka tą patį procesą tuo pačiu metu, bet bendrai apibendrinti fazes yra įmanoma [5].

Šioje fazėje labai svarbus yra deguonis. Jis naudojamas mielių sterolių sintezei ir nesočiosioms riebalų rūgštims, kurios yra būtinos augimui. Be deguonies nevyks šie procesai ir mielių augimas bus labai ribotas. Steroliai ir riebalų rūgštis taip pat yra labai svarbi ląstelės membranos struktūros dalis, kuri reaguoja į išorinius ir į vidinius įtempius. Latentinėje fazėje mielės nesidaugina, o tik smarkiai didėja jos tūris. Vykstantys procesai naudoja vidines ląstelių energijos sąnaudas ir vyksta tol, kol jos membrana tampa pralaidi cukrui.

Logaritminėje arba augimo fazėje mielės eksponentiškai auga. Mielės, kurios buvo paruoštos latentinėje fazėje, aktyviai dauginasi. Tai pati produktyviausia mielių auginimo fazė, kurioje vyksta aktyvus biomasės augimas [6]. Jos dauginasi pumpuravimo būdu: naujai atsiradusios ląstelės yra identiškos senesnėms. Mielių kokybiniai ir kiekybiniai rodikliai labai priklauso nuo visų įvedamų maistinių medžiagų balanso. Augimas vyksta tol kol yra pakankamai deguonies ir maistingųjų medžiagų. Kai visas deguonis yra sunaudojamas – prasideda fermentavimo ciklas. Mielių ląstelės perdirba skystyje esantį cukrų į alkoholį, CO2 ir kvapiąsias bei aromatines medžiagas.

Dauguma aromatinių ir kvapiųjų medžiagų junginių yra šalutiniai produktai, todėl daug alaus daryklų bando apriboti mielių augimą ir taip sumažinti esterio sintezę. Vienas iš mažinimo būdų yra laikyti fermentus šaltai.

Stacionari fazė yra paskutinė, kurioje ląstelių skaičius nesikeičia. Šioje fazėje mielių populiacija pasiekia maksimalų tankį ir sunaudoja paskutinį likusį cukrų [5]. Išnaudojus visą

cukrų, fermentavimo pabaigoje prasideda sedimentacija. Mielės sulimpa tarpusavyje ir sudaro akimi matomas daleles, kurios nusėda indo apačioje, šis procesas vadinamas flokuliacija. Šiuo metu mielės kaupia glikogeną, kuris bus naudojamas ateityje mielių dauginimuisi ir aprimsta, pereidamos į negatyvią būseną. Sedimentacijos procesas vyksta skirtingu greičiu, priklausomai nuo mielių rūšies [7].

Kaip kuriuos šaltiniuose išskiriama dar viena fazė, t.y. mirties fazė, kurioje mielės žūva. Taip gali atsitikti dėl maisto, medžiagų trūkumo, nukrypusios leistinos temperatūros ar dėl kitų pavojingų sąlygų. [8]

1.3.Mielių matematinis auginimo proceso modelis

1.3.1. Kinetinis modelis

Mielių augimą galima aprašyti trimis būdais:

- aerobinis biomasės augimas naudojant gliukozę: $S + C \overset{\mu_s^o}{\rightarrow} X + C$; (1.1)
- anaerobinis biomasės augimas naudojant gliukozę (rūgimas) $S \overset{\mu_s^f}{\rightarrow} X + E + C$; (1.2)
- aerobinis biomasės augimas naudojant etanolį $E + O \overset{\mu_e^0}{\rightarrow} X + C$. (1.3)

Kur: *S* – gliukozės koncentracija, *O* – deguonies koncentracija, *X* – biomasės koncentracija, *E* – etanolio koncentracija, *C*– anglies dioksido koncentracija, μ_s^o, μ_s^f, μ_e^0 augimo charakteristikos.

Rūgimas ir aerobinis augimas yra vienas kitam konkuruojantys procesai. Ši konkurencija paremta ląstelių kvėpavimo pajėgumu. Jei deguonies įsisavinimo pajėgumas yra didesnis už tiekiamos gliukozės srautui oksiduoti reikalingą kiekį, tai dalis deguonies yra sunaudojama etanolio (1.3), esančio terpėje oksidavimui. Kai momentinis deguonies suvartojimo pajėgumas yra nepakankamas, tada dalis gliukozės naudojama aerobiniam augimui oksiduojant gliukozę, o (1.1), likusi gliukozės dalis rūgimo reakcijos metu paverčiama etanoliu. (1.2) [9].

1.3.2. Matematinio modelio lygtys

Matematinis mielių fermentacijos modelis sudaromas taikant masių balanso lygtis, darant prielaidą, kad visi komponentai reaktoriuje išsimaišo tolygiai. Tada matematinio modelio lygtys yra:

Biomasės koncentracija:

$$\frac{dX}{dt} = \left(\mu_s^o + \mu_s^f + \mu_e^o - D\right)X. \quad (1.4)$$

kur: $\mu_s^o -$ santykinis biomasės augimas aerobinėse sąlygose naudojant gliukozę; $\mu_s^f -$ santykinis biomasės augimas anaerobinėse sąlygose naudojant gliukozę; $\mu_e^o -$ santykinis biomasės augimas aerobinėse sąlygose naudojant etanolį.

Gliukozės (cukraus) koncentracija:

$$\frac{dS}{dt} = \left(-\frac{\mu_s^o}{Y_{X/S}^o} - \frac{\mu_s^f}{Y_{X/S}^f}\right)X + (S_f - S)D; \quad (1.5)$$

kur: $Y_{X/S}^o -$ biomasės/gliukozės išeigos koeficientas aerobinėse sąlygose, g/g; $Y_{X/S}^f -$ biomasės/gliukozės išeigos koeficientas anaerobinėse sąlygose, g/g; $S_f -$ substrato koncentracija pamaitinimo tirpale, g/l.

Etanolio koncentracija:

$$\frac{dE}{dt} = \left(\frac{\mu_s^f}{Y_{X/E}^f} - \frac{\mu_e^o}{Y_{X/E}^{oe}}\right)X - DE; \quad (1.6)$$

kur: $Y_{X/E}^f -$ biomasės/etanolio išeigos koeficientas anaerobinėse sąlygose, g/g; $Y_{X/E}^{oe} -$ biomasės/etanolio išeigos koeficientas aerobinėse sąlygose kai dalis deguonies yra sunaudojama etanolio, g/g.

Ištirpusio deguonies koncentracija:

$$\frac{dO}{dt} = \left(-\frac{\mu_s^o}{Y_{X/O}^o} - \frac{\mu_e^o}{Y_{X/E}^{oe}}\right)X - DO + OTR; \quad (1.7)$$

kur: $Y_{X/O}^o -$ biomasės/deguonies išeigos koeficientas aerobinėse sąlygose, g/g; $Y_{X/E}^{oe} -$ biomasės/deguonies išeigos koeficientas aerobinėse sąlygose kai dalis deguonies yra sunaudojama etanolio, g/g.

Ištirpusio anglies dioksido koncentracija:

$$\frac{dC}{dt} = \left(\frac{\mu_s^o}{Y_{X/C}^o} + \frac{\mu_s^f}{Y_{X/C}^f} + \frac{\mu_e^o}{Y_{X/C}^{oe}}\right)X - DC + CTR, \quad (1.8)$$

kur: $Y_{X/C}^o -$ biomasės/anglies dioksido išeigos koeficientas aerobinėse sąlygose, g/g; $Y_{X/C}^f -$ biomasės/anglies dioksido išeigos koeficientas anaerobinėse sąlygose, g/g; $Y_{X/C}^{oe} -$ biomasės/anglies dioksido išeigos koeficientas aerobinėse sąlygose kai dalis deguonies yra sunaudojama etanolio, g/g.

kur: D – yra skiedimo greitis:

$$D = \frac{F}{V} = \frac{\text{įtekančio srauto greitis}}{\text{kiekis}} \quad (1.9)$$

Maitinimo metu, įtekančio srauto kiekis aprašomas lygtimi:

$$\frac{dV}{dt} = DV. \quad (1.10)$$

Deguonies tiekimo greitis aprašomas:

$$OTR = k_L \mathrm{a}(O^* - O), \quad (1.11)$$

o anglies dioksido tiekimo greitis:

$$CTR = k_L \mathrm{a}(C - C^*)c_{CO2} \quad , \quad (1.12)$$

kur: $k_L a$ – masės perdavimo iš dujinės į skystąją fazę koeficientas, 1/h; c_{CO2} – CO_2 tirpumo konstanta, O^* ir C^* atitinkamai deguonies ir anglies dioksido maksimalūs tirpumo koeficientai skystoje kultivavimo terpėje, g/l [9].

Biomasės augimo greičio išraiška aerobinėse sąlygose:

$$\mu_s^o = \frac{\mu_{\max\_so} S}{K_s^o + S} \frac{O_2}{K_{O_2} + O_c} \qquad kai\ S > 0; \quad (1.13)$$

kur: $\mu_{\max\_so}$ – maksimali biomasės augimo greičio išraiška aerobinėse sąlygose, g/l; K_s^o – Mono koeficientas, juo įvertinimas $\mu_{\max\_so}$ priklausomybės nuo *S* intensyvumas; K_{O_2}– limitavimas deguonimi.

Biomasės augimo greičio išraiška anaerobinėse sąlygose:

$$\mu_s^f = \frac{\mu_{\max\_sf} S}{K_s^f + S} \frac{O_2}{K_{O_2} + O_c} \qquad kai\ S > 0{,}5; \quad (1.14)$$

kur: $\mu_{\max\_sf}$ – maksimali biomasės augimo greičio išraiška anaerobinėse sąlygose, g/l; K_s^f – Mono koeficientas, juo įvertinimas $\mu_{\max\_sf}$ priklausomybės nuo *S* intensyvumas.

Augimo greitis naudojant etanolį anaerobinėse augimo sąlygose

$$\mu_e^o = \frac{\mu_{\max\_eo} S}{K_e^o + S} \frac{O_2}{K_{O_2} + O_c} \qquad kai\ S < 0{,}5; \quad (1.15)$$

kur: $\mu_{\max\_eo}$ – maksimali biomasės augimo greičio išraiška aerobinėse sąlygose, kai dalis deguonies yra sunaudojama etanolio, g/l; K_e^o – Mono koeficientas, juo įvertinimas $\mu_{\max\_eo}$ priklausomybės nuo *S* intensyvumas.

2. E. COLI BAKTERIJŲ AUGINIMO PROCESAS

2.1. E. Coli bakterija

Escherichia coli (dažnai trumpinama iki E. coli) tai lazdelės formos bakterijos, kurios normaliai gyvena žmogaus ir gyvūnų žarnyne. Šios bakterijos sudaro apie 0,1% viso žarnyno augmenijos bei yra naudingos tuo, kad gamina vitaminą K2 ir gali apsaugoti nuo patogeninių bakterijų kolonizacijos žarnyne [10].

Escherichia coli bakterijos, pirmą kartą buvo atrastos 1885 metais žmogaus storojoje žarnoje. Šias bakterijas atrado vokiečių bakteriologas Theodar Escherich, pradžioje pavadindamas tiesiog coli bakterijomis. Vėliau pagerbiant atradusio mokslininko atminimą, coli bakterijos pakeistos Escherichia coli. Dažnai minima, kad E. Coli yra labiausiai ištirti laisvai gyvenantys organizmai, šių bakterijų atmainų nustatyta daugiau nei 700. Išskiriant bakterijas „O“ ir „H“ antigenus, jie padeda išskirti atskiras atmainas [11].

E. coli bakterijų yra daug tipų ir dauguma jų nekenksmingos, tačiau yra dalis, kuri vis dėlto gali pakenkti žarnynui. Pagrindiniai simptomai užsikrėtus šia bakterija yra viduriavimas, pilvo skausmas, karščiavimas. Sunkesnė simptomų forma gali sukelti viduriavimą krauju, dehidrataciją ar net inkstų nepakankamumą. Užsikrėsti E. coli bakterijomis galima per netinkamai apdorotą maistą, užterštą vandenį, tiesiogiai kontaktuojant su užkrėstu žmogumi ar gyvūnais [12].

Į natūralią aplinką E coli bakterijos patenka per išmatas, kuriose pirmąsias 3 dienas aerobinėse sąlygose vyksta intensyvus jų dauginimas, po to jų kiekis labai lėtai mažėja. Jos lauko sąlygomis sugeba išlikti tam tikrą apribotą laiką, o tai palengvina mokslinius tyrinėjimus su šiomis bakterijomis [13].

E. coli buvo intensyviai tiriamos daugiau nei 60 metų, kadangi šios bakterijos gali būti lengvai auginamos ir kultivuojamos nebrangioje laboratorijoje. Šių bakterijų skilimui reikalinga energija gaunama iš cheminių reakcijų, kurios turi apibrėžti anglies dioksidą ir reikalingą energijos šaltinį. E. coli augimui reikalingi šie organiniai junginiai: gliukozė, amonio fosfatas, natrio chloridas, magnio sulfatas, kalio sulfatas ir vanduo. Escherichia coli yra plačiausiai ištirtas prokariotinis organizmas, labai svarbus biotechnologijos ir mikrobiologijos srityse [11].

2.2.E. Coli bakterijų biologija

E. coli bakterijos yra lazdelės formos, kurios yra 2,0 mikrometrų (μm) ilgio, 0,25 - 1,0 μm diametro bei 0,6 – 0,7 μm^3 dydžio [10].

Bakterijoms naudojamas gramo dažymo metodas, kadangi jos turi peptidoglikano sluoksnį bei išorinę membraną. Naudojant šį būda, pagal chemines ir fizikines savybes bakterijų ląstelių sienelių savybes atskiriamos gram – teigiamų ir gram – neigiamų bakterijų grupės. Šiame procese

svarbiausia, kad gram – teigiamos ir gram – neigiamos bakterijos skirtingai reaguoja į dažus nuplaunantį skystį, nes gram – teigiamas bakterijos dengiančiame peptidoglikano sluoksnyje vyksta dehidratacija ir kristaliniai dažai „įstringa“ ląstelėje, o tuo tarpu gram – neigiamose bakterijos skystis pradžioje ištirpdo riebalų sluoksnį, o plonas toliau sekantis peptidoglikano sluoksnis nėra pajėgus sulaikyti dažus [14] [15].

Bakterijos gali gyventi ant įvairių substratų ir anaerobinėmis sąlygomis, kadangi naudoja mišrią rūgšties fermentaciją, gaminant laktatą, etanolį, acetatą ir anglies dioksidą. Naudojant mišrią rūgšties fermentaciją yra gaminamos vandenilio dujos, kurių kiekis E. coli bakterijoms gali būti per mažas, jei jos gyvena su organizmais kuriems reikalingas vandenilis, pvz.: metanogenais [14].

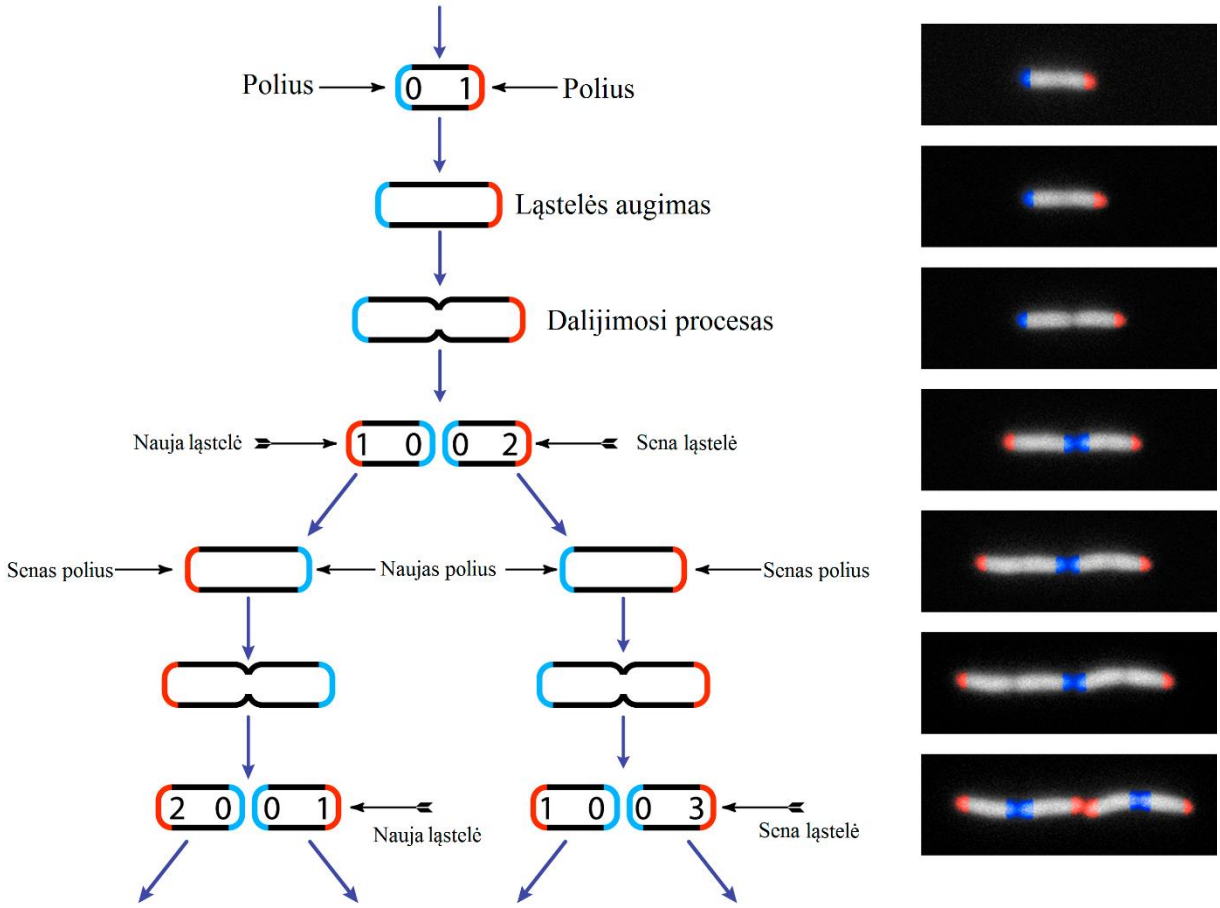
Optimalus bakterijų augimas vyksta, kai žmogaus kūno temperatūra siekia 37 °C, nors kai kuriose laboratorijose gali būti naudojama net iki 49 °C. Augimas gali vykti aerobinio ir anaerobinio kvėpavimo metu, naudojant daug oksidacijos – redukcijos reakcijų. Paprastai E. coli yra fakultatyviniai aerobiniai organizmai, kadangi augant naudoja deguonį. Tačiau jos gali augti ir anaerobinėmis sąlygomis. Nesant deguonies bakterijos gauna energijos alternatyviniu būdu, naudojant fermentaciją ar anaerobinio kvėpavimo būdą. Augimas, kai nėra deguonies yra didelis pranašumas prieš kitas baterijas, nes jos gali išgyventi ir vandenyje [13].

E. coli dauginimosi ir vystymosi ciklas vyksta įprastai, taip kaip ir kitų bakterijų. Bakterijos masė augimo metu didėja, bet palyginti lėčiau negu jos paviršius. Pasiekus tam tikrą, jos rūšiai būdingą masės ir paviršiaus santykį, bakterija nustoja augus, tuomet ji pasidalija pusiau. Šį ciklą galima susiskirstyti į pagrindinius periodus:

- B periodas – didėja bakterijos ląstelės masė;
- C periodas – vyksta DNR replikacija;
- D periodas – vyksta dalijamasis pusiau.

Motininė bakterijos ląstelė, kuri virsta dviem dukterinėmis ląstelėmis, ne miršta bet atsinaujina. Atsinaujinimo esmę sudaro tai, kad ląstelių lyginimasis paviršius joms pasidalijus, pasidaro didesnis ir dėl to naujos ląstelės gali sparčiau imti maistą bei tuo pačiu greičiau augti, nors tai laiko tarpo tarp B ir C periodo neįtakoja [14].

2.1 paveikslėlyje pavaizduota E. coli bakterijos dalijamose. Kaip matyti, po kiekvieno pasidalijimo ląstelėje atsiranda du nauji ląstelės poliai, kurie yra sužymėti spalvomis. Nauji poliai pavaizduota mėlyna spalva, seni poliai raudona. Taip pat kiekvienam poliui rašomas skaičius, kurio dėka galime žinoti kiekvienos atsiradusios ląstelės amžių bei iš kokios ląstelės ji atsirado [15].



2.1 pav. E. coli bakterijos dalijamasis [adaptuota pagal 15].

2.3.E. coli auginimo matematinis modelis

Matematinis modelis kuriamas periodinio su pamaitinimu E. coli kultivavimo procesui. Šiuo atveju modelis bus kuriamas E. coli BL21 kultūrai.

Visų pirma prieš sukuriant E. coli proceso matematinį modelį yra priimamos tokios prielaidos:

a) Pagrindiniai šio biotechnologinio proceso produktai yra biomasė, anglies dioksidas, vanduo ir tam tikromis sąlygomis išsiskiriantis acetatas. Išskyrus acetatą, jokie kiti šalutiniai produktai nėra sukuriami dideliais kiekiais.

b) Substrato (gliukozės) metabolizmas gali vykti tiek aerobinėmis tiek anaerobinėmis sąlygomis. Tada atitinkamų normų ir kiekio koeficientai yra skirtingi.

c) Gliukozės sunaudojimo oksidacijai kiekis aprašomas panaudojant Mono priklausomybę. Taip pat papildomai įvertinamas acetato inhibicijos augimas.

d) Gliukozės oksidacijai įvertinimas butelio kaklelio efektas. Kai gliukozės koncentracija ir jos suvartojimo norma viršija kritinę vertę, tai dalis gliukozės paverčiama į acetatą,

vietoj to, kad būtų oksiduota į anglies dioksidą. Laikoma, kad acetato gamyba yra tiesiškai priklausanti nuo biomasės augimo greičio.

e) Ląstelėms negavus pakankamai pagrindinio substrato (t.y. kai santykinis substrato greitis yra mažesnis už kritinę vertę) acetatas gali būti sunaudojimas verčiant į anglies dioksidą ir metabolizuojamas tik aerobiniu būdu.

f) Butelio kaklelio dydis (kritinė substrato įsisavinimo vertė) yra laikomas priklausomu nuo ištirpusio deguonies koncentracijos, atsižvelgiant į Mono priklausomybę. Tačiau ištirpusio deguonies koncentracijos įtaka yra sumažinama iki minimumo, tada kai deguonies koncentracija turi būti palaikoma aukščiau apibrėžtos kritinės vertės.

g) Įvairios augimo tempo, acetato gamybos ir sunaudojimo variacijos žymiai nekeičia pagrindinės biomasės sudėties. Tokiu atveju prielaida būtų, kad biomasės augimas yra subalansuotas.

h) Biomasės fermentacija priklauso nuo gliukozės ir acetato sunaudojimo [16].

Nagrinėjamam E. coli su pamaitinimu kultivavimo procesui parenkamos modeliuojamų komponentų santykinių reakcijos greičių modeliai.

Visų pirma sudaroma santykinio gliukozės (substrato) greičio lygtis σ. Santykinio substrato sunaudojimo greičio modelis sudarytas:

$$\sigma = \sigma_{\max} \frac{S}{K_s + S} \frac{K_{si}}{K_{si} + S} \frac{K_{ai}}{K_{ai} + A} - K_{xx} X^2; \quad (2.1)$$

$\sigma = \sigma_{\max}$ maksimalus substrato sunaudojimo greitis,

$\frac{S}{K_s + S}$ substrato ribojimas,

$\frac{K_{si}}{K_{si} + S}$ substrato inhibicija,

$\frac{K_{ai}}{K_{ai} + A}$ acetato inhibicija,

$K_{xx} X^2$ biomasės inhibicija,

kur: K_s – Mono koeficientas, g/L; K_i – inhibicijos koeficientas, g/L; S – substrato koncentracija, g/L; A – acetato koncentracija, g/l.

Biomasės augimo greitis μ_s modeliuojamas atsižvelgiant į substrato greitį, σ:

$$\mu_s = \mu_{cr} Y_{xs} - m, \quad \text{jei } \mu_s > \mu_{cr}, \quad (2.2)$$

$$\mu_s = \mu_s S + \alpha_c Y_{xa} - m, \quad \text{jei } \mu_s \leq \mu_{cr}, \quad (2.3)$$

kur: μ_{cr} – kritinė substrato sunaudojimo greičio vertė, ją viršijus sunaudojamas substratas negali būti tiesiogiai oksiduojamas, l/h; Y_{xs} – biomasės/gliukozės išeiga g/g; Y_{xa} – biomasės/acetato išeiga g/g; m – gyvybinių funkcijų palaikymo koeficientas.

Acetato gamybos greičio modelis, α_p:

$$\alpha_p = (\mu_s - \mu_{cr})Y_{as}, \quad \text{jei } \mu_s > \mu_{cr}, \tag{2.4}$$

$$\alpha_p = 0, \quad \text{jei } \mu_s \leq \mu_{cr}, \tag{2.5}$$

Y_{as} – acetato/gliukozės išeiga g/g.

Acetato sunaudojimo greičio modelis, α_c:

$$\alpha_c = \left(1 - {}^{\mu_s}/_{\mu_{cr}}\right)\frac{A}{(A + K_a)}, \quad \text{jei } \mu_s \leq \mu_{cr}, \tag{2.6}$$

$$\alpha_c = 0, \quad \text{jei } \mu_s > \mu_{cr}, \tag{2.7}$$

Jeigu nėra gliukozės mišinyje (S=0 g/l) arba jos koncentracija labai maža, tai μ_s (biomasės augimas) reikšmė yra lygi nuliui, o α_c maksimali (2.5 lygtis). O kuo didesnė μ_s reikšmė, tai tuo mažesnis acetato sunaudojimo proporcingumo koeficientas α_c [14][15].

Šalutinio produkto naudojimo inhibiciją įvertina trečias kintamasis. Suminė santykinė acetato reakcijos greičio išraiška lygi jo pagaminto ir sunaudoto kiekio skirtumui:

$$\alpha = \alpha_p - \alpha_c. \tag{2.8}$$

Toliau sudaromos masės balanso lygtys. Pradedama nuo biomasės koncentracijos, X, g/l:

$$\frac{dX}{dt} = \mu_s X - \left(\frac{F}{V}\right)X, \tag{2.9}$$

kur: F – į bioreaktorių tiekiamų medžiagų srautas, l/h; V – kultivavimo terpės tūris, l.

Formulė FX/V yra įtekančio srauto skiedimo greitis.

Gliukozės (substrato) koncetracija kultivavimo procese, S, g/l:

$$\frac{dS}{dt} = -\sigma X - \frac{F}{V}S + \frac{F_s}{V}S_f, \tag{2.10}$$

kur: F_s – į bioreaktorių įtekantis gliukozės pamaitinimo srautas, l/h; S_f – įtekančiame sraute gliukozės koncentracija, g/l.

Formulė FS/V yra įtekančio srauto skiedimo greitis.

Acetato koncentracija:

$$\frac{dA}{dt} = \alpha X - \frac{F}{V}A, \tag{2.11}$$

kur: FA/V yra įtekančio srauto skiedimo greitis.

Terpės tūris:

$$\frac{dV}{dt} = F - F_{smp}. \tag{2.12}$$

Į bioreaktorių tiekiamų medžiagų srautas F, randamas iš:

$$F = F_s + F_b + F_{evp} + F_{co2}; \tag{2.13}$$

F_b – buferinio (šarminio) tirpalo tiekimo greitis (kai valdomas pH), l/h; F_{evp} – vandens garų išgaruotas kiekis, l/h; F_{co2} – anglies išmetimas su CO_2 ištekančiose dujose, l/h.

Kintamieji paminėti 2.14 formulėje atitinkamai randami:

$$F_b = Y_{bx}\mu XV, \quad (2.14)$$

kur: bazės/biomasės išeiga, g/g.

$$F_{CO_2} = -Y_{O_2x}OUR_V, \quad (2.15)$$

kur: Y_{O_2x} – CO_2/biomasės išeiga, g/g; OUR_V – deguonies sunaudojimo greitis, g/h.

$$F_{evp} = -k_eV, \quad (2.16)$$

kur: k_e – garavimo greitis.

Norint rasti OUR_V visų pirma randama deguonies sunaudojimas tūrio vienetui, o po to visam terpės tūriui:

$$OUR = Y_{ox}\mu X + \omega X, \quad (2.17)$$

$$OUR_V = OUR \cdot V, \quad (2.18)$$

kur: Y_{ox} – deguonies/biomasės augimo išeiga, g/g. [16][17].

3. ŽINDUOLIŲ LĄSTELIŲ AUGINIMO MODELIS

3.1.Apie žinduolių ląsteles

Žinduolių ląstelių panaudojimo poreikis baltymų gamybai nuolat auga. Tad labai svarbu joms sukurti palankias sąlygas kultivuoti (augti). Ląstelės yra auginamos *in vitro*, dirbtinai sukurtose ir kontroliuojamuose sąlygose, kurios privalo maksimaliai atitikti natūralaus augimo sąlygas. Pagrindinis kultivavimo tikslas yra ląstelių ir audinių padauginimas ir jų išskiriamų produktų panaudojimas sveikatos gerinimo tikslais [18]. Ląstelių auginimas *in vitro* prasidėjo nuo 1880 metų. Mokslininkai Ross Granville Harrrison ir Wilhelm Roux yra laikomi ląstelių kultivavimo pradininkais.

Viena iš biotechnologijos rūšių yra raudonoji biotechnologija. Raudonoji biotechnologija naudoja visus gyvus organizmus (gyvūnų ląstelės, mielės, bakterijos), kuriuos genų inžinerijos metodais modifikuoja taip, kad tokios ląstelės pradeda gaminti joms nebūdingas medžiagas (žmogaus baltymus, nukleino rūgštis) [19].

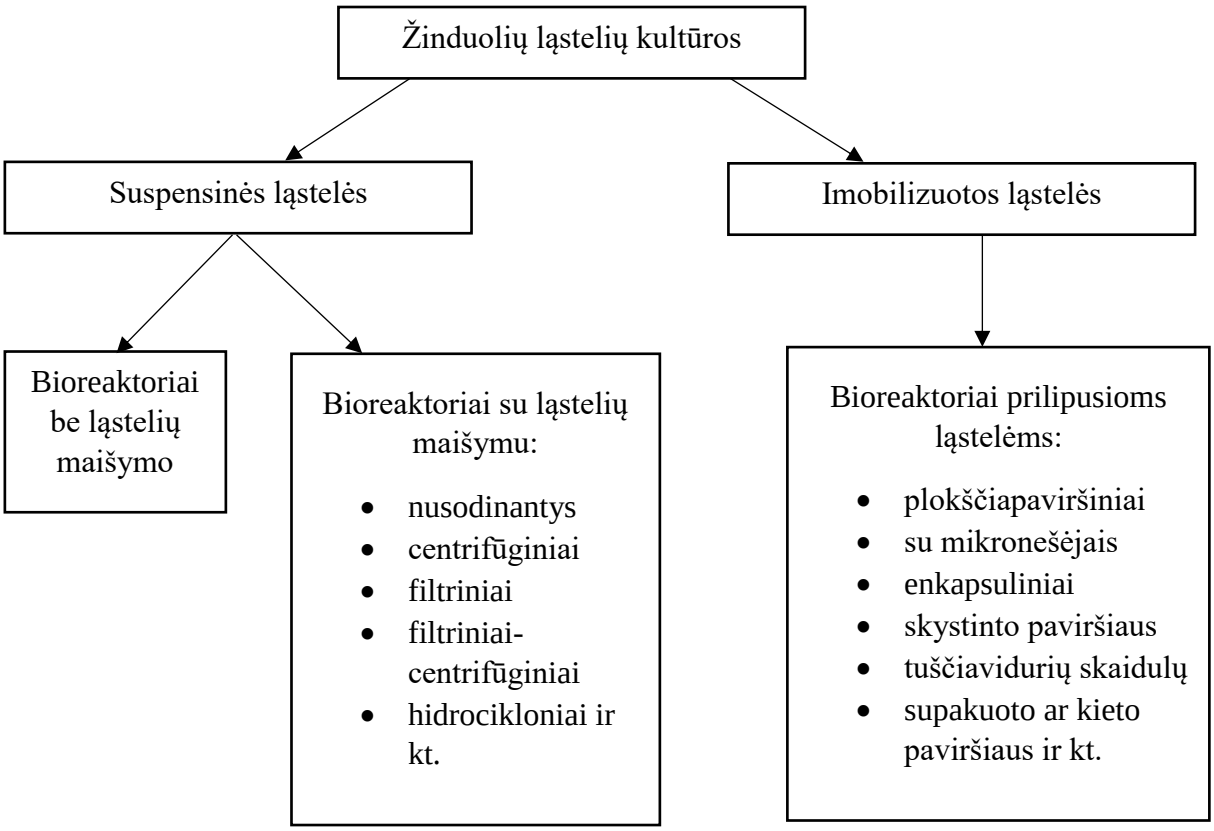
Raudonoji biotechnologija, t.y. žmogaus baltymų išgavimas iš žinduolių ląstelių buvo pradėta tyrinėti 1970-1980 metais. Tuo metu buvo išrastas ir interferonas, buvo pradėtas monokloninių antikūnų kūrimas, kurie šiuolaikinėje medicinoje naudojami piktybinių navikų ir kitų ligų gydymui. 1986 metais FDA aprobavo pirmąjį monokloninį antikūną muromonabą - CD3 (OKT3) [18]. Vėliau rekombinantiniai baltymai buvo pradėti išgauti iš kiniško žiurkėno kiaušidžių (CHO) ląstelių įskaitant tPA ir eritropoetiną [20]. Tokio tipo baltymų išgavimas yra ypač svarbus, nes ląstelės gamina glikozilintus baltymus, o jie yra identiški žmonių baltymams bei jų dirbtinė gamyba *in vitro* yra labai sunkiai įgyvendinama [18].

Žinduolių ląstelių kultivavimas reikalauja nemažai laiko ir pastangų. Šios ląstelės gali augti tiek skystoje aplinkoje, tiek prilipusios prie kieto paviršiaus, kitaip tariant, kultūros gali būti suspensinės ir imobilizuotos. Plačiau yra naudojamos suspensinės organizmų kultūros, kadangi produktas yra pašalinamas kartu su ištekančiu skysčiu ir jų kultivavimui nėra reikalingas kietas paviršius. Imobilizavimas reiškia ląstelių arba kitų organizmų prilipimą prie įvairių paviršių. Dėl šių paviršių didelės įvairovės galima atlikti biokatalizę (reakcijos ant paviršių) su medžiagomis tokiomis kaip žinduolių ir augalų ląstelės ar fermentai. Pagrindinis įmobilizuotų ląstelių privalumas prieš suspensines kultūras yra tai, kad jas galima auginti ilgai ir jos nepasišalina kartu su skysčiais [19][20].

3.1.3. Bioreaktoriai

Šiuolaikinėje medicinoje labai išpopuliarėjo gydymas antikūnais panaudojant žinduolių ląsteles, todėl toks gydymas reikalauja labai didelių ląstelių kiekių. Todėl antikūnų panaudojimo

poreikiai iškėlė biotechnologinį uždavinį: kaip praplėsti ir pagreitinti kultūrų kultivavimo galimybes, ypač stengiantis išvengti papildomų investicijų. Tam tikslui buvo pradėti kurti bioreaktoriai, paspartinantys ląstelių ir audinių gamybą *in vitro*. 3.1 paveikslėlyje pavaizduota dažniausiai taikomų bioreaktorių tipai.



3.1 pav. Bioreaktoriai, kurie naudojami žinduolių ląstelių kultivavimui [18].

Pagrindinis šių bioreaktorių tikslas yra palaikyti biologiškai aktyvią aplinką, kuri turi būti palanki daugintis ląstelėms. Bioreaktorių sudaro indas, kuriame procesas gali būti aerobinis ir anaerobinis.

Sukurti gerą bioreaktorių yra gana sudėtingas procesas. Bioreaktoriuje sukurtos sąlygos veikia jame esančius organizmus, todėl tikslus jų sukūrimas ir palaikymas yra vienas iš svarbiausių uždavinių, kad užtikrinti tinkamą bioreaktoriaus veikimą. Todėl jiems keliama nemažai reikalavimų, tokių kaip:

- Produktų gamybos ekonomiškumas;
- Švelnus maišymas ir ventiliavimas nepažeidžiant ląstelių;
- Žemas toksinių metabolitų lygis;
- Atitinkamų sąlygų: temperatūros, pH, deguonies ir ištirpusio CO^2 palaikymas;
- Didelė ląstelių ir jų gaminamų produktų koncentracija;
- Tinkamos terpės ir jos priedų naudojimas;

- Susidariusio produkto stabilumas;
- Įvairių procesų automatizavimas [18][21].

3.2.Žinduolių ląstelių matematinis auginimo modelis

Matematinis modelis kuriamas kiniško žiurkėno kiaušidžių (angl. *CHO*) ląstelėms, kurios yra pagrindinės išgaunant rekombinantinius baltymus. Modelyje bus palaikomos atitinkamos gliukozės ir glutamino koncentracijos. Apskirtai modelyje numatomi tokie kintamieji kaip biomasė, gliukozė, glutaminas, laktatas ir amonis. Taigi, modelio būsenos vektorius c yra:

$$c = [X_t, X_V, G_{lc}, G_{ln}, L_{ac}, NH_4^-], \quad (3.1)$$

kur: X_t – visas ląstelės tankis; X_V – gyvybingos ląstelės tankis; G_{lc} – gliukozės koncentracija, mM; G_{ln} – glutamino koncentracija, mM; L_{ac} – laktato koncentracija, mM; NH_4 – amonio koncentracija, mM.

Masių balanso lygtis gyvybingų ląstelių kiekiui yra:

$$\frac{dX_V}{dt} = \mu \cdot X_V - \frac{F}{W} \cdot X_V, \quad (3.2)$$

kur: μ – santykinis biomasės augimo greitis, h^{-1}; $F = F_{Gln} + F_{Gln} + F_{base}$.

Bendras ląstelių kiekis:

$$\frac{dX_t}{dt} = \mu_{net} \cdot X_V - \frac{F}{W} \cdot X_t, \quad (3.3)$$

kur: μ_{net} – santykinis gryno kiekio augimo greitis, h^{-1}.

Gliukozės koncentracija terpėje:

$$\frac{dGlc}{dt} = -q_{Glc} \cdot X_V - \frac{(F_{Gln} + F_{base})}{W} + \frac{F_{Glc}}{W} \cdot (c_{F_{Glc}} - Glc), \quad (3.4)$$

kur: q_{Glc} – santykinis gliukozės sunaudojimo greitis, mmol 10^9 ląstelių$^{-1}$ h^{-1}; $c_{F_{Glc}}$ – įtekančiame sraute esančios gliukozės koncentracija, mM.

Glutamino koncentracija terpėje:

$$\frac{dGln}{dt} = -q_{Gln} \cdot X_V - k_{deg} \cdot Gln - \frac{(F_{Glc} + F_{base})}{W} + \frac{F_{Gln}}{W} \cdot (c_{F_{Gln}} - Gln), \quad (3.5)$$

kur: q_{Gln} – specifinis glutamino sunaudojimo greitis, mmol 10^9 ląstelių$^{-1}$ h^{-1}; $c_{F_{Gln}}$ – įtekančiame sraute esančios gliukozės koncentracija, mM; k_{deg}– glutamino dehidratacijos konstanta, h^{-1}.

Laktato koncentracija terpėje:

$$\frac{dLac}{dt} = q_{Lac} \cdot X_V - \frac{F}{W} \cdot Lac, \quad (3.6)$$

kur: q_{Lac} – specifinis laktato sunaudojimo greitis, mmol 10^9 ląstelių$^{-1}$ h^{-1}.

Amonio sulfato koncentracija terpėje:

$$\frac{dNH_4}{dt} = q_{NH_4} \cdot X_V + k_{deg} \cdot Gln - \frac{F}{W} \cdot NH_4, \quad (3.7)$$

kur: q_{NH_4} – specifinis amonio sunaudojimo greitis, mmol 10^9 ląstelių$^{-1}$ h^{-1}.

Terpės tūrio kiekis:

$$\frac{dW}{dt} = F = F_{Glc} + F_{Gln} + F_{Base}, \quad (3.8)$$

kur: F – bendras įtekantis srautas, kg h^{-1}; W – kultūros svoris, kg; F_{Glc} – įtekantis gliukozės kiekis, kg h^{-1}; F_{Gln} – įtekantis glutamino kiekis, kg h^{-1}; F_{Base} – pagrindo suvartojimo greitis, kg h^{-1}.

Specifinius rodiklius esančius (3.2-3.8) lygtyse galime rasti iš šių lygybių:

$$\mu = \left(\frac{q_{Glc}}{Y_{GlcX} \cdot \left(\frac{G_{lc}}{k_1 + G_{lc}} \right)} + \frac{q_{Gln} - m_{Gln}}{Y_{GlnX}} \right) \cdot \left(1 - e^{-\frac{t}{t_{lag}}} \right), \quad (3.9)$$

kur: Y_{GlcX}– gliukozės sunaudojimas naudingumas (išeiga), mmol 10^9 ląstelių$^{-1}$ (parodo kiek gliukozės reikia, kad pagaminti 1 000 000 000 žinduolių ląstelių); k_1 – pirma konstanta gliukozei, mM; m_{Gln} – pastovaus glutamino palaikymo konstanta, mmol 10^9 ląstelių$^{-1}$ h^{-1}; Y_{GlnX} – glutamino sunaudojimo naudingumas, mmol 10^9 ląstelių$^{-1}$.

$$\mu_{net} = \mu - k_d, \quad (3.10)$$

kur: k_d – ląstelės irimo greičio koeficientas, h^{-1};

$$k_d = k_{dmax} \cdot \left(\frac{L_{ac}}{K_{Lac} + L_{ac}} + \frac{NH_4^+}{K_{NH_4^+} + NH_4^+} \right), \quad (3.11)$$

kur: k_{dmax} – maksimalus ląstelių irimo greitis, h^{-1}; K_{Lac} – laktato slopinimo koeficientas, mM; $K_{NH_4^+}$ – amonio slopinimo koeficientas, mM;

$$q_{Glc} = q_{Glcmax} \cdot \frac{G_{lc}}{G_{lc} + K_{Glc}}, \quad (3.12)$$

kur: q_{Glcmax} – maksimalus gliukozės sunaudojimo greitis, mmol 10^9 ląstelių$^{-1}$ h^{-1}; K_{Glc} – gliukozės slopinimo koeficientas, mM;

$$q_{Gln} = q_{Glnmax} \cdot \frac{G_{ln}}{G_{ln} + K_{Gln}} + m_{Gln}, \quad (3.13)$$

kur: q_{Glnmax} – maksimalus glutamino sunaudojimo koeficientas, mmol 10^9 ląstelių$^{-1}$ h^{-1}; K_{Gln} – glutamino slopinimo koeficientas, mM;

$$q_{Lac} = Y_{LacGlc} \cdot \frac{G_{lc}}{G_{lc} + k_2} \cdot q_{Glc}, \quad (3.14)$$

kur: Y_{LacGlc} – laktato išeiga gaminant gliukozę, mmol mmol^{-1}; k_2 – antra konstanta gliukozei, mM;

$$q_{NH_4^+} = Y_{NH_4^+X} \cdot \mu + m_{AA}, \quad (3.15)$$

kur: $Y_{NH_4^+X}$ - amonio sunaudojimas išeiga, mmol 10^9 ląstelių$^{-1}$, m_{AA}– specifinis amonio gamybos greitis, kai yra maža glutamino koncentracija, mmol 10^9 ląstelių$^{-1}$ h^{-1}.

Deguonies sunaudojimo santykis yra svarbiausias kintamasis, nes jo reikia norint apskaičiuoti valdymo proceso kintamuosius. Manoma, kad deguonies suvartojimo norma yra nuolatinėje pusiausvyroje su biomasės koncentracija, todėl šis ryšis gali būti išreiškiamas panaudojant Luedeking-Piret sąryšį:

$$OUR = q_{O_2} \cdot X_V + m_O \cdot X_V, \tag{3.16}$$

kur: kintamasis q_{O_2}lygus:

$$q_{O2} = Y_{OGln} \cdot q_{Gln} + Y_{OGlc} \cdot q_{Glc} \cdot \frac{K_O}{G_{lc} + K_O}, \tag{3.17}$$

kur: m_o – pastovaus deguonies palaikymo konstanta, mmol 10^9 ląstelių$^{-1}$ h^{-1}; Y_{OGln}– deguonies išeiga gaminant glutaminą, mmol mmol^{-1}; Y_{OGlc} – deguonies išeiga gaminant gliukozę, mmol mmol^{-1}; K_o – deguonies slopinimo konstanta, mM.

Svarbu paminėti, tai kad žinduolių ląstelių kultūroje, kiekybiniai santykiai tarp reaguojančiu medžiagų keičiasi bėgant laikui. Tai reikalauja keisti modelio parametrus kultivavimo metu [22].

4. BIOTECHNOLOGINIŲ PROCESŲ MODELIŲ REALIZAVIMAS

4.1. Mielių auginimo modelio realizavimas

4.1.1. Matematinis mielų auginimo modelis Simulink aplinkoje

Mielių auginimo modelis realizuojamas *MATLAB* programinėje įrangoje, *Simulink* aplinkoje. Visų pirma DEE bloke (4.1 pav.) surašomos modelio masės balanso lygtys, kurios pateiktos 1.3.2 skyriuje.

4.1 pav. *DEE* bloke aprašytos mielių modelio masių balanso lygtys

MATLAB Function bloke apsirašomos santykinės augimo bei tiekimo greičio išraiškos. Augimo greičio išraiškose priimamos tokios kintamųjų vertės:

$\mu_{\max\_so}$ – 0,18 [g/l];

K_s^o – 0,1;

K_{O_2} – 0,0005.

$\mu_{\max\_sf}$ – 0,05 [g/l];

K_s^f – 10;

$\mu_{\max\_eo}$ – 0,025 [g/l];

K_e^o – 1;

Kaip atrodo augimo greičiai *Simulink* aplinkoje pateikta 1 priede.

Deguonies ir anglies dioksido tiekimo greičio išraiškose priimami tokie dydžiai:

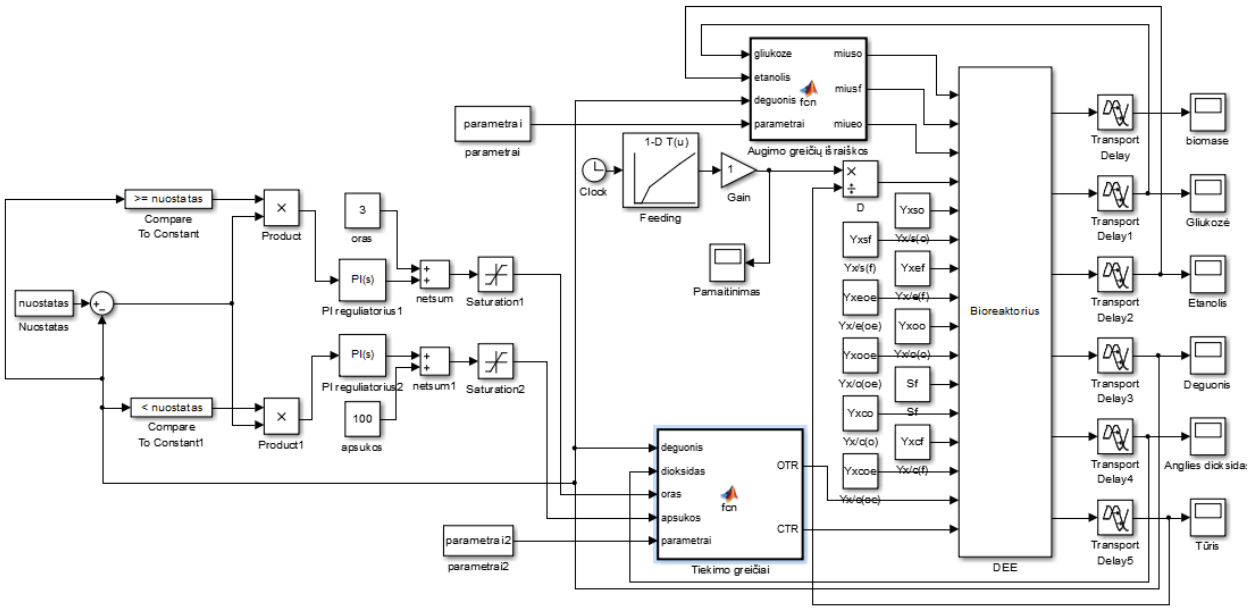
k_La – 400 [1/h];

O^* - 0,006 [g/l];

C^* - 0,0001 [g/l].

Tiekimo greičio išraiškų programos kodas pateiktas 2 priede.

Modelis sukurtas naudojant duomenų lentelę. Joje pats vartotojas gali įsivesti norimas pamaitinimo srauto reikšmes. Sudarytas matematinis mielių augimo modelis pateiktas 4.2 paveikslėlyje.

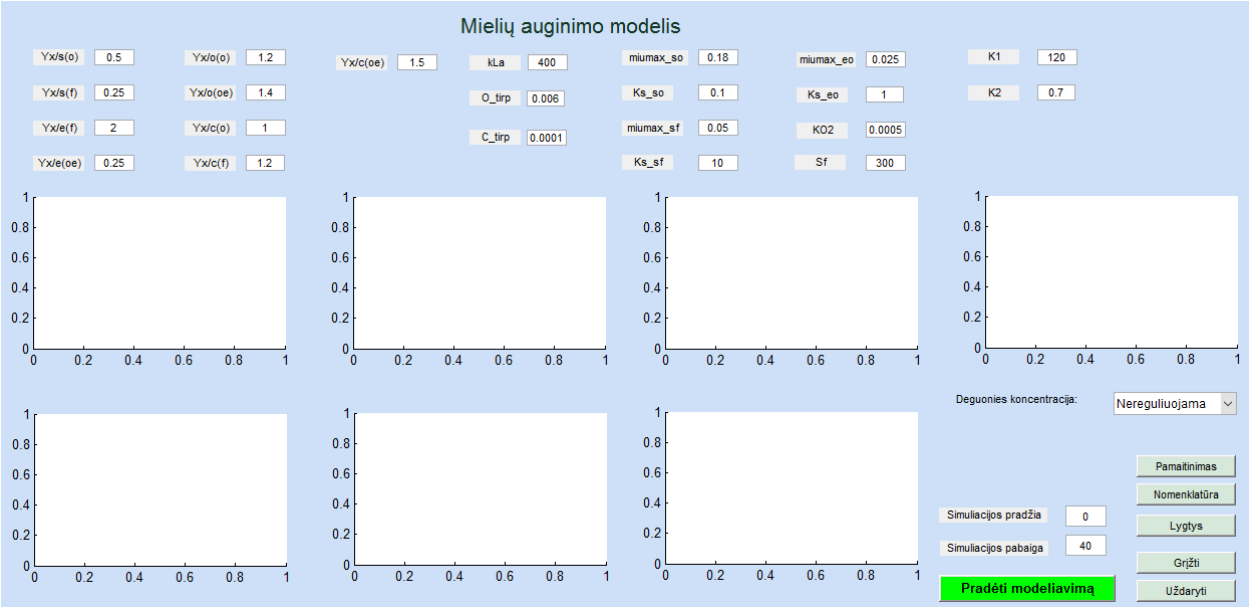


4.2 pav. Mielių augimo matematinis modelis realizuotas *MATLAB/Simulink*

4.1.2. Vartotojo sąsaja mielių auginimo modeliui

Mielių auginimo modeliui (žiūrėti į 4.1.1 skyrių), buvo sukurta grafinė sąsaja. Šios sąsajos pagalba vartotojas lengvai ir suprantamai gali keisti modelio parametrus, stebėti gautus rezultatus. Sąsaja buvo kuriama naudojant *Matlab*/*Guide* įrankį. Kaip matyti iš 4.3 pav. yra grafiškai atvaizduojami visi proceso kintamieji su pamaitinimo srautu, kurį kaip minėta pagal poreikius užsiduoda pats vartotojas.

Programos kodas realizuoti grafinei sąsajai pateiktas 3 priede.



4.3 pav. Mielių modelio grafinė vartojo sąsaja

Paspaudus mygtuką „Lygtys“ atidaromas paveikslėlis, kuriame pavaizduotos masės balanso lygtys kurios naudotos šiam mielių matematiniui modeliui:

Biomasės koncentracija:

$$\frac{dX}{dt} = (\mu_s^o + \mu_s^f + \mu_e^o - D)X.$$

Gliukozės (cukraus) koncentracija:

$$\frac{dS}{dt} = \left(-\frac{\mu_s^o}{Y_{X/S}^o} - \frac{\mu_s^f}{Y_{X/S}^f}\right)X + (S_f - S)D.$$

Etanolio koncentracija:

$$\frac{dE}{dt} = \left(\frac{\mu_s^f}{Y_{X/E}^f} - \frac{\mu_e^o}{Y_{X/E}^{oe}}\right)X - DE.$$

Ištirpusio deguonies masės koncentracija:

$$\frac{dO}{dt} = \left(-\frac{\mu_s^o}{Y_{X/O}^o} - \frac{\mu_e^o}{Y_{X/E}^{oe}}\right)X - DO + OTR.$$

Ištirpusio anglies dioksido koncentracija:

$$\frac{dC}{dt} = \left(\frac{\mu_s^o}{Y_{X/C}^o} + \frac{\mu_s^f}{Y_{X/C}^f} + \frac{\mu_e^f}{Y_{X/C}^{oe}}\right)X - DC + CTR.$$

4.4 pav. Masės balanso lygtys mielių matematiniai modeliui

Paspaudus mygtuką „Nomenklatūra“ atidaromas paveikslėlis, kuriame paaiškinti visi grafinėje sąsajoje naudoti kintamieji su matavimo vienetais:

Yx/s(o)	biomasės/gliukozės išeigos koeficientas aerobinėse sąlygose	g/g
Yx/s(f)	biomasės/gliukozės išeigos koeficientas anaerobinėse sąlygose	g/g
Yx/e(f)	biomasės/etanolio išeigos koeficientas anaerobinėse sąlygose	g/g
Yx/e(oe)	biomasės/etanolio išeigos koeficientas aerobinėse sąlygose kai dalis deguonies yra sunaudojama etanolio	g/g
Yx/o(o)	biomasės/deguonies išeigos koeficientas aerobinėse sąlygose	g/g
Yx/o(oe)	biomasės/deguonies išeigos koeficientas aerobinėse sąlygose kai dalis deguonies yra sunaudojama etanolio	g/g
Yx/c(o)	biomasės/anglies dioksido išeigos koeficientas aerobinėse sąlygose	g/g
Yx/c(f)	biomasės/anglies dioksido išeigos koeficientas anaerobinėse sąlygose	g/g
Yx/c(oe)	biomasės/anglies dioksido išeigos koeficientas aerobinėse sąlygose kai dalis deguonies yra sunaudojama etanolio	g/g
kLa	masės perdavimo iš dujinės į skystąją fazę koeficientas	1/h
O_tirp	maksimalus deguonies tirpumo koeficientas skystoje kultivavimo terpėje	g/l
C_tirp	maksimalus anglies dioksido tirpumo koeficientas skystoje kultivavimo terpėje	g/l
miumax_so	maksimali biomasės augimo greičio išraiška aerobinėse sąlygose	g/l
Ks_so	mono koeficientas	-
miumax_sf	maksimali biomasės augimo greičio išraiška anaerobinėse sąlygose	g/l
Ks_sf	mono koeficientas	-
miumax_eo	maksimali biomasės augimo greičio išraiška aerobinėse sąlygose, kai dalis deguonies yra sunaudojama etanolio	g/l
Ks_eo	mono koeficientas	-
KO2	limitavimas deguonimi	-
Sf	substrato koncentracija pamaitinimo tirpale	g/l
K1	Konstanta	-
K2	Konstanta	-

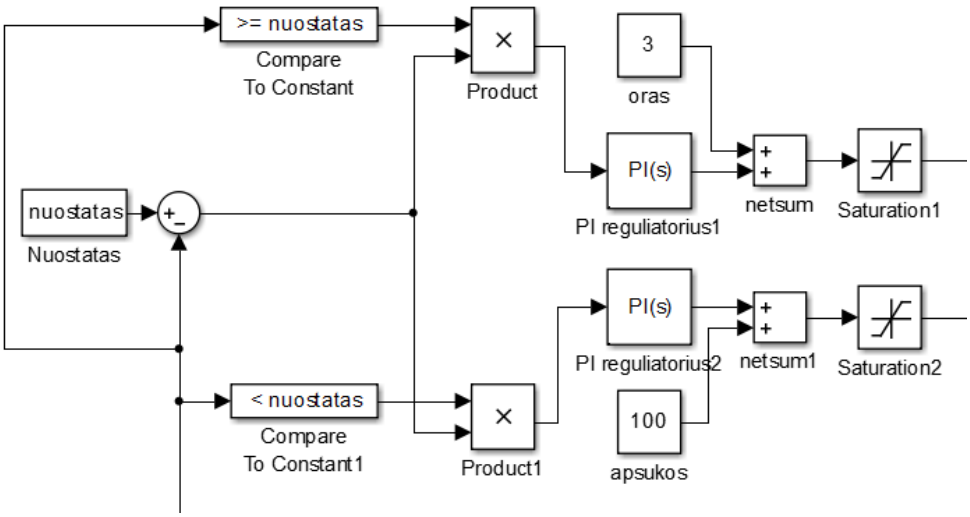
4.5 pav. Mielių sąsajos nomenklatūra

4.1.3. Ištirpusio deguonies koncentracijos valdymas

Mielių auginimo modelis buvo realizuotas dviem variantais. Pirmas iš jų, kai k_La kintamasis turi pastovią reikšmę, kitas kai ji kinta nuo esamos deguonies koncentracijos. Taigi, atliktas ištirpusios deguonies reguliavimas keičiant k_La dydį. Masės perdavimo iš dujinės į skystąją fazę koeficientas buvo keičiamas pagal tokią supaprastintą priklausomybę:

$$k_L\mathrm{a} = K_1 \cdot \frac{oro\ srautas}{t\bar{u}ris} + K_2 \cdot maišyklės\ apsukos\ [24]. \quad (4.1)$$

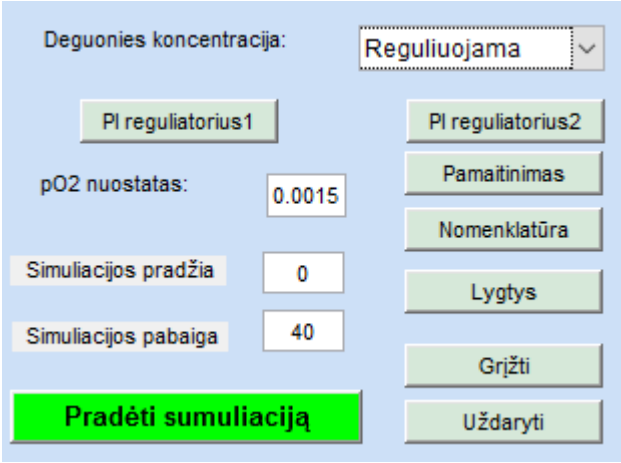
Jeigu esama ištirpusio deguonies vertė lygi arba yra didesnė už užduotą nuostatą, tai keičiamas oro srautas, o jei mažesnė, reguliuojamos maišyklės apsukos. Tokiam valdymui buvo naudojami du reguliatoriai, kurie pavaizduoti *Simulink* aplinkoje 4.1 paveikslėlyje.



4.6 pav. Ištrupusio deguonies koncentracijos valdymas

Pradinė paduodama oro srauto reikšmė priimta 3 l/min, o apsukos 100 per minutę.

Grafinėje mielių sąsajoje (4.7 pav.) vartotojui pasirinkus deguonies koncentracijos valdymą, galima keisti abiejų reguliatorių parametrus bei pasirinkti nuostatą. Atlikus norimus pakeitimus galime stebėti kintamųjų kitimą grafine forma.

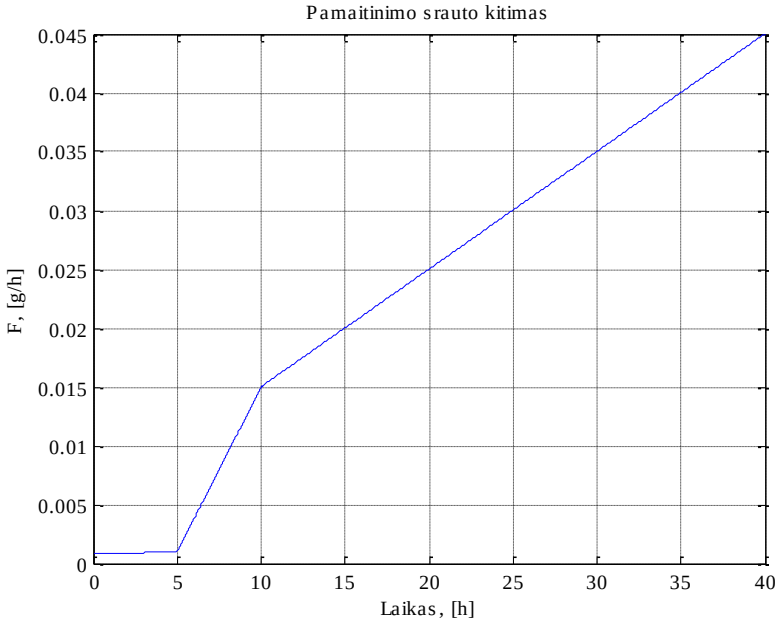


4.7 pav. Deguonies koncentracijos valdymas

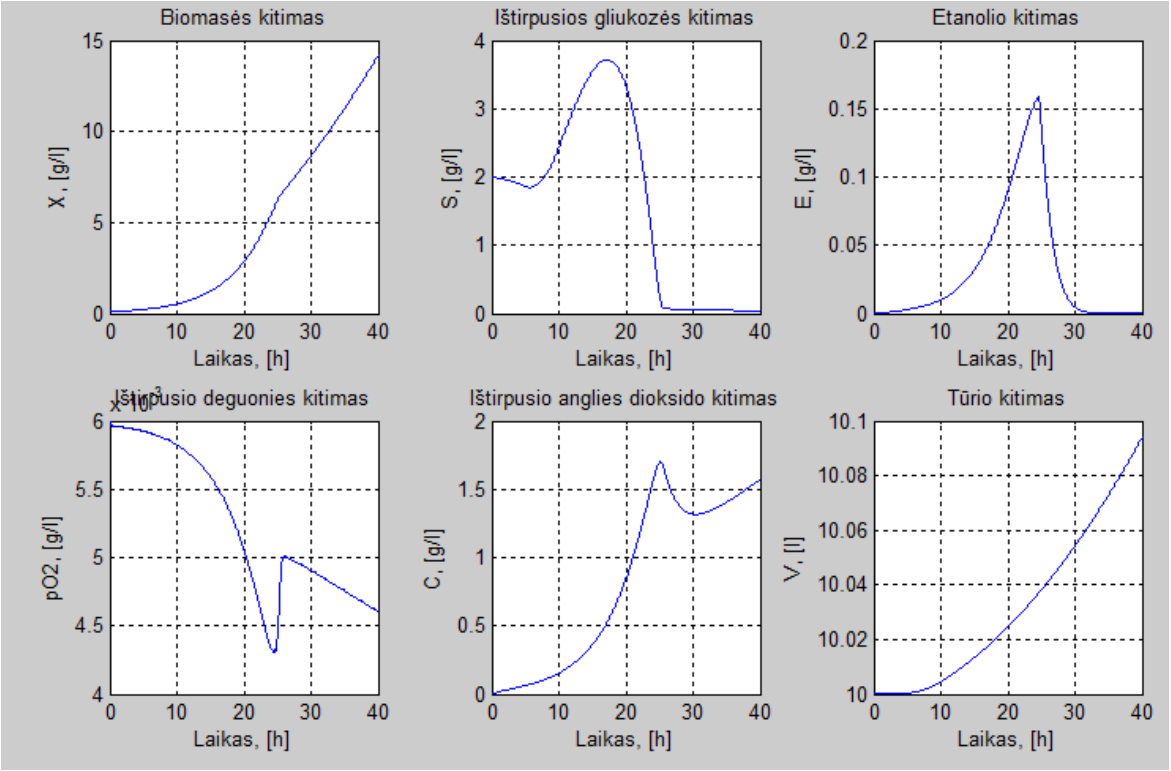
4.1.4. Mielių auginimo modelio modeliavimo rezultatai

Rezultatuose pateikiama biomasės, ištirpusios gliukozės, etanolio, ištirpusio deguonies, anglies dioksido bei tūrio kitimas bėgant laikui (40 valandų laikotarpiu).

Visų pirma rezultatai pateikiami be reguliavimo, kai k_La yra pastovus ir lygus 400, 1/h. Bioreaktoriaus pamaitinimo srautas užsiduodamas laisvai 4.8 pav., o modeliavimo rezultatai pateikti 4.9 pav.

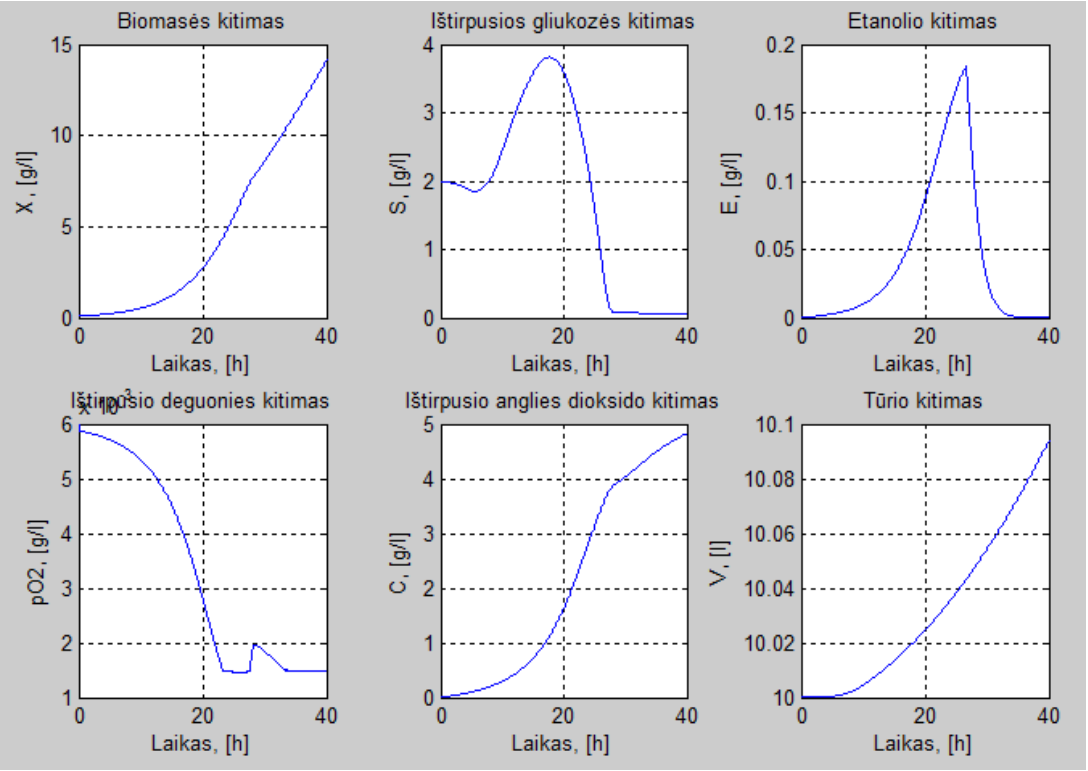


4.8 pav. Pamaitinimo srautas



4.9 pav. Mielių auginimo modelio modeliavimo rezultatai

Toliau rezultatai pateikiami jau su ištirpusio deguonies koncentracijos valdymu. Pamaitinimo srautas paliekamas toks pat ir įvedamas ištirpusio deguonies koncentracijos nuostatas - 0,0015 g.

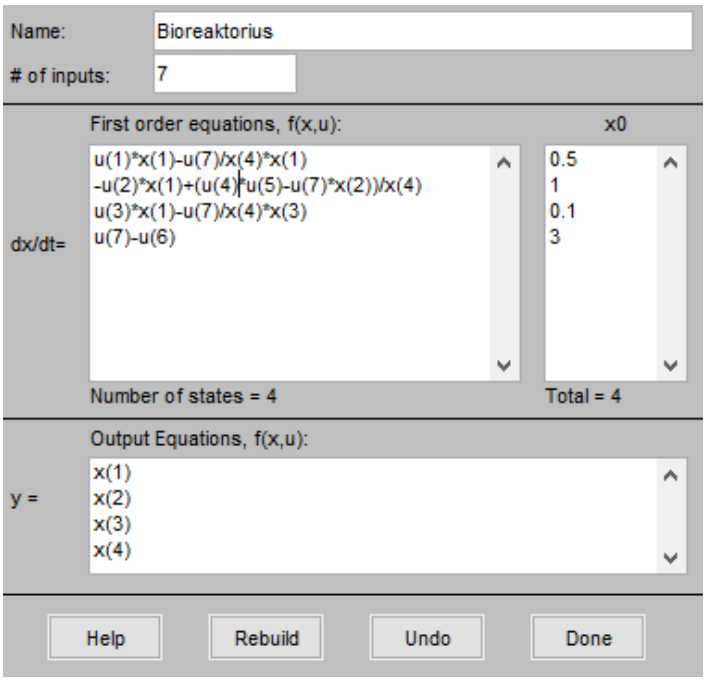


4.10 pav. Ištirpusios deguonies koncentracijos valdymas

4.2. E. coli bakterijų auginimo modelio realizavimas

4.2.1. Matematinis E. coli bakterijų auginimo modelis Simulink aplinkoje

E. coli bakterijų auginimo modelio masių balanso lygtys (2.9 - 2.12) surašomos imitaciniame bioreaktoriuje, kuris kuriamas naudojant *Matlab/Simulink* aplinką.



4.11 pav. *DEE* bloke aprašytos E. coli modelio masių balanso lygtys

Reakcijos greičio bei į bioreaktorių tiekiamų medžiagų srautas aprašomas naudojant *MATLAB Function* blokus. Reakcijos augimo greičio išraiškose, priimamos tokios kintamųjų vertės:

$\sigma_{\max}$ – 1,3 [1/h];

K_s – 0,01 [g/kg];

K_{si} – 100 [g/kg];

$K_{ai} - 10$ [g/kg];

$Y_{xa} - 0{,}5$ [g/kg];

$m - 0{,}0235$ [g/g/h];

$Y_{as} - 0{,}6$ [g/g];

$K_a - 1{,}0$ [g/kg].

Reakcijos greičių išraiškų programos kodas pateiktas 4 priede.

Į bioreaktorių įtekančio srauto apskaičiavimo kodas pateiktas 5 priede. Skaičiavimuose priimami tokie dydžiai:

Y_{bx} – 0,0009 [g/g];

Y_{ox} – 0,75 [g/g/h];

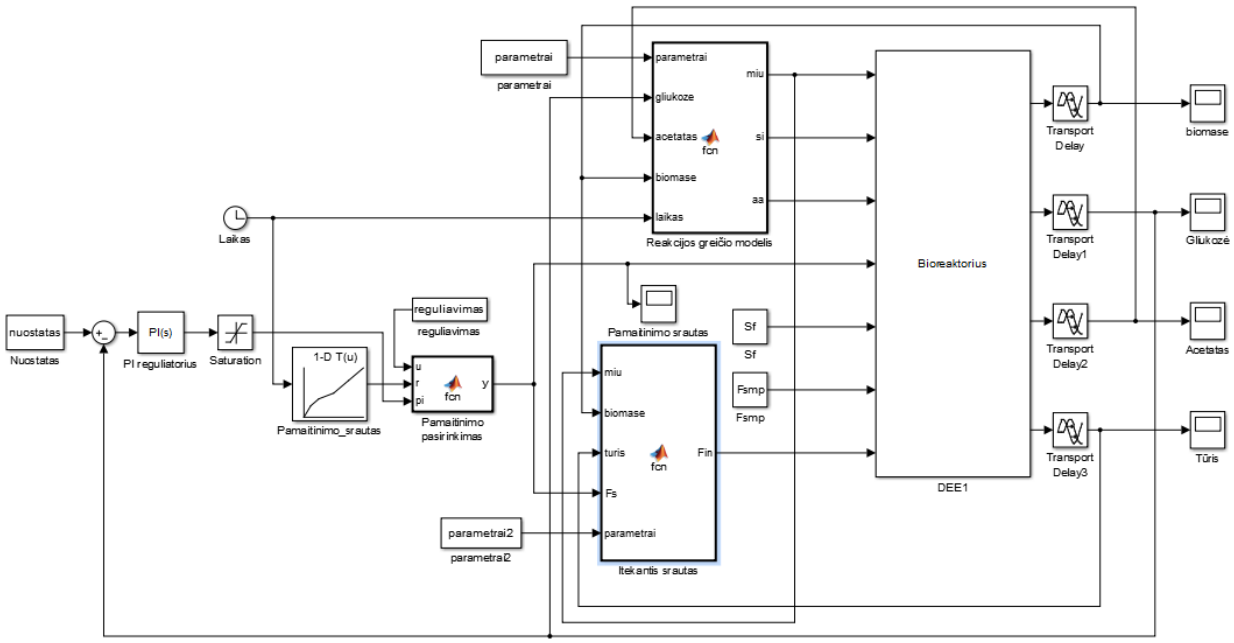
$\omega - 0{,}015$ [g/g/h];

$m - 0{,}0235$ [g/g/h];

Y_{O_2x} – -0,000375 [g/kg];

k_e – -0,001.

E. coli bakterijų pamaitinimo srautas taip pat sukurtas naudojant duomenų lentelę.

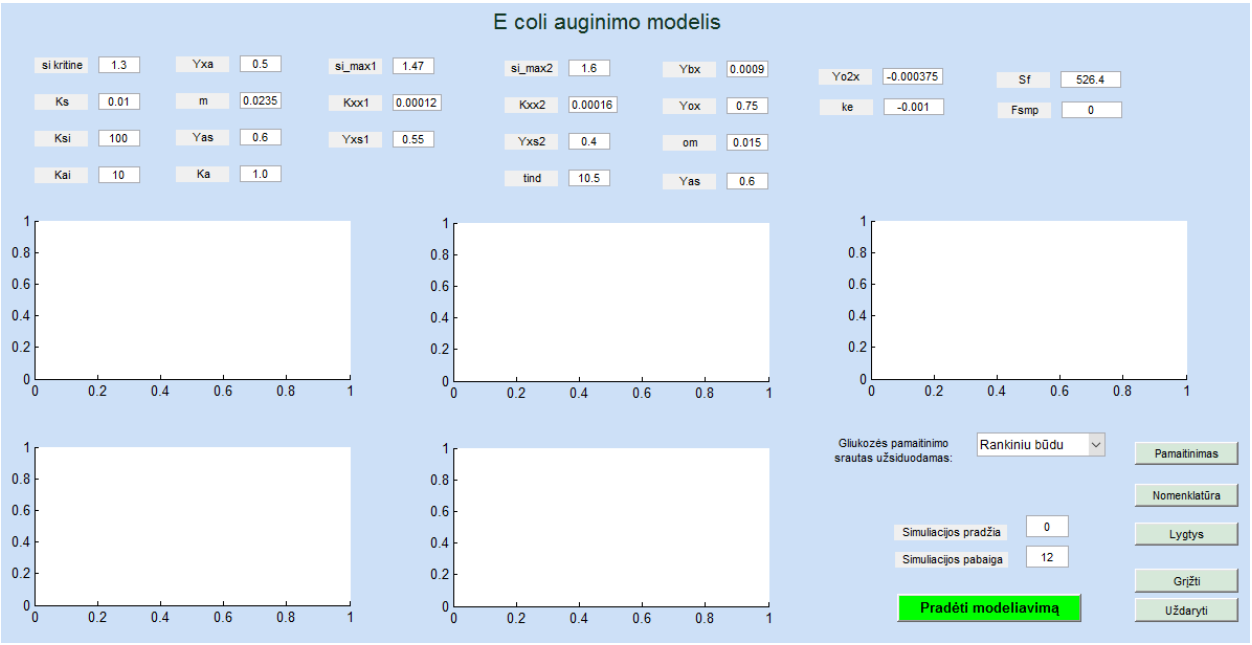


4.12 pav. E. coli bakterijų augimo matematinis modelis realizuotas *MATLAB/Simulink*

4.2.2. Vartotojo sąsaja E. coli bakterijų auginimo modeliui

E. coli bakterijų auginimo modeliui, kaip ir mielių, buvo sukurta grafinė sąsaja, kad vartotojas lengvai galėtų keisti parametrus, bei stebėti gautus rezultatus. Sąsaja sukurta 4.2.1 skyriuje pateiktam modeliui, naudojant *Matlab/GUI* įrankį.

Programos kodas sąsajai realizuoti pateiktas 6 priede.



4.13 pav. E. coli bakterijų auginimo modelio vartotojo sąsaja

Paspaudus mygtuką „Lygtys“ atidaromas paveikslėlis, kuriame pavaizduotos masės balanso lygtys kurios naudotos šiam matematiniam modeliui

Biomasės koncentracija:

$$\frac{dX}{dt} = \mu_s X - \left(\frac{F}{V}\right) X,$$

Gliukozės (substrato) koncentracija:

$$\frac{dS}{dt} = -\sigma X - \frac{F}{V} S + \frac{F_s}{V} S_f,$$

Acetato koncentracija:

$$\frac{dA}{dt} = \alpha X - \frac{F}{V} A,$$

Terpės tūris:

$$\frac{dV}{dt} = F - F_{smp}.$$

4.14 pav. Masės balanso lygtys E. coli bakterijų matematiniai modeliui

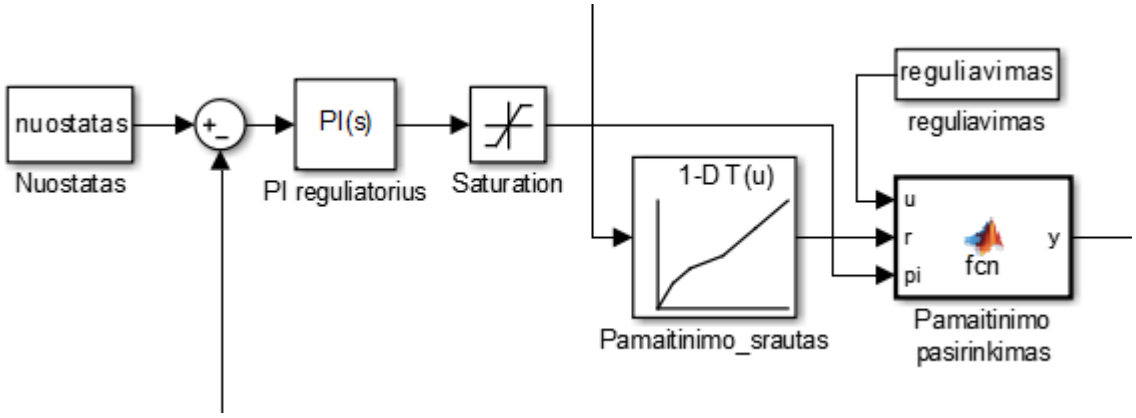
Paspaudus mygtuką „Nomenklatūra“ atidaromas paveikslėlis, kuriame paaiškinti visi grafinėje sąsajoje naudoti kintamieji su matavimo vienetais:

si kritine	kritinė substrato sunaudojimo greičio vertė	1/h
Ks	Mono koeficientas substratui	g/kg
Ksi	substrato inhibicijos konstanta	g/kg
Kai	acetato inhibicijos konstanta	g/kg
Yxa	biomasės/gliukozės išeiga	g/g
m	gyvybinių funkcijų palaikymo koeficientas	g/g/h
Yas	acetato/substrato išeiga	g/g
Ka	acetato inhibicijos konstanta	g/g
si_max1	maksimalus substrato sunaudojimo greitis	1/h
Kxx1	substrato sunaudojimo inhibicija biomasei	-
Yxs1	biomasės/gliukozės išeiga	g/g
si_max2	maksimalus substrato sunaudojimo greitis (po indukcijos)	g/1
Kxx2	substrato sunaudojimo inhibicija biomasei (po indukcijos)	-
Yxs2	biomasės/gliukozės išeiga (po indukcijos)	g/g
tind	indukcijos laikas	h
Ybx	bazės/biomasės išeiga	g/g
Yox	deguonies/biomasės augimo išeiga, g/g.	g/g
Yas	acetato/gliukozės išeiga	g/g
Yo2x	CO_2/biomasės išeiga	g/g
ke	garavimo greitis	-
Sf	substrato koncentracija pamaitinimo tirpale	g/kg
Fsmp	paimtų mėginių kiekis	1/h

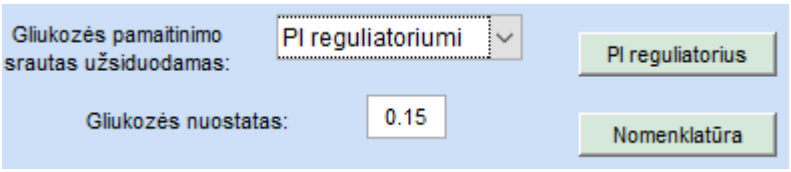
4.15 pav. E. coli sąsajos nomenklatūra

4.2.3. Ištirpusios gliukozės koncentracijos valdymas

E. coli bakterijų modelio pamaitinimo srautas realizuotas dviem atvejais: kai vartotojas pats užsiduoda dydžius vartotojo lentelėje ir kai atliekamas gliukozės valdymas, pagal užduotą nuostato signalą. Valdymui realizuoti į modelį įterpiamas PI reguliatorius (4.16 pav.), o vartotojo sąsajoje sutiekiama galimybė, pasirinkti pamaitinimo srauto tipą, bei keisti nuostato vertę su reguliatoriaus parametrais (4.17 pav.).



4.16 pav. Ištirpusios gliukozės valdymas su PI reguliatoriumi

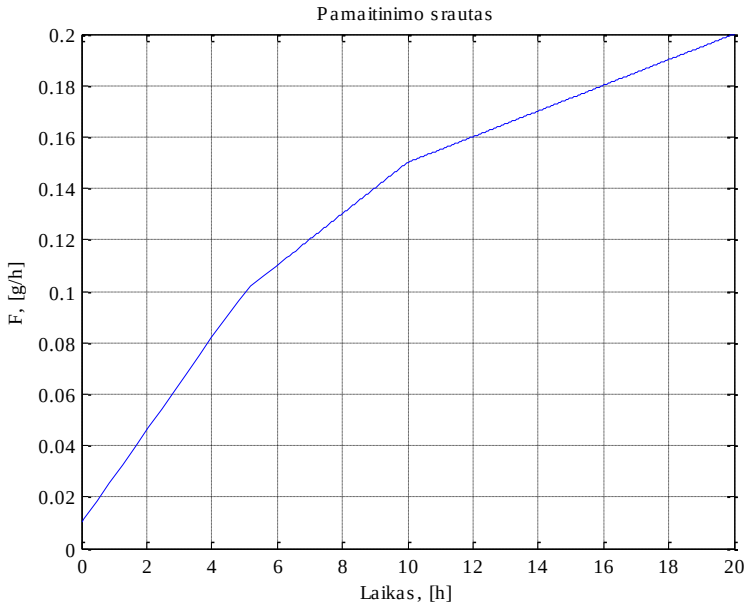


4.17 pav. Ištirpusios gliukozės koncentracijos valdymas

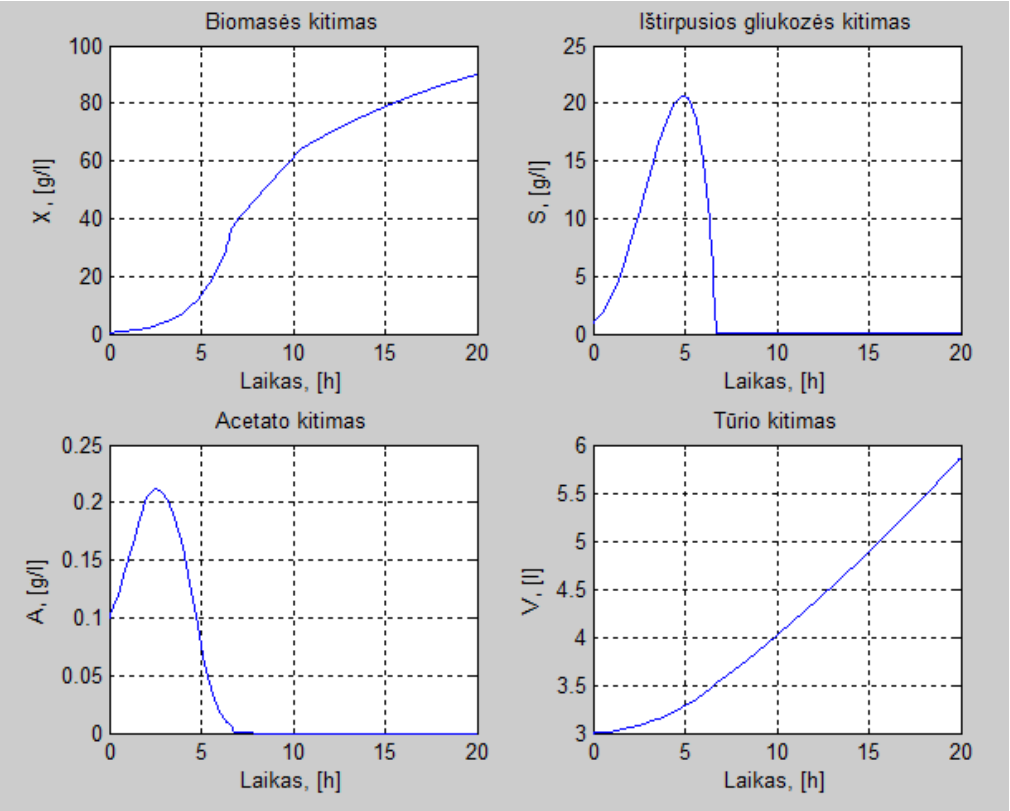
4.2.4. E. coli bakterijų modelio modeliavimo rezultatai

Rezultatuose pateikiami biomasės, gliukozės, acetato bei tūrio kitimas bėgant laikui (12 valandų laikotarpiu).

Pirmuoju atveju modeliavimo pamaitinimo srautas užduodamas be valdymo. Jis realizuojamas naudojant duomenų lentelę, kurioje laisvai galima pačiam įsivesti norimas reikšmes.

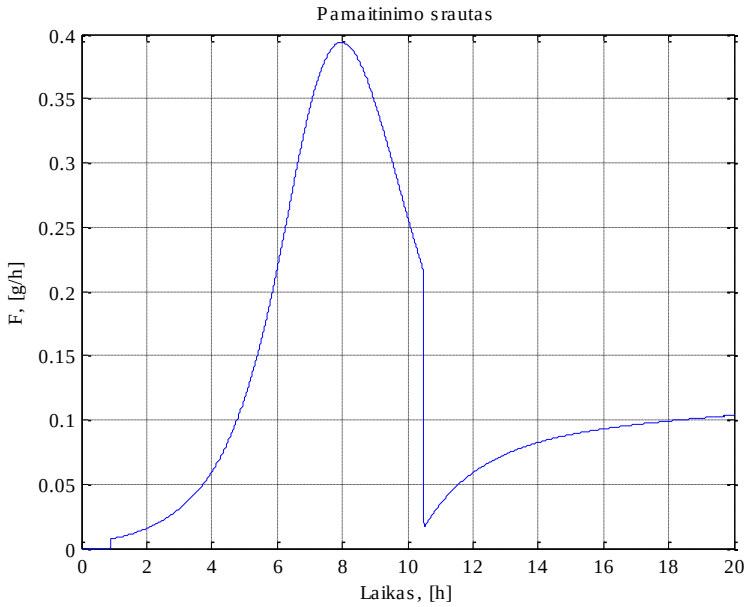


4.18 pav. Pamaitinimo srautas

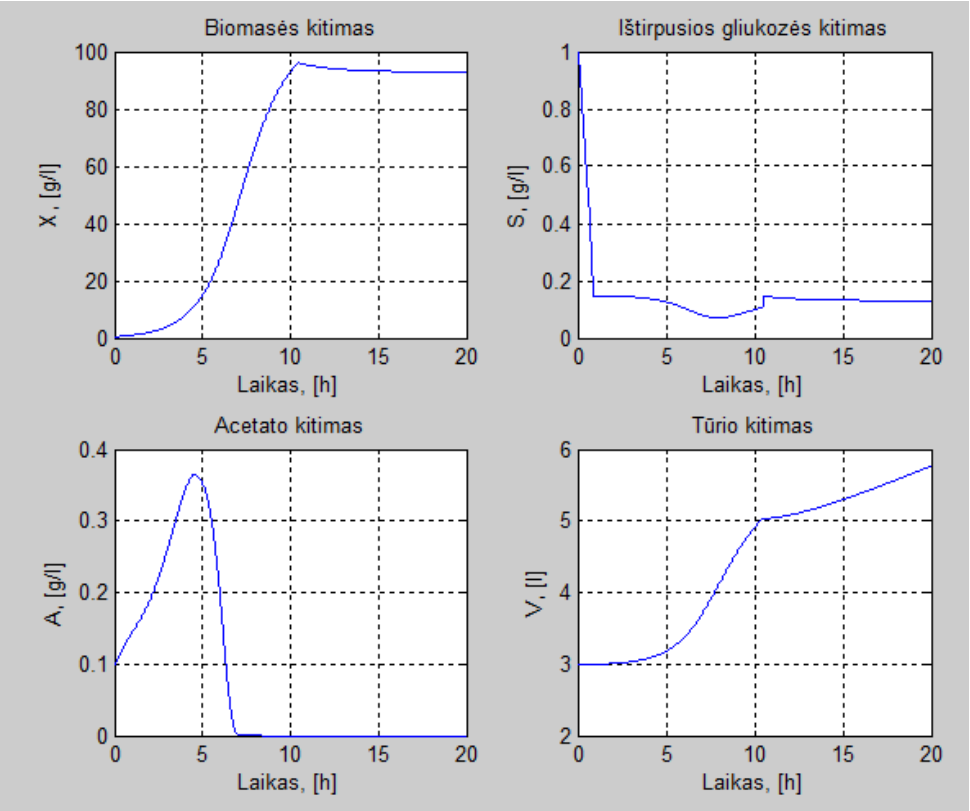


4.19 pav. E. coli bakterijų auginimo modelio modeliavimo rezultatai

Toliau rezultatai pateikiami jau matematiniame modelyje įterpiant reguliatorių ir valdant gliukozės koncentraciją terpėje. Užsiduodamas gliukozės nuostatas: 0.15. Atlikus modeliavimą gaunamas toks pamaitinimo srautas į bioreaktorių:



4.20 pav. Pamaitinimo srautas kai atliekamas gliukozės koncentracijos terpėje valdymas

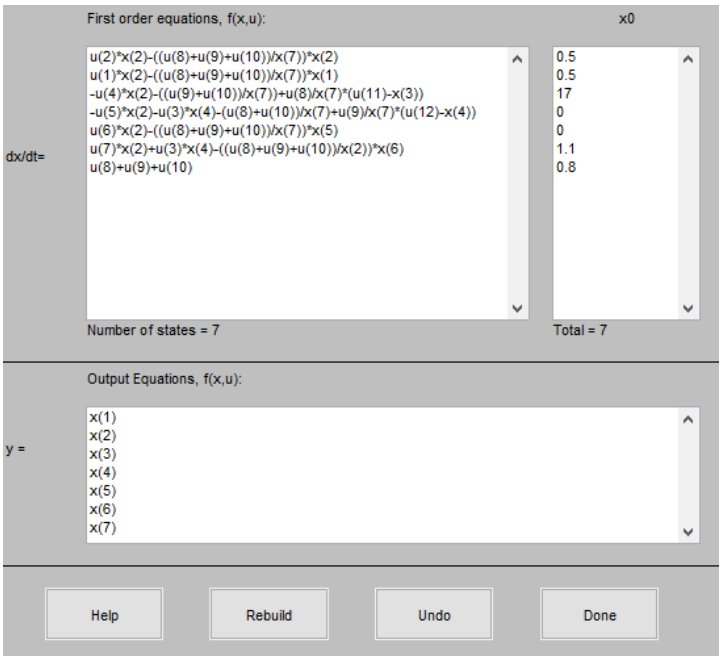


4.21 pav. E. coli bakterijų auginimo modelio modeliavimo rezultatai kai atliekamas gliukozės koncentracijos terpėje valdymas

4.3. Žinduolių ląstelių auginimo modelio realizavimas

4.3.1. Simulink aplinkoje sukurtas matematinis modelis

Žinduolių ląstelių auginimo modelis, kaip ir prieš tai du buvę, buvo kuriamas *Matlab/Simulink* aplinkoje. Kaip ir įprasta, *DEE* bloke surašomos masių balanso lygtys:



4.22 pav. Masių balanso lygtys žinduolių ląstelių auginimo modeliui

Proceso augimo greičių išraiškoms aprašyti, priimamos apačioje pateiktos kintamųjų vertės. Remiantis šiais dydžiais sukurtas programos kodas, aprašantis dydžių augimo greičio išraiškas. Šis kodas pateiktas 7 priede.

$c_{F_{Glc}}$ – 0,83 [mM];

$c_{F_{Gln}}$ – 0,20 [mM];

q_{Glcmax} – 0,18 [mmol 10^9 cells^{-1} h^{-1}];

q_{Glnmax} – 0,06 [mmol 10^9 cells^{-1} h^{-1}];

K_{Glc} – 2,25 [mM];

K_{Gln} – 0,23 [mM];

$K_{Lac} - 52{,}26$ [mM];

$K_{NH_4X} - 5{,}39$ [mM];

$k_{dmax} - 6{\times}10^{-3}$ [h^{-1}];

$Y_{GlcX} - 6{,}97$ [mmol 10^9 cells^{-1}];

$Y_{GlnX} - 0{,}97$ [mmol 10^9 cells^{-1}];

$Y_{LacGlc} - 1{,}59$ [mmol mmol^{-1}];

$Y_{NH_4X} - 0{,}48$ [mmol 10^9 cells^{-1}];

$Y_{OGlc} - 2{,}57$ [mmol mmol^{-1}];

$Y_{OGln} - 10{,}43$ [mmol mmol^{-1}];

k_1 – 0,32 [mM];

k_2 – 3,14 [mM];

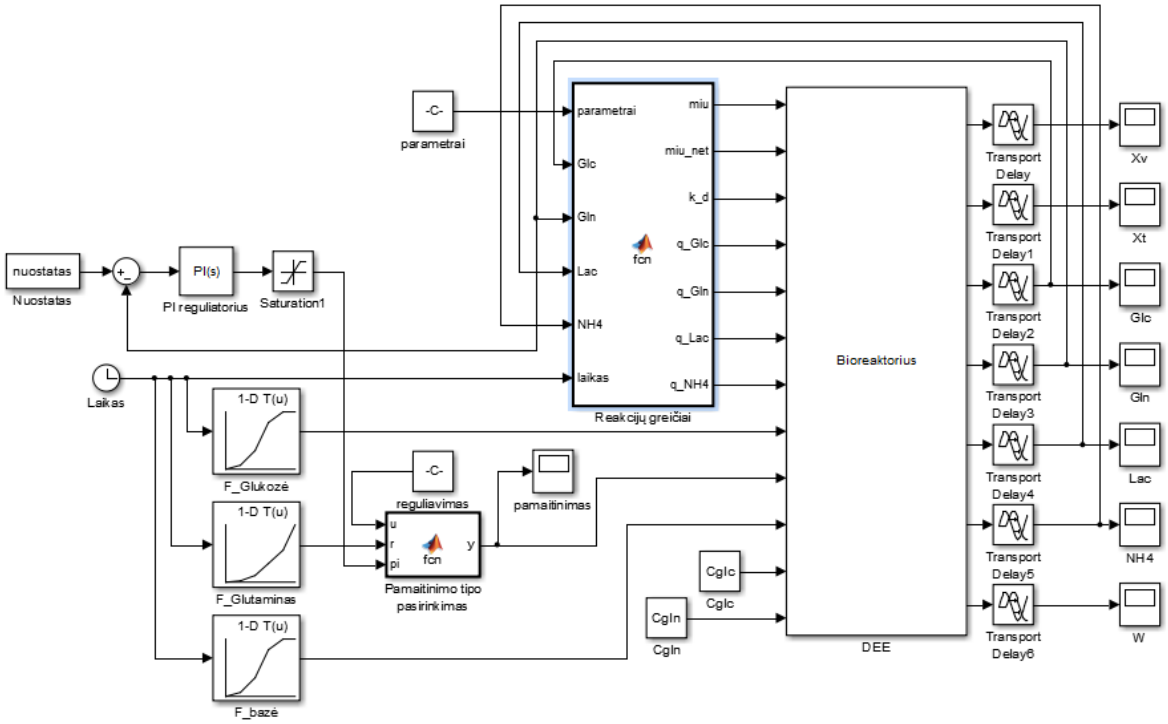
m_{Gln} – 5 × 10^{-3} [mmol 10^9 cells^{-1} h^{-1}];

m_{AA} – 2 × 10^{-4} [mmol 10^9 cells^{-1} h^{-1}];

K_O – 1,05 [mM];

m_O – 0,06 [mmol 10^9 cells^{-1} h^{-1}].

4.23 paveikslėlyje sukurtas žinduolių ląstelių matematinis auginimo modelis. Modelyje naudojami du pamaitinimo srautai: gliukozės, glutamino. Bazės (šarmo) srautas reikalingas terpės pH palaikyti. Visiems srautams imituoti sukurta vartotojo duomenų lentelė, kurioje vartotojas pats pagal poreikius gali juos keisti.

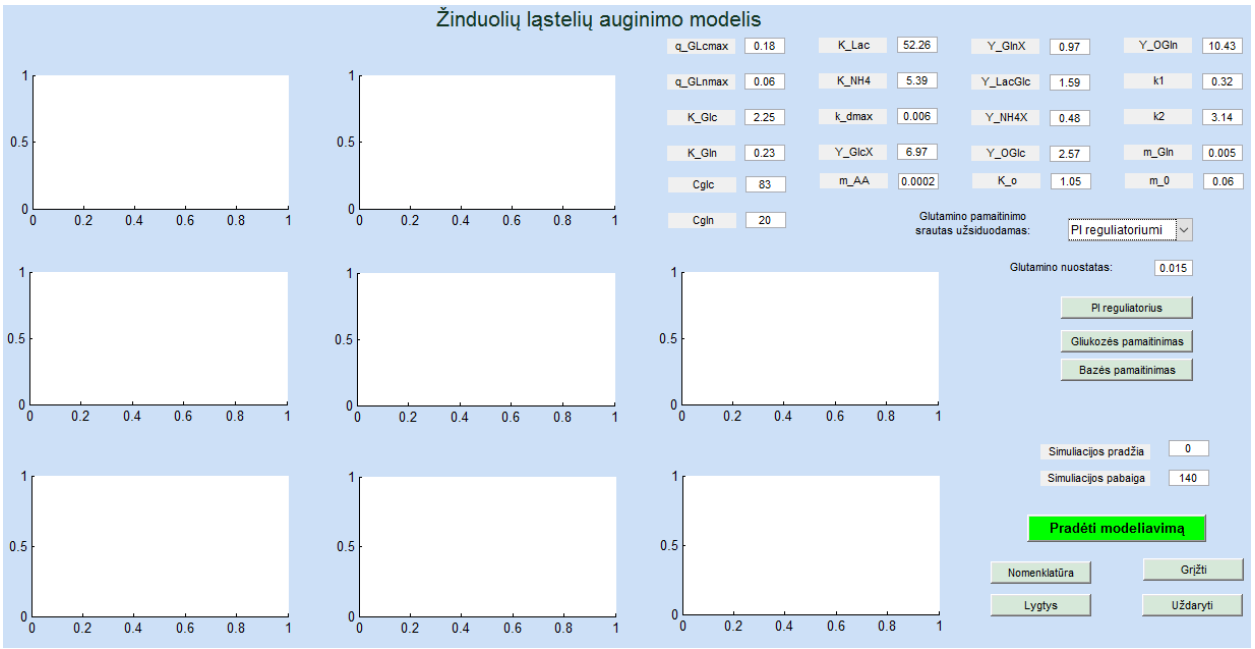


4.23 pav. Žinduolių ląstelių augimo matematinis modelis realizuotas *MATLAB/Simulink*

4.3.2. Vartotojo sąsaja žinduolių ląstelių auginimo modeliui

Žinduolių ląstelių matematiniam modeliui, kaip ir mielėms su E. coli bakterijoms, sukurta grafinė vartotojo sąsaja (4.24 pav.). Sąsajoje yra stebimas tik vienas iš pamaitinimo srautų, t.y. glutaminas. Jis stebimas todėl, nes vėliau bus atliekamas glutamino koncentracijos valdymas keičiant būtent jo įtekėjimo srautą.

Vartotojo sąsajai realizuoti programos kodas pateiktas 8 priede.



4.24 pav. Žinduolių ląstelių auginimo modelio grafinė sąsaja

Grafinėje sąsajoje paspaudus mygtukus „Lygtys“ ir „Nomenklatūra“ atitinkamai atidaromi paveiksliukai, kuriuose pavaizduotos naudotos masių balanso lygtys, bei žinduolių ląstelių auginimo matematinio modelio nomenklatūra:

Bendras ląstelių kiekis:

$$\frac{dX_t}{dt} = \mu_{net} \cdot X_V - \frac{F}{W} \cdot X_t,$$

Gyvybingų ląstelių kiekis:

$$\frac{dX_V}{dt} = \mu \cdot X_V - \frac{F}{W} \cdot X_V,$$

Gliukozės koncentracija:

$$\frac{dGlc}{dt} = -q_{Glc} \cdot X_V - \frac{(F_{Gln} + F_{base})}{W} + \frac{F_{Glc}}{W} \cdot (c_{F_{Glc}} - Glc),$$

Glutamino koncentracija:

$$\frac{dGln}{dt} = -q_{Gln} \cdot X_V - k_{deg} \cdot Gln - \frac{(F_{Glc} + F_{base})}{W} + \frac{F_{Gln}}{W} \cdot (c_{F_{Gln}} - Gln),$$

Laktato koncentracija:

$$\frac{dLac}{dt} = q_{Lac} \cdot X_V - \frac{F}{W} \cdot Lac,$$

Amonio sulfato koncentracija:

$$\frac{dNH_4}{dt} = q_{NH_4} \cdot X_V + k_{deg} \cdot Gln - \frac{F}{W} \cdot NH_4,$$

Terpės tūris:

$$\frac{dW}{dt} = F = F_{Glc} + F_{Gln} + F_{Base}.$$

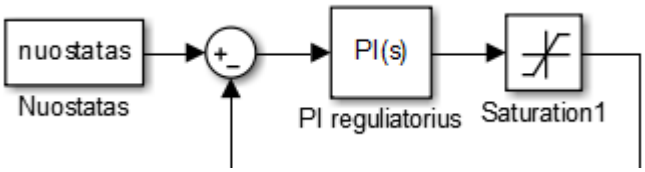
4.25 pav. Masės balanso lygtys žinduolių ląstelių matematiniai modeliui

q_GLcmax	maksimalus gliukozės sunaudojimo greitis	mmol 10^9 ląstelių$^{-1}$ h^{-1}
q_GLnmax	maksimalus glutamino sunaudojimo greitis	mmol 10^9 ląstelių$^{-1}$ h^{-1}
K_Glc	gliukozės slopinimo koeficientas	mM
K_Gln	glutamino slopinimo koeficientas	mM
Cglc	įtekančiame sraute esančios gliukozės koncentracija	mM
Cgln	įtekančiame sraute esanti glutamino koncentracija	mM
K_Lac	laktato slopinimo koeficientas	mM
K_NH4	amonio slopinimo koeficientas	mM
k_dmax	maksimalus ląstelių irimo greitis	h^{-1}
Y_GlcX	gliukozės/biomasės išeiga	mmol 10^9 ląstelių$^{-1}$
m_AA	amonio gamybos greitis	mmol 10^9 ląstelių$^{-1}$ h^{-1}
Y_GlnX	glutamino/biomasės išeiga	mmol 10^9 ląstelių$^{-1}$
Y_LacGlc	laktato/gliukozės išeiga	mmol 10^9 ląstelių$^{-1}$
Y_NH4X	amonio/biomasės išeiga	mmol 10^9 ląstelių$^{-1}$
Y_OGlc	gliukozės/biomasės išeiga	mmol 10^9 ląstelių$^{-1}$
K_o	deguonies slopinimo konstanta	mM
Y_OGln	deguonies/glutamino išeiga	mmol 10^9 ląstelių$^{-1}$
k1	konstanta	mM
k2	konstanta	mM
m_Gln	glutamino palaikymo konstanta	mmol 10^9 ląstelių$^{-1}$ h^{-1}
m_0	deguonies palaikymo konstanta	mmol 10^9 ląstelių$^{-1}$ h^{-1}

4.26 pav. Žinduolių ląstelių matematinio modelio nomenklatūra

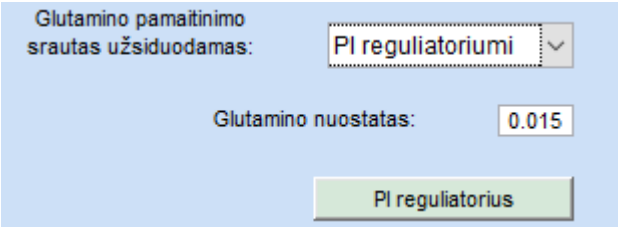
4.3.3. Glutamino koncentracijos valdymas

Šiam modeliui, kaip buvo minėta 4.3.2 skyriuje, realizuotas glutamino koncentracijos valdymas, keičiant jo pamaitinimo srautą. Jo valdymui naudojamas PI reguliatorius:



4.27 pav. Glutamino koncentracijos valdymas

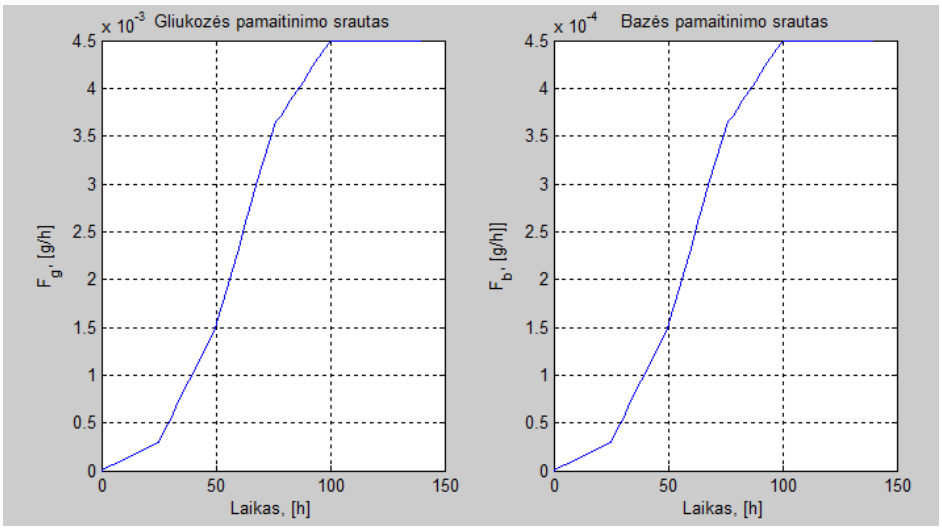
Vartotojas grafinėje sąsajoje turi galimybę pasirinkti, kokiu režimu modeliuoti procesą, bei gali lengvai keisti norimą glutamino nuostatą (4.28 pav.).



4.28 pav. Glutamino valdymas

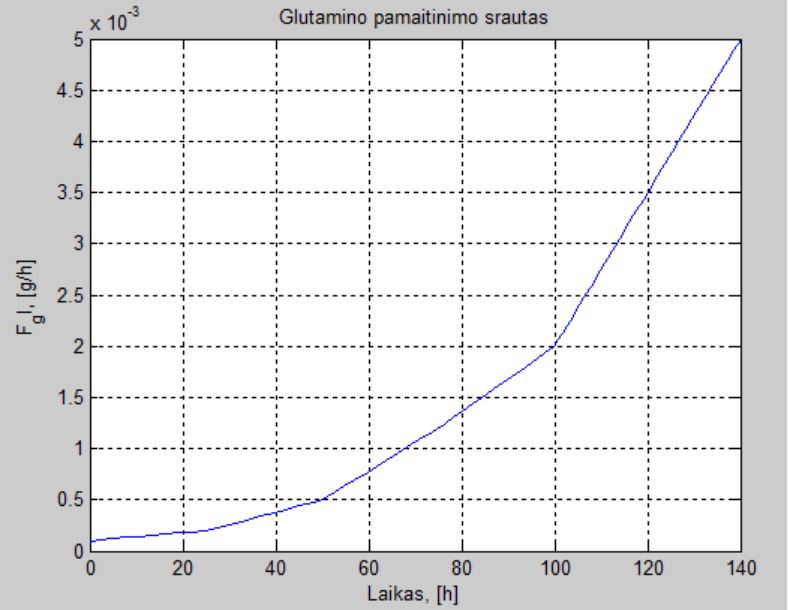
4.3.4. Žinduolių ląstelių modelio modeliavimo rezultatai

Žinduolių ląstelių matematiniame modelyje stebimi 7 proceso kintamieji t.y. gyvybingų ląstelių skaičius, bendras ląstelių skaičius, gliukozė, glutaminas, laktatas, amonio sulfatas bei terpės tūris. Tiek atliekant reguliavimą, tiek be jo, gliukozės ir bazės pamaitinimo srautas yra nekeičiamas ir išlieka toks pat (4.29 pav.). Šarmo, reikalingo palaikyti terpės pH, tiekimo srautas priimamas proporcingas gliukozės, srautui $F_b=0.1F_{gl}$



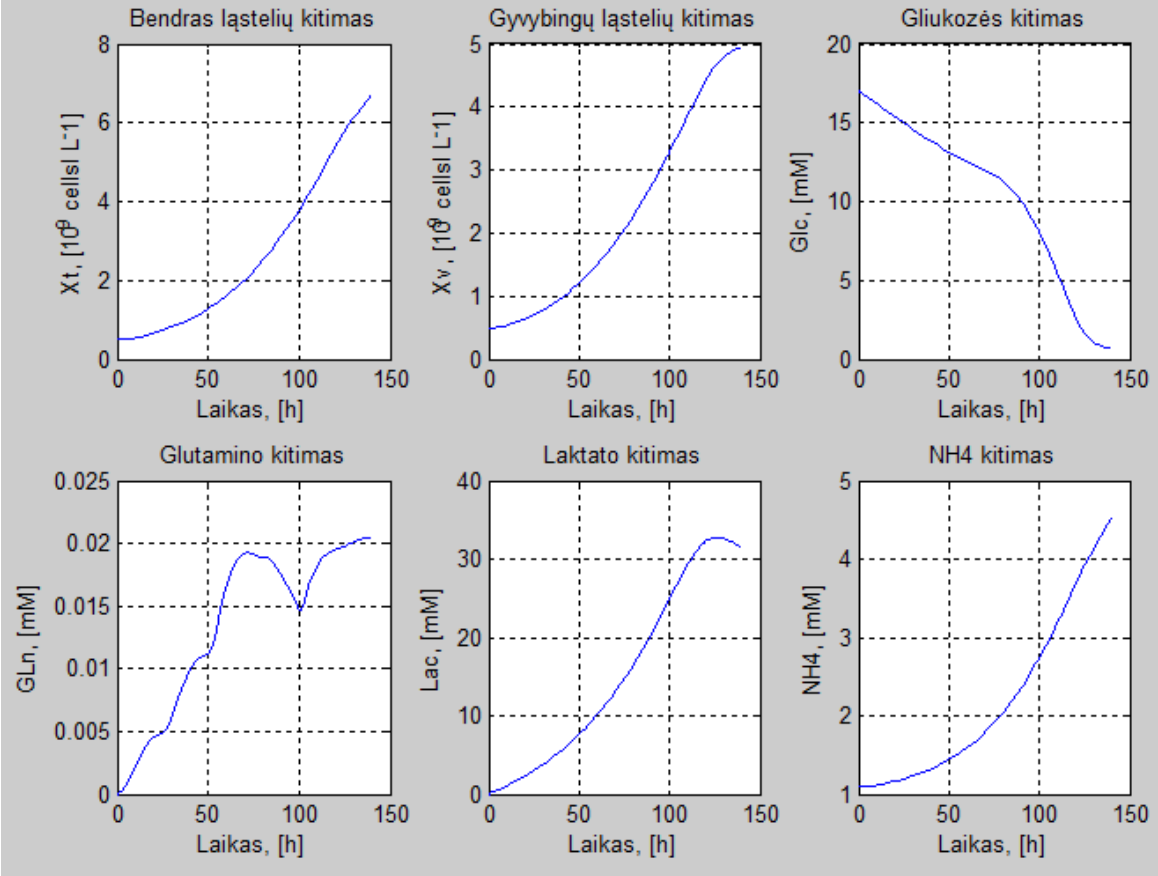
4.29 pav. Gliukozės ir bazės pamaitinimo srautas

Glutamino pamaitinimo srautas, kai sistema veikia be reguliatoriaus pateiktas 4.30 paveikslėlyje.



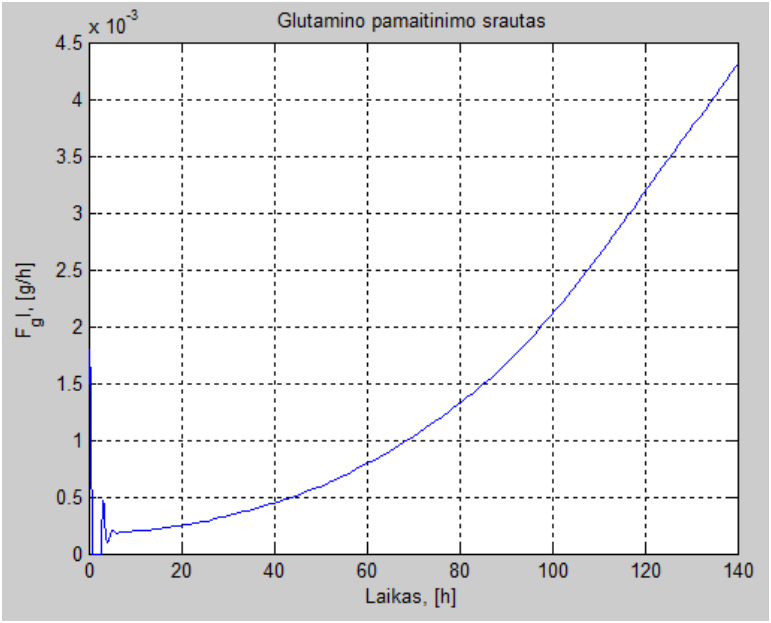
4.30 pav. Glutamino pamaitinimo srautas be glutamino koncentracijos reguliavimo

Modelio rezultatai be reguliavimo pateikti 4.31 pav. Modeliavimo trukmė 140 valandų.

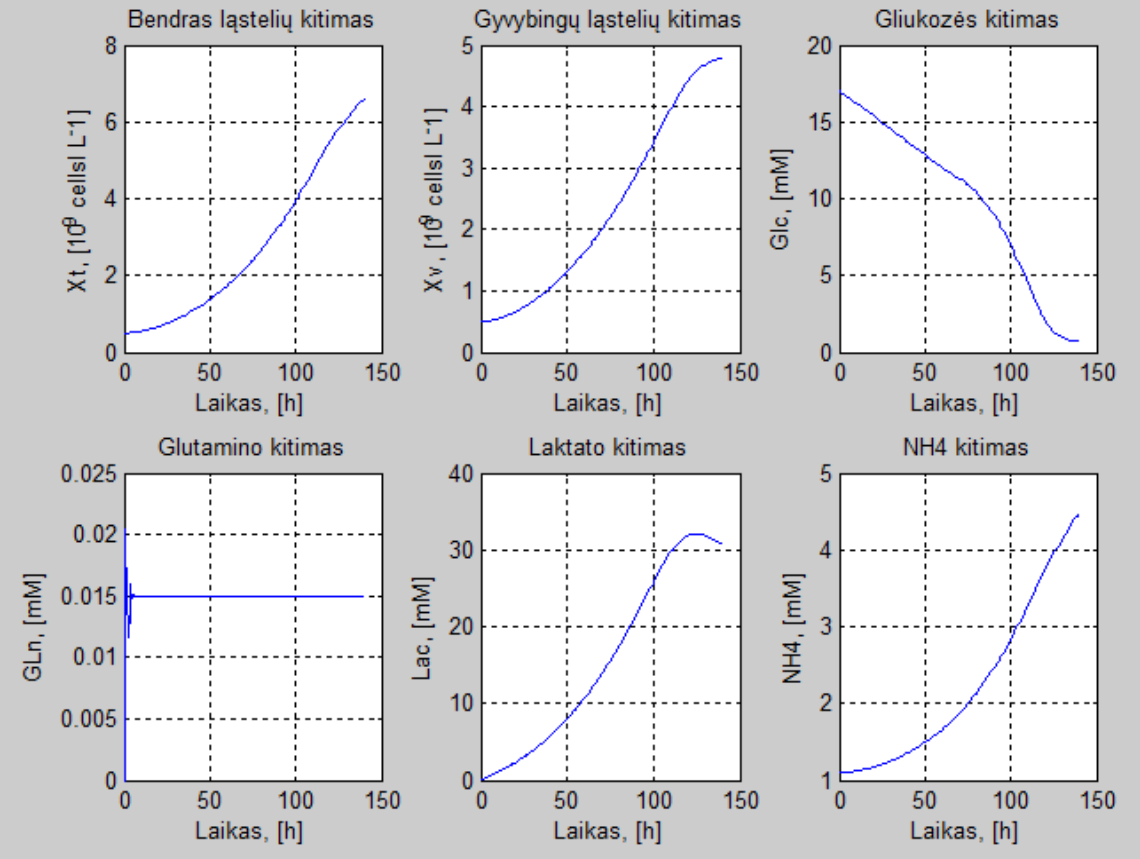


4.31 pav. Žinduolių ląstelių auginimo modeliavimo rezultatai be glutamino koncentracijos valdymo

Atliekant glutamino koncentracijos valdymą, buvo keičiama jo pamaitinimo srautas. Kaip jis keitėsi galime pamatyti 4.32 pav., o kitų kintamųjų kitimą 4.33 pav., kai užduotas glutamino koncentracijos nuostatas: 0,015 mM.



4.32 pav. Glutamino pamaitinimo srautas atliekant glutamino koncentracijos terpėje reguliavimą



4.33 pav. Žinduolių ląstelių auginimo modeliavimo rezultatai atliekant glutamino koncentracijos terpėje reguliavimą

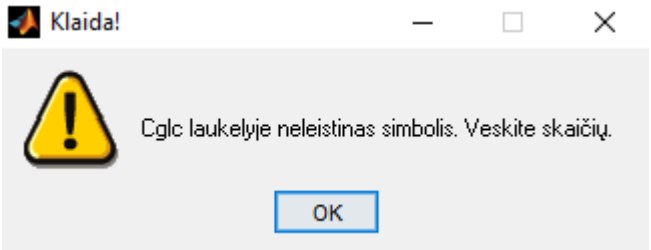
4.4. „*Grįžti*“ ir „*Uždaryti*“ mygtukai

Siekiant padaryti patogesnį modelių perėjimą, iš vieno į kitą, buvo sukurti papildomi mygtukai: „*Grįžti*“ ir „*Uždaryti*“. Paspaudus mygtuką „*Uždaryti*“ yra išjungiama vartotojo sąsaja su kuria tuo metu dirbama, o nuspaudus „*Grįžti*“ vartotojas nukreipiamas į pradinį langą (4.34 pav.), kuriame galima pasirinktį norimą biotechnologinį procesą ir jį analizuoti.

4.34 pav. Pradinis langas

4.5. Apsauga nuo neleistinų simbolių

Siekiant užtikrinti stabilų programos veikimą bei išvengti rezultatų iškraipymo, buvo įvesta apsauga nuo neleistinų simbolių įrašymo. Įrašius laukelyje neleistiną simbolį ar labai nelogišką kintamojo reikšmę yra išmetama klaida, kurioje nurodoma, kokiame laukelyje būtina atlikti pakeitimus. Matematinio modelio simuliacija nepradedam tol kol nėra ištaisomos visos klaidos.



4.35 pav. Apsauga nuo neleistinų simbolių

5. REZULTATAI IR IŠVADOS

1. Darbe išsamiai pagrįstas modeliavimo aplinkos biotechnologinių procesų vystymuisi aktualumas;
2. Apžvelgti mielių kultivavimo procesai, išsiaiškintos ir aprašytos augimo fazės. Sudarytos kinetinio ir matematinio mielių auginimo modelio masių balansu lygtys;
3. Išaiškinta bakterijos E. coli kultivavimo procesai, išanalizuota E. Coli jų auginimo specifika. Sudarytas bakterijų kultivavimo matematinis modelis su masių balanso lygtimis;
4. Apžvelgtos žinduolių ląstelės auginimo technologijos Sudarytas CHO ląstelių auginimo modelis su masių balanso lygtimis;
5. *MATLAB* programiniame pakete, *Simulink* aplinkoje realizuoti visi trys modeliai: mielių, E. coli bakterijų ir žinduolių ląstelių kultivavimui. Jiems sukurta vartotojo grafinės sąsaja. Vartotojo sąsaja leidžia patogiai keisti analizuojamo proceso parametrus bei stebėti proceso kintamųjų kitimą grafine forma;
6. Modeliavimo aplinka suteikia vartotojui galimybę reguliuoti pasirinktus kintamuosius ir stebėti parametrų įtaką reguliavimo kokybei (mielių modelyje reguliuoti ištirpusią deguonies koncentraciją, E. coli modelyje reguliuoti gliukozės koncentraciją terpėje, žinduolių glutamino koncentraciją terpėje);
7. Sukurta modeliavimo aplinka aktuali pasirenkant racionalius kultivavimo režimus, vykdant naujų darbuotojų apmokymus bei gali būti naudinga studijų metu.

6. INFORMACIJOS ŠALTINIŲ SĄRAŠAS

1. RAUDYTĖ, Daiva. *Biotechnologijos* [interaktyvus]. Žiūrėta 2016-05-03 d. Prieiga per: http://pagegiummt.weebly.com/uploads/2/4/4/3/24433358/biotechnologijos.pdf
2. ABEL, O. ir kiti. *Productivity optimization of an industrial semi-batch polymerization reactor under safety constraints* [interaktyvus]. Voketija, 1999 [žiūrėta 2016-05-03 d.]. Prieiga per: http://www.sciencedirect.com.ezproxy.ktu.edu/science/article/pii/S0959152499000499
3. *Grybai* [interaktyvus]. Žiūrėta 2016-03-04 d. Prieiga per: https://lt.wikipedia.org/wiki/Grybai
4. PEČIULIS, Juozas. *Mikrobiologija*. Vilnius: Mokslas. 1983.
5. *What is Yeast* [interaktyvus]. Žiūrėta 2016-03-13 d. Priega per: https://www.wyeastlab.com/he-yeast-fundamentals.cfm
6. SARGŪNAS, Gediminas, Eitvydas JOKUBAUSKIS, Silvija ARLAITĖ ir Lygija ŠKUTIENĖ. *Mielių gamybos būdas* [interaktyvus]. 1998 [žiūrėta 2016-03-13 d.]. Prieiga per: http://www.tb.lt/PIC/Fondas/isradimai/Pilni%20aprasymai/LIETUVOS%20PATENTAI/1998_04_27/4320.PDF
7. *Gimtadienio proga – apie alaus mieles* [interaktyvus]. 2012 [žiūrėta 2016-03-13 d.]. Prieiga per: http://alusalus.lt/2012/11/gimtadienio-proga-apie-alus-mieles/
8. *Bacterial growth* [interaktyvus]. Žiūrėta 2016-03-14 d. Prieiga per: https://en.wikipedia.org/wiki/Bacterial_growth
9. MACHADO, Carlos, Pedro GOMES, Rui SOARES, Silvia PEREITA ir Filomena O. SOARES. *Control of baker's yeast fermentation: PID and fuzzy algorithms* [interaktyvus]. Minho universitetas [žiūrėta 2016-03-18 d.]. Prieiga per: http://repositorium.sdum.uminho.pt/bitstream/1822/6250/1/IECON_FilomenaSoares%2528Final%2529.pdf
10. *Escherichia coli* [interaktyvus]. Žiūrėta 2016-03-25. Prieiga per: https://lt.wikipedia.org/wiki/Escherichia_coli
11. *E. coli Food Poisoning* [interaktyvus]. Žiūrėta 2016-03-25 d. Prieiga per: http://www.about-ecoli.com/
12. KIM, Steven. *What is an Instestinal Infection Due to E. coli?* [interaktyvus]. 2015 [žiūrėta 2016-03-25 d.]. Prieiga per: http://www.healthline.com/health/e-coli-infection#Overview1
13. *Bacteria, An introduction to Earth's largest family* [interaktyvus]. Žiūrėta 2016-03-29 d. Prieiga per:

https://books.google.lt/books?id=PbH_CgAAQBAJ&pg=PA237&lpg=PA237#v=onepage&q&f=false

14. *Gramo dažymas* [interaktyvus]. Žiūrėta 2016-03-25. Prieiga per: https://lt.wikipedia.org/wiki/Gramo_da%C5%BEymas
15. STEWART J, Eric, Richard MADDEN, Gregory PAUL ir Francois TADDEI. *Aging and Death in an Organism That Reproduces by Morphologically Symmetric Division*. 2005 [žiūrėta 2016-03-30 d.]. Prieiga per: http://dx.doi.org/10.1371/journal.pbio.0030045
16. GALVANAUSKAS, Vytautas, Oskars GRIGS, Juris VANAGS, Konstantins DUBENCOVS ir Valerija BOBILEVA. *Model-based optimization and pO2control offed-batch Escherichia coli and Saccharomyces cerevisiae cultivation processes* [interaktyvus]. 2005 [žiūrėta 2016-03-30 d.]. DOI: 10.1002/elsc.201200012. Prieiga per: https://www.researchgate.net/publication/235913607_Model-based_optimization_and_pO2_control_of_fed-batch_Escherichia_coli_and_Saccharomyces_cerevisiae_cultivation_processes
17. GALVANAUSKAS, Vytautas ir Donatas LEVIŠAUSKAS. *Biotechnologinių procesų modeliavimas, optimizavimas ir valdymas*. Vilnius: Vilniaus pedagoginio universiteto leidykla, 2008. ISBN 978-9955-20-281-3.
18. *Masinė žinduolių ląstelių produkcija. Bioreaktoriai* [interaktyvus]. Žiūrėta 2016-03-03 d. Prieiga per: http://www.imcdb.lt/courses/927/
19. *Molekulinė (bendroji) biotechnologija* [interaktyvs]. Žiūrėta 2016-03-03 d. Prieiga per: http://molbio.vdu.lt/medziaga/Molekuline%20(bendroji)%20biotechnologija_tekstas.pdf
20. JARMALAITĖ, Sonata. *Vėžio genomo paslaptys* [interaktyvus]. Žiūrėta 2016-03-03 d. Prieiga per: http://gamta.vdu.lt/bakalaurai/pop_straipsniai/vezio_genomas/vezio_genomas.htm
21. MARQUIS, C.P. *Mammalian cell culture* [interaktyvus]. Naujojo pietų Velso universitetas [žiūrėta 2016-03-03 d.]. Prieiga per: http://www.eolss.net/sample-chapters/c17/e6-58-01-04.pdf
22. AEHLE, Mathias, Kaya BORK, Sebastian SCHAEPE, Artur KUPRIJANOV, Rudiger HORSTKORTE, Rimvydas SIMUTIS ir Andres LUBBERT. *Increasing batch-to-batch reproducibility of CHO-cell culutures using a model predictive control approach* [interaktyvus]. 2012 [žiūrėta 2016-04-06 d.]. DOI: 10.1007/s10616-012-9438-1 Prieiga per: https://www.ncbi.nlm.nih.gov/pubmed/22451075
23. *E. COLI*, [interaktyvus]. Žiūrėta 2017-03-23 d. Prieiga per: https://www.15min.lt/tema/ecoli-6064

24. SCHAEPE, Sebastian, Artur KUPRIJANOV, Rimvydas SIMUTIS ir Andreas LUBBERT. Avoiding overfeeding in high cell density fed-batch cultures of E. coli during the production of heterologous proteins. *Journal of biotechnology*. 2014, pp. 146-153. ISSN 0168-1656

7. PRIEDAI

7.1.Priedas Nr. 1. Augimo greičių išraiškos mielių modelyje

```
function [miuso, miusf, miueo]  = fcn(gliukoze, etanolis, deguonis, parametrai)

% Kintamųjų priskiriamas konkrečioms reikšmėms, jos paiimamos iš workspac'e
% esančio kintomojo matricos

miumax_so = parametrai(1,1);
Ks_so = parametrai(1,2);
miumax_sf = parametrai(1,3);
Ks_sf = parametrai(2,1);
miumax_eo = parametrai(2,2);
Ks_eo = parametrai(2,3);
KO2 = parametrai(3,1);

  % gliukozės gamybos greičio skaičiavimas aerobinėse sąlygose
if gliukoze>0,
    miuso = (miumax_so*gliukoze/(Ks_so+gliukoze))*(deguonis/(KO2+deguonis));
else
    miuso = 0;
end

  % gliukozės gamybos greičio skaičiavimas anaerobinėse sąlygose
if gliukoze>0.5,
    miusf = (miumax_sf*gliukoze/(Ks_sf+gliukoze))*(deguonis/(KO2+deguonis));
else
    miusf = 0;
end

  % etanolios gamybos greičio skaičiavimas anaerobinėse sąlygose
if gliukoze<0.5,
    miueo = miumax_eo*etanolis/(Ks_eo+etanolis)*(deguonis/(KO2+deguonis));
else
    miueo=0;
end
```

7.2. Priedas Nr. 2. Deguonies ir anglies dioksido tiekimo greičiai mielių modelyje

```
function [OTR, CTR]= fcn(deguonis, dioksidas, oras, apsukos, parametrai)

% Kintamųjų priskiriamas konkrečioms reikšmėms

O_tirp = parametrai(1,1);
C_tirp = parametrai(1,2);
kLa1 = parametrai(2,1);
reg = parametrai(2,2);
K1 = parametrai(3,1);
K2 = parametrai(3,2);

V = 10;
kLa2 = K1*(oras/V)+K2*apsukos;

if reg == 0
    kLa = kLa1;
else
    kLa = kLa2;
end

OTR = (O_tirp-deguonis)*kLa;    % deguonies tiekimo greitis
CTR = (dioksidas-C_tirp)*0.001*kLa;   % anglies dioksido tiekimo greitis
```

7.3. Priedas Nr. 3. Vartotojo sąsajos kodas Mielių modeliui

```
function varargout = mieliu_sasaja(varargin)

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @mieliu_sasaja_OpeningFcn, ...
                   'gui_OutputFcn',  @mieliu_sasaja_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

function mieliu_sasaja_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);

function varargout = mieliu_sasaja_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;

function pradeti_Callback(hObject, eventdata, handles)
procesas = get(handles.popupmenu2,'value');
% aprašomas kintamasis kurio nusakomas srauto itėkjimo būdas
if procesas == 1
    reguliavimas = 0;
    assignin('base', 'reguliavimas', reguliavimas);
else
    reguliavimas = 1;
    assignin('base', 'reguliavimas', reguliavimas);
end

stopas = 0;

Yxso=get(handles.Yxso, 'string'); Yxso=str2double(Yxso);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(Yxso) || isnan(Yxso)
    % Formuojamas pranešimas.
    message = sprintf('Yx/s(o) laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
Yxso_apatine = 0.1;
Yxso_virsutine = 1;
if (Yxso < Yxso_apatine) || (Yxso > Yxso_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('Yxso laukelyje neleistinas ribos. Veskite nuo %0.1g iki %0.2g.',
Yxso_apatine, Yxso_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Parametras siunčiamas į Workspac'ą
assignin('base', 'Yxso', Yxso);

Yxsf=get(handles.Yxsf, 'string'); Yxsf=str2double(Yxsf);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(Yxsf) || isnan(Yxsf)
    % Formuojamas pranešimas.
    message = sprintf('Yx/s(f) laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
Yxsf_apatine = 0.1;
Yxsf_virsutine = 1;
if (Yxsf < Yxsf_apatine) || (Yxsf > Yxsf_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('Yxsf laukelyje neleistinas ribos. Veskite nuo %0.1g iki %0.2g.',
Yxsf_apatine, Yxsf_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
```

```
    stopas = 1;
end
% Parametras siunčiamas į Workspac'ą
assignin('base', 'Yxsf', Yxsf);

Yxef=get(handles.Yxef, 'string'); Yxef=str2double(Yxef);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(Yxef) || isnan(Yxef)
    % Formuojamas pranešimas.
    message = sprintf('Yx/e(f) laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
Yxef_apatine = 1;
Yxef_virsutine = 3;
if (Yxef < Yxef_apatine) || (Yxef > Yxef_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('Yxef laukelyje neleistinas ribos. Veskite nuo %0.1g iki %0.2g.',
Yxef_apatine, Yxef_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Parametras siunčiamas į Workspac'ą
assignin('base', 'Yxef', Yxef);

Yxeoe=get(handles.Yxeoe, 'string'); Yxeoe=str2double(Yxeoe);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(Yxeoe) || isnan(Yxeoe)
    % Formuojamas pranešimas.
    message = sprintf('Yx/e(oe) laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
Yxeoe_apatine = 0.1;
Yxeoe_virsutine = 1;
if (Yxeoe < Yxeoe_apatine) || (Yxeoe > Yxeoe_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('Yxeoe laukelyje neleistinas ribos. Veskite nuo %0.1g iki %0.2g.',
Yxeoe_apatine, Yxeoe_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Parametras siunčiamas į Workspac'ą
assignin('base', 'Yxeoe', Yxeoe);

Yxoo=get(handles.Yxoo, 'string'); Yxoo=str2double(Yxoo);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(Yxoo) || isnan(Yxoo)
    % Formuojamas pranešimas.
    message = sprintf('Yx/o(o) laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
Yxoo_apatine = 0.6;
Yxoo_virsutine = 2.5;
if (Yxoo < Yxoo_apatine) || (Yxoo > Yxoo_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('Yxoo laukelyje neleistinas ribos. Veskite nuo %0.1g iki %0.2g.',
Yxoo_apatine, Yxoo_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Parametras siunčiamas į Workspac'ą
assignin('base', 'Yxoo', Yxoo);

Yxooe=get(handles.Yxooe, 'string'); Yxooe=str2double(Yxooe);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(Yxooe) || isnan(Yxooe)
    % Formuojamas pranešimas.
    message = sprintf('Yx/o(oe) laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
```

```
% užsiduodamas apatinė ir žemutinė ribos
Yxooe_apatine = 0.7;
Yxooe_virsutine = 3;
if (Yxooe < Yxooe_apatine) || (Yxooe > Yxooe_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('Yxooe laukelyje neleistinas ribos. Veskite nuo %0.1g iki %0.2g.',
Yxooe_apatine, Yxooe_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Parametras siunčiamas į Workspac'ą
assignin('base', 'Yxooe', Yxooe);

Yxco=get(handles.Yxco, 'string'); Yxco=str2double(Yxco);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(Yxco) || isnan(Yxco)
    % Formuojamas pranešimas.
    message = sprintf('Yx/c(o) laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
Yxco_apatine = 0.1;
Yxco_virsutine = 2;
if (Yxco < Yxco_apatine) || (Yxco > Yxco_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('Yxco laukelyje neleistinas ribos. Veskite nuo %0.1g iki %0.2g.',
Yxco_apatine, Yxco_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Parametras siunčiamas į Workspac'ą
assignin('base', 'Yxco', Yxco);

Yxcf=get(handles.Yxcf, 'string'); Yxcf=str2double(Yxcf);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(Yxcf) || isnan(Yxcf)
    % Formuojamas pranešimas.
    message = sprintf('Yx/c(f) laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
Yxcf_apatine = 0.6;
Yxcf_virsutine = 3.5;
if (Yxcf < Yxcf_apatine) || (Yxcf > Yxcf_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('Yxcf laukelyje neleistinas ribos. Veskite nuo %0.1g iki %0.2g.',
Yxcf_apatine, Yxcf_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Parametras siunčiamas į Workspac'ą
assignin('base', 'Yxcf', Yxcf);

Yxcoe=get(handles.Yxcoe, 'string'); Yxcoe=str2double(Yxcoe);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(Yxcoe) || isnan(Yxcoe)
    % Formuojamas pranešimas.
    message = sprintf('Yx/c(oe) laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
Yxcoe_apatine = 0.6;
Yxcoe_virsutine = 3;
if (Yxcoe < Yxcoe_apatine) || (Yxcoe > Yxcoe_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('Yxcoe laukelyje neleistinas ribos. Veskite nuo %0.1g iki %0.2g.',
Yxcoe_apatine, Yxcoe_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Parametras siunčiamas į Workspac'ą
assignin('base', 'Yxcoe', Yxcoe);

kLa=get(handles.kLa, 'string'); kLa=str2double(kLa);
```

```
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(kLa) || isnan(kLa)
    % Formuojamas pranešimas.
    message = sprintf('kLa laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
kLa_apatine = 200;
kLa_virsutine = 800;
if (kLa < kLa_apatine) || (kLa > kLa_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('kLa laukelyje neleistinas ribos. Veskite nuo %0.3g iki %0.3g.',
kLa_apatine, kLa_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end

O_tirp=get(handles.O_tirp, 'string'); O_tirp=str2double(O_tirp);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(O_tirp) || isnan(O_tirp)
    % Formuojamas pranešimas.
    message = sprintf('O_tirp laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
O_tirp_apatine = 0.001;
O_tirp_virsutine = 0.01;
if (O_tirp < O_tirp_apatine) || (O_tirp > O_tirp_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('O_tirp laukelyje neleistinas ribos. Veskite nuo %0.3g iki %0.2g.',
O_tirp_apatine, O_tirp_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end

C_tirp=get(handles.C_tirp, 'string'); C_tirp=str2double(C_tirp);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(C_tirp) || isnan(C_tirp)
    % Formuojamas pranešimas.
    message = sprintf('C_tirp laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
C_tirp_apatine = 0.00001;
C_tirp_virsutine = 0.01;
if (C_tirp < C_tirp_apatine) || (C_tirp > C_tirp_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('C_tirp laukelyje neleistinas ribos. Veskite nuo %0.10g iki %0.3g.',
C_tirp_apatine, C_tirp_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end

Sf=get(handles.Sf, 'string'); Sf=str2double(Sf);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(Sf) || isnan(Sf)
    % Formuojamas pranešimas.
    message = sprintf('Sf laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
Sf_apatine = 100;
Sf_virsutine = 600;
if (Sf < Sf_apatine) || (Sf > Sf_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('Sf laukelyje neleistinas ribos. Veskite nuo %0.3g iki %0.3g.', Sf_apatine,
Sf_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Parametras siunčiamas į Workspac'ą
assignin('base', 'Sf', Sf);
```

```
miumax_so=get(handles.miumax_so, 'string'); miumax_so=str2double(miumax_so);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(miumax_so) || isnan(miumax_so)
    % Formuojamas pranešimas.
    message = sprintf('miumax_so laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
miumax_so_apatine = 0.01;
miumax_so_virsutine = 0.4;
if (miumax_so < miumax_so_apatine) || (miumax_so > miumax_so_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('miumax_so laukelyje neleistinas ribos. Veskite nuo %0.1g iki %0.2g.',
miumax_so_apatine, miumax_so_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end

Ks_so=get(handles.Ks_so, 'string'); Ks_so=str2double(Ks_so);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(Ks_so) || isnan(Ks_so)
    % Formuojamas pranešimas.
    message = sprintf('Ks_so laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
Ks_so_apatine = 0.01;
Ks_so_virsutine = 0.4;
if (Ks_so < Ks_so_apatine) || (Ks_so > Ks_so_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('Ks_so laukelyje neleistinas ribos. Veskite nuo %0.1g iki %0.2g.',
Ks_so_apatine, Ks_so_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end

miumax_sf=get(handles.miumax_sf, 'string'); miumax_sf=str2double(miumax_sf);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(miumax_sf) || isnan(miumax_sf)
    % Formuojamas pranešimas.
    message = sprintf('miumax_sf laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
miumax_sf_apatine = 0.01;
miumax_sf_virsutine = 0.1;
if (miumax_sf < miumax_sf_apatine) || (miumax_sf > miumax_sf_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('miumax_sf laukelyje neleistinas ribos. Veskite nuo %0.1g iki %0.2g.',
miumax_sf_apatine, miumax_sf_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end

Ks_sf=get(handles.Ks_sf, 'string'); Ks_sf=str2double(Ks_sf);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(Ks_sf) || isnan(Ks_sf)
    % Formuojamas pranešimas.
    message = sprintf('Ks_sf laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
Ks_sf_apatine = 5;
Ks_sf_virsutine = 15;
if (Ks_sf < Ks_sf_apatine) || (Ks_sf > Ks_sf_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('Ks_sf laukelyje neleistinas ribos. Veskite nuo %0.1g iki %0.2g.',
Ks_sf_apatine, Ks_sf_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
```

```
miumax_eo=get(handles.miumax_eo, 'string'); miumax_eo=str2double(miumax_eo);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(miumax_eo) || isnan(miumax_eo)
    % Formuojamas pranešimas.
    message = sprintf('miumax_eo laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
miumax_eo_apatine = 0.01;
miumax_eo_virsutine = 0.1;
if (miumax_eo < miumax_eo_apatine) || (miumax_eo > miumax_eo_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('miumax_eo laukelyje neleistinas ribos. Veskite nuo %0.1g iki %0.2g.',
miumax_eo_apatine, miumax_eo_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end

Ks_eo=get(handles.Ks_eo, 'string'); Ks_eo=str2double(Ks_eo);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(Ks_eo) || isnan(Ks_eo)
    % Formuojamas pranešimas.
    message = sprintf('Ks_eo laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
Ks_eo_apatine = 0.5;
Ks_eo_virsutine = 2;
if (Ks_eo < Ks_eo_apatine) || (Ks_eo > Ks_eo_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('Ks_eo laukelyje neleistinas ribos. Veskite nuo %0.1g iki %0.2g.',
Ks_eo_apatine, Ks_eo_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end

KO2=get(handles.KO2, 'string'); KO2=str2double(KO2);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(KO2) || isnan(KO2)
    % Formuojamas pranešimas.
    message = sprintf('KO2 laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
KO2_apatine = 0.0001;
KO2_virsutine = 0.001;
if (KO2 < KO2_apatine) || (KO2 > KO2_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('KO2 laukelyje neleistinas ribos. Veskite nuo %0.1g iki %0.2g.',
KO2_apatine, KO2_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end

K1=get(handles.K1, 'string'); K1=str2double(K1);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(K1) || isnan(K1)
    % Formuojamas pranešimas.
    message = sprintf('K1 laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
K1_apatine = 60;
K1_virsutine = 240;
if (K1 < K1_apatine) || (K1 > K1_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('K1 laukelyje neleistinas ribos. Veskite nuo %0.2g iki %0.3g.', K1_apatine,
K1_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
```

```
K2=get(handles.K2, 'string'); K2=str2double(K2);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(K2) || isnan(K2)
    % Formuojamas pranešimas.
    message = sprintf('K2 laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
K2_apatine = 0.1;
K2_virsutine = 1.5;
if (K2 < K2_apatine) || (K2 > K2_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('K2 laukelyje neleistinas ribos. Veskite nuo %0.1g iki %0.2g.', K2_apatine,
K2_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end

% Simulink'o realizacija
% Simuliacijos laiko aprašymas
Ton=get(handles.laikas_pradzia, 'string'); ton=str2double(Ton);
if isempty(ton) || isnan(ton)
    % Formuojamas pranešimas.
    message = sprintf('Simuliacijos pradžios laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
ton_apatine = 0;
ton_virsutine = 10;
if (ton < ton_apatine) || (ton > ton_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('Simuliacijos pradžios laukelyje neleistinas ribos. Veskite nuo %0.2g iki
%0.3g.', ton_apatine, ton_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end

Toff=get(handles.laikas_pabaiga, 'string'); toff=str2double(Toff);
if isempty(toff) || isnan(toff)
    % Formuojamas pranešimas.
    message = sprintf('Simuliacijos pabaigos laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
toff_apatine = 1;
toff_virsutine = 100;
if (toff < toff_apatine) || (toff > toff_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('Simuliacijos pabaigos laukelyje neleistinas ribos. Veskite nuo %0.2g iki
%0.3g.', toff_apatine, toff_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end

nuostatas=get(handles.nuostatas, 'string'); nuostatas=str2double(nuostatas);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(nuostatas) || isnan(nuostatas)
    % Formuojamas pranešimas.
    message = sprintf('Nuostato laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
nuostatas_apatine = 0.0001;
nuostatas_virsutine = 0.01;
if (nuostatas < nuostatas_apatine) || (nuostatas > nuostatas_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('Nuostato laukelyje neleistinas ribos. Veskite nuo %0.6g iki %0.2g.',
nuostatas_apatine, nuostatas_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
```

```
% Parametras siunčiamas į Workspac'ą
assignin('base', 'nuostatas', nuostatas);

% parametrai surašymas į matricą
parametrai = [ miumax_so, Ks_so,      miumax_sf;
               Ks_sf,     miumax_eo,  Ks_eo;
               KO2,        0,         0;
              ];
assignin('base', 'parametrai', parametrai);

% parametrai skirti tiekimo greičiui
parametrai2 = [ O_tirp, C_tirp;
                kLa,    reguliavimas;
                K1,     K2;
                ];
assignin('base', 'parametrai2', parametrai2);

% Pradedama simuliacija jeigu visi simboliai yra leistini
if stopas == 0
options=simset('DstWorkspace', 'current',...
               'SrcWorkspace', 'current',...
               'OutputPoints', 'specified',...
               'SaveFormat', 'Array');
% open('mieliu_modelis.slx');
sim('mieliu_modelis.slx', [ton, toff],options);

% Grafikų piešimas
% Biomasės
axes(handles.g_biomase);
g_biomase=plot(biomase(:,1), biomase(:,2), 'k'); grid on
title('Biomasės kitimas'); ylabel ('X, [g/l]'); xlabel ('Laikas, [h]');
% Gliukozės
axes(handles.g_gliukoze);
g_gliukoze=plot(gliukoze(:,1), gliukoze(:,2), 'k'); grid on
title('Ištirpusios gliukozės kitimas'); ylabel ('S, [g/l]'); xlabel ('Laikas, [h]');
% Etanolio
axes(handles.g_etanolis);
g_etanolis=plot(etanolis(:,1), etanolis(:,2), 'k'); grid on
title('Etanolio kitimas'); ylabel ('E, [g/l]'); xlabel ('Laikas, [h]');
% Deguonies
axes(handles.g_deguonis);
g_deguonis=plot(deguonis(:,1), deguonis(:,2), 'k'); grid on
title('Ištirpusios deguonies kitimas'); ylabel ('pO2, [g/l]'); xlabel ('Laikas, [h]');
% Anglies dioksidas
axes(handles.g_dioksidas);
g_dioksidas=plot(dioksidas(:,1), dioksidas(:,2), 'k'); grid on
title('Ištirpusio anglies dioksido kitimas'); ylabel ('C, [g/l]'); xlabel ('Laikas, [h]');
% Tūris
axes(handles.g_turis);
g_turis=plot(turis(:,1), turis(:,2), 'k'); grid on
title('Tūrio kitimas'); ylabel ('V, [l]'); xlabel ('Laikas, [h]');
% Pamaitininimo srautas
axes(handles.g_feedingas);
g_feedingas=plot(pamaitinimas(:,1), pamaitinimas(:,2), 'k'); grid on
title('Pamaitinimo srauto kitimas'); ylabel ('F, [g/h]'); xlabel ('Laikas, [h]');
end

function nomenklatura_Callback(hObject, eventdata, handles)
norm = imread('nomenklatura.jpg');
figure(1);
imshow(norm);

function lygtys_Callback(hObject, eventdata, handles)
form = imread('formules.jpg');
figure(1);
imshow(form);


function feeding_Callback(hObject, eventdata, handles)
open_system('mieliu_modelis/Feeding')

function gryzti_Callback(hObject, eventdata, handles)
set(gcf,'Visible','off')
run('pradinis.m')

function uzdaryti_Callback(hObject, eventdata, handles)
close

function popupmenu2_Callback(hObject, eventdata, handles)
procesas = get(hObject,'value');
```

```
if procesas == 1
%     Jei pasirinktas rankiniu būdu užduodamas srautas tai įjungiamas
%     mygtukas leidžiantis keisti parametrus
    set(handles.PI1,'visible','off')
    set(handles.PI2,'visible','off')
    set(handles.t_nuostatas,'visible','off')
    set(handles.nuostatas,'visible','off')
else
%     Jei pasirenkamas PI2 reguliatoriumi užduodamas srautas tai įjungiamas
%     mygtukas leidžiantis keisti reguliatoriaus parametrus
    set(handles.PI1 ,'visible','on')
    set(handles.PI2 ,'visible','on')
    set(handles.t_nuostatas,'visible','on')
    set(handles.nuostatas,'visible','on')
end

function nuostatas_Callback(hObject, eventdata, handles)

function PI2_Callback(hObject, eventdata, handles)
open_system('mieliu_modelis/PI reguliatorius2')

function PI1_Callback(hObject, eventdata, handles)
open_system('mieliu_modelis/PI reguliatorius1')
```

7.4. Priedas Nr. 4. Reakcijos greičio augimo išraiškos E. coli modelyje

```
function [miu, si, aa]  = fcn(parametrai, gliukoze, acetatas, biomase, laikas)

% Kintamųjų priskiriamas konkrečioms reikšmėms

si_kritine = parametrai(1,1);
Ks = parametrai(1,2);
Ksi = parametrai(1,3);
Kai = parametrai(1,4);
Yxa = parametrai(2,1);
m = parametrai(2,2);
Yas = parametrai(2,3);
Ka = parametrai(2,4);
si_max1 = parametrai(3,1);
Kxx1 = parametrai(3,2);
Yxs1 = parametrai(3,3);
si_max2 = parametrai(3,4);
Kxx2 = parametrai(4,1);
Yxs2 = parametrai(4,2);
tind = parametrai(4,3);

   % Parametrų išskaidymas prieš indukciją ir po jos
if laikas<=tind,
    si_max=si_max1;
    Kxx=Kxx1;
    Yxs=Yxs1;
  else
    si_max=si_max2;
    Kxx=Kxx2;
    Yxs=Yxs2;
end

% substrato greičio lygtis
si = si_max*gliukoze/(Ks+gliukoze)*...
Ksi/(Ksi+gliukoze)*...
Kai/(Kai+acetatas) - Kxx*biomase*biomase;

% apskaičiuojamas pagamintas ir sunaudotas acetatas bei biomasės augimo
% gritis
if si<=si_kritine
  ac = (1-si/si_kritine)*acetatas/(acetatas+Ka);
  ap = 0;
  miu = si*Yxs+ac*Yxa - m;
else
  ac = 0;
  ap = (si-si_kritine)*Yas;
  miu = si_kritine*Yxs - m;
end
```

```
aa = ap-ac;
```

7.5. Priedas Nr. 5. Įtekančio srauto greičio išraiškos E. coli modelyje

```
function Fin    = fcn(miu, biomase, turis, Fs, parametrai)

% Kintamųjų priskiriamas konkrečioms reikšmėms

Ybx = parametrai(1,1);
Yox = parametrai(1,2);
om = parametrai(1,3);
Yo2x = parametrai(2,1);
ke = parametrai(2,2);

% buferinio (šarminio) tirpalo tiekimo greitis
Fbase = Ybx*miu*biomase*turis;
% deguonies sunaudojimo greitis
OUR  = Yox*miu*biomase+om*biomase;
% anglies išmetimas su CO2 ištekančiose dujose
Fco2    = Yo2x*OUR*turis;
% vandens garų išgaruotas kiekis
Fevp    = ke*turis;
% į bioreaktorių įtekantis srautas
Fin     = Fs + Fbase + Fco2 + Fevp;
```

7.6.Priedas Nr. 6. Vartotojo sąsajos kodas E. coli modeliui

```
function varargout = ecoli_sasaja(varargin)
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @ecoli_sasaja_OpeningFcn, ...
                   'gui_OutputFcn',  @ecoli_sasaja_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

function ecoli_sasaja_ResizeFcn(hObject, eventdata, handles)

function ecoli_sasaja_OpeningFcn(hObject, eventdata, handles, varargin)

handles.output = hObject;
guidata(hObject, handles);


function varargout = ecoli_sasaja_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;

function pradeti_Callback(hObject, eventdata, handles)
procesas = get(handles.popupmenu1,'value');
% aprašomas kintamasis kurio nusakomas srauto itėkjimo būdas
if procesas == 1
    reguliavimas = 0;
    assignin('base', 'reguliavimas', reguliavimas);
else
    reguliavimas = 1;
    assignin('base', 'reguliavimas', reguliavimas);
end
    stopas = 0;
```

```
si_kritine=get(handles.si_kritine, 'string'); si_kritine=str2double(si_kritine);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(si_kritine) || isnan(si_kritine)
    % Formuojamas pranešimas.
    message = sprintf('si kritine laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
si_kritine_apatine = 0.1;
si_kritine_virsutine = 3.2;
if (si_kritine < si_kritine_apatine) || (si_kritine > si_kritine_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('si kritine laukelyje neleistinas ribos. Veskite nuo %0.1g iki %0.2g.',
si_kritine_apatine, si_kritine_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end

Ks=get(handles.Ks, 'string'); Ks=str2double(Ks);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(Ks) || isnan(Ks)
    % Formuojamas pranešimas.
    message = sprintf('Ks laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
Ks_apatine = 0.001;
Ks_virsutine = 1;
if (Ks < Ks_apatine) || (Ks > Ks_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('Ks laukelyje neleistinas ribos. Veskite nuo %0.2g iki %0.2g.', Ks_apatine,
Ks_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end

Ksi=get(handles.Ksi, 'string'); Ksi=str2double(Ksi);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(Ksi) || isnan(Ksi)
    % Formuojamas pranešimas.
    message = sprintf('Ksi laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
Ksi_apatine = 50;
Ksi_virsutine = 150;
if (Ksi < Ksi_apatine) || (Ksi > Ksi_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('Ksi laukelyje neleistinas ribos. Veskite nuo %0.2g iki %0.3g.',
Ksi_apatine, Ksi_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end

Kai=get(handles.Kai, 'string'); Kai=str2double(Kai);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(Kai) || isnan(Kai)
    % Formuojamas pranešimas.
    message = sprintf('Kai laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
Kai_apatine = 5;
Kai_virsutine = 65;
if (Kai < Kai_apatine) || (Kai > Kai_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('Kai laukelyje neleistinas ribos. Veskite nuo %0.2g iki %0.3g.',
Kai_apatine, Kai_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
```

```
Yxa=get(handles.Yxa, 'string'); Yxa=str2double(Yxa);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(Yxa) || isnan(Yxa)
    % Formuojamas pranešimas.
    message = sprintf('Yxa laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
Yxa_apatine = 0;
Yxa_virsutine = 1;
if (Yxa < Yxa_apatine) || (Yxa > Yxa_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('Yxa laukelyje neleistinas ribos. Veskite nuo %0.2g iki %0.3g.',
Yxa_apatine, Yxa_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end

m=get(handles.m, 'string'); m=str2double(m);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(m) || isnan(m)
    % Formuojamas pranešimas.
    message = sprintf('m laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
m_apatine = 0;
m_virsutine = 1;
if (m < m_apatine) || (m > m_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('m laukelyje neleistinas ribos. Veskite nuo %0.2g iki %0.3g.', m_apatine,
m_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end

Yas=get(handles.Yas, 'string'); Yas=str2double(Yas);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(Yas) || isnan(Yas)
    % Formuojamas pranešimas.
    message = sprintf('Yas laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
Yas_apatine = 0;
Yas_virsutine = 2;
if (Yas < Yas_apatine) || (Yas > Yas_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('Yas laukelyje neleistinas ribos. Veskite nuo %0.2g iki %0.3g.',
Yas_apatine, Yas_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end

Ka=get(handles.Ka, 'string'); Ka=str2double(Ka);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(Ka) || isnan(Ka)
    % Formuojamas pranešimas.
    message = sprintf('Ka laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
Ka_apatine = 0.1;
Ka_virsutine = 2;
if (Ka < Ka_apatine) || (Ka > Ka_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('Ka laukelyje neleistinas ribos. Veskite nuo %0.2g iki %0.3g.', Ka_apatine,
Ka_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
```

```
si_max1=get(handles.si_max1, 'string'); si_max1=str2double(si_max1);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(si_max1) || isnan(si_max1)
    % Formuojamas pranešimas.
    message = sprintf('si_max1 laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
si_max1_apatine = 0.5;
si_max1_virsutine = 3;
if (si_max1 < si_max1_apatine) || (si_max1 > si_max1_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('si_max1 laukelyje neleistinas ribos. Veskite nuo %0.2g iki %0.3g.',
si_max1_apatine, si_max1_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end

Kxx1=get(handles.Kxx1, 'string'); Kxx1=str2double(Kxx1);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(Kxx1) || isnan(Kxx1)
    % Formuojamas pranešimas.
    message = sprintf('Kxx1 laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
Kxx1_apatine = 0;
Kxx1_virsutine = 0.5;
if (Kxx1 < Kxx1_apatine) || (Kxx1 > Kxx1_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('Kxx1 laukelyje neleistinas ribos. Veskite nuo %0.2g iki %0.3g.',
Kxx1_apatine, Kxx1_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end

Yxs1=get(handles.Yxs1, 'string'); Yxs1=str2double(Yxs1);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(Yxs1) || isnan(Yxs1)
    % Formuojamas pranešimas.
    message = sprintf('Yxs1 laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
Yxs1_apatine = 0;
Yxs1_virsutine = 1;
if (Yxs1 < Yxs1_apatine) || (Yxs1 > Yxs1_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('Yxs1 laukelyje neleistinas ribos. Veskite nuo %0.2g iki %0.3g.',
Yxs1_apatine, Yxs1_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end

si_max2=get(handles.si_max2, 'string'); si_max2=str2double(si_max2);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(si_max2) || isnan(si_max2)
    % Formuojamas pranešimas.
    message = sprintf('si_max2 laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
si_max2_apatine = 0.5;
si_max2_virsutine = 3;
if (si_max2 < si_max2_apatine) || (si_max2 > si_max2_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('si_max2 laukelyje neleistinas ribos. Veskite nuo %0.2g iki %0.3g.',
si_max2_apatine, si_max2_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
```

```
Kxx2=get(handles.Kxx2, 'string'); Kxx2=str2double(Kxx2);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(Kxx2) || isnan(Kxx2)
    % Formuojamas pranešimas.
    message = sprintf('Kxx2 laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
Kxx2_apatine = 0;
Kxx2_virsutine = 0.5;
if (Kxx2 < Kxx2_apatine) || (Kxx2 > Kxx2_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('Kxx2 laukelyje neleistinas ribos. Veskite nuo %0.2g iki %0.3g.',
Kxx2_apatine, Kxx2_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end

Yxs2=get(handles.Yxs2, 'string'); Yxs2=str2double(Yxs2);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(Yxs2) || isnan(Yxs2)
    % Formuojamas pranešimas.
    message = sprintf('Yxs2 laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
Yxs2_apatine = 0;
Yxs2_virsutine = 1;
if (Yxs2 < Yxs2_apatine) || (Yxs2 > Yxs2_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('Yxs2 laukelyje neleistinas ribos. Veskite nuo %0.2g iki %0.3g.',
Yxs2_apatine, Yxs2_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end

tind=get(handles.tind, 'string'); tind=str2double(tind);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(tind) || isnan(tind)
    % Formuojamas pranešimas.
    message = sprintf('tind laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
tind_apatine = 8;
tind_virsutine = 12;
if (tind < tind_apatine) || (tind > tind_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('tind laukelyje neleistinas ribos. Veskite nuo %0.2g iki %0.3g.',
tind_apatine, tind_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end

Ybx=get(handles.Ybx, 'string'); Ybx=str2double(Ybx);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(Ybx) || isnan(Ybx)
    % Formuojamas pranešimas.
    message = sprintf('Ybx laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
Ybx_apatine = 0;
Ybx_virsutine = 1;
if (Ybx < Ybx_apatine) || (Ybx > Ybx_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('Ybx laukelyje neleistinas ribos. Veskite nuo %0.2g iki %0.3g.',
Ybx_apatine, Ybx_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
```

```
Yox=get(handles.Yox, 'string'); Yox=str2double(Yox);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(Yox) || isnan(Yox)
    % Formuojamas pranešimas.
    message = sprintf('Yox laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
Yox_apatine = 0;
Yox_virsutine = 1.4;
if (Yox < Yox_apatine) || (Yox > Yox_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('Yox laukelyje neleistinas ribos. Veskite nuo %0.2g iki %0.3g.',
Yox_apatine, Yox_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end

om=get(handles.om, 'string'); om=str2double(om);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(om) || isnan(om)
    % Formuojamas pranešimas.
    message = sprintf('om laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
om_apatine = 0;
om_virsutine = 1;
if (om < om_apatine) || (om > om_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('om laukelyje neleistinas ribos. Veskite nuo %0.2g iki %0.3g.', om_apatine,
om_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end

Yas=get(handles.Yas, 'string'); Yas=str2double(Yas);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(Yas) || isnan(Yas)
    % Formuojamas pranešimas.
    message = sprintf('Yas laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
Yas_apatine = 0;
Yas_virsutine = 1.2;
if (Yas < Yas_apatine) || (Yas > Yas_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('Yas laukelyje neleistinas ribos. Veskite nuo %0.2g iki %0.3g.',
Yas_apatine, Yas_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end

Yo2x=get(handles.Yo2x, 'string'); Yo2x=str2double(Yo2x);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(Yo2x) || isnan(Yo2x)
    % Formuojamas pranešimas.
    message = sprintf('Yo2x laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
Yo2x_apatine = -0.2;
Yo2x_virsutine = 0.2;
if (Yo2x < Yo2x_apatine) || (Yo2x > Yo2x_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('Yo2x laukelyje neleistinas ribos. Veskite nuo %0.2g iki %0.3g.',
Yo2x_apatine, Yo2x_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
```

```
ke=get(handles.ke, 'string'); ke=str2double(ke);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(ke) || isnan(ke)
    % Formuojamas pranešimas.
    message = sprintf('ke laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
ke_apatine = -0.2;
ke_virsutine = 0.2;
if (ke < ke_apatine) || (ke > ke_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('ke laukelyje neleistinas ribos. Veskite nuo %0.2g iki %0.3g.', ke_apatine,
ke_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end

Sf=get(handles.Sf, 'string'); Sf=str2double(Sf);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(Sf) || isnan(Sf)
    % Formuojamas pranešimas.
    message = sprintf('Sf laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
Sf_apatine = 300;
Sf_virsutine = 600;
if (Sf < Sf_apatine) || (Sf > Sf_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('Sf laukelyje neleistinas ribos. Veskite nuo %0.3g iki %0.3g.', Sf_apatine,
Sf_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Parametras siunčiamas į Workspac'ą
assignin('base', 'Sf', Sf);

Fsmp=get(handles.Fsmp, 'string'); Fsmp=str2double(Fsmp);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(Fsmp) || isnan(Fsmp)
    % Formuojamas pranešimas.
    message = sprintf('Fsmp laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
Fsmp_apatine = 0;
Fsmp_virsutine = 600;
if (Fsmp < Fsmp_apatine) || (Fsmp > Fsmp_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('Fsmp laukelyje neleistinas ribos. Veskite nuo %0.2g iki %0.3g.',
Fsmp_apatine, Fsmp_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Parametras siunčiamas į Workspac'ą
assignin('base', 'Fsmp', Fsmp);

nuostatas=get(handles.nuostatas, 'string'); nuostatas=str2double(nuostatas);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(nuostatas) || isnan(nuostatas)
    % Formuojamas pranešimas.
    message = sprintf('Nuostato laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
nuostatas_apatine = 0.001;
nuostatas_virsutine = 10;
if (nuostatas < nuostatas_apatine) || (nuostatas > nuostatas_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
```

```
    message = sprintf('Nuostato laukelyje neleistinas ribos. Veskite nuo %0.1g iki %0.2g.',
nuostatas_apatine, nuostatas_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Parametras siunčiamas į Workspac'ą
assignin('base', 'nuostatas', nuostatas);

% Simulink'o realizacija
% Simuliacijos laiko aprašymas
Ton=get(handles.laikas_pradzia, 'string'); ton=str2double(Ton);
if isempty(ton) || isnan(ton)
    % Formuojamas pranešimas.
    message = sprintf('Simuliacijos pradžios laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
ton_apatine = 0;
ton_virsutine = 12;
if (ton < ton_apatine) || (ton > ton_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('Simuliacijos pradžios laukelyje neleistinas ribos. Veskite nuo %0.2g iki
%0.3g.', ton_apatine, ton_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end

Toff=get(handles.laikas_pabaiga, 'string'); toff=str2double(Toff);
if isempty(toff) || isnan(toff)
    % Formuojamas pranešimas.
    message = sprintf('Simuliacijos pabaigos laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
toff_apatine = 1;
toff_virsutine = 30;
if (toff < toff_apatine) || (toff > toff_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('Simuliacijos pabaigos laukelyje neleistinas ribos. Veskite nuo %0.2g iki
%0.3g.', toff_apatine, toff_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end

% parametrai surašomi į matricą, iš kurios vėliau gaunamos kintamųjų vertės
parametrai = [ si_kritine, Ks,   Ksi,    Kai;
               Yxa,        m,    Yas,    Ka;
               si_max1,    Kxx1, Yxs1,   si_max2;
               Kxx2,       Yxs2, tind,   0;
               ];
assignin('base', 'parametrai', parametrai);

% parametrai skirti pamaitinimo srautui apsirašyti
parametrai2 = [ Ybx, Yox,   om;
                Yo2x,ke,    0;
               ];
assignin('base', 'parametrai2', parametrai2);

% Pradedama simuliacija jeigu visi simboliai yra leistini
if stopas == 0
options=simset('DstWorkspace', 'current',...
               'SrcWorkspace', 'current',...
               'OutputPoints', 'specified',...
               'SaveFormat', 'Array');
% open('ecoli_modelis.slx');
sim('ecoli_modelis.slx', [ton, toff],options);

% Grafikų piešimas
% Biomasės
axes(handles.g_biomase);
g_biomase=plot(biomase(:,1), biomase(:,2), 'k'); grid on
title('Biomasės kitimas'); ylabel ('X, [g/l]'); xlabel ('Laikas, [h]');
% Gliukozės
axes(handles.g_gliukoze);
g_gliukoze=plot(gliukoze(:,1), gliukoze(:,2), 'k'); grid on
title('Ištirpusios gliukozės kitimas'); ylabel ('S, [g/l]'); xlabel ('Laikas, [h]');
```

```
% % Acetato
axes(handles.g_acetatas);
g_acetatas=plot(acetatas(:,1), acetatas(:,2), 'k'); grid on
title('Acetato kitimas'); ylabel ('A, [g/l]'); xlabel ('Laikas, [h]');
% % Tūris
axes(handles.g_turis);
g_turis=plot(turis(:,1), turis(:,2), 'k'); grid on
title('Tūrio kitimas'); ylabel ('V, [l]'); xlabel ('Laikas, [h]');
% Pamaitinimo srautas
axes(handles.g_pamaitinimas);
g_pamaitinimas=plot(pamaitinimas(:,1), pamaitinimas(:,2), 'k'); grid on
title('Pamaitinimo srauto kitimas'); ylabel ('V, [g/h]'); xlabel ('Laikas, [h]');
end

function feeding_Callback(hObject, eventdata, handles)
open('ecoli_modelis.slx');
open_system('ecoli_modelis/Pamaitinimo_srautas')

function grizti_Callback(hObject, eventdata, handles)
set(gcf,'Visible','off')
run('pradinis.m')


function uzdaryti_Callback(hObject, eventdata, handles)
close

function lygtys_Callback(hObject, eventdata, handles)
form = imread('formules.jpg');
figure(1);
imshow(form);

function nomenklatura_Callback(hObject, eventdata, handles)
norm = imread('nomenklatura.jpg');
figure(1);
imshow(norm);

function popupmenu1_Callback(hObject, eventdata, handles)
procesas = get(hObject,'value');
if procesas == 1
%     Jei pasirinktas rankiniu būdu užduodamas srautas tai įjungiamas
%     mygtukas leidžiantis keisti parametrus
    set(handles.feeding,'visible','on')
    set(handles.PI,'visible','off')
    set(handles.nuostatas,'visible','off')
    set(handles.t_nuostatas,'visible','off')
else
%     Jei pasirenkamas PI reguliatoriumi užduodamas srautas tai įjungiamas
%     mygtukas leidžiantis keisti reguliatoriaus parametrus
    set(handles.PI,'visible','on')
    set(handles.feeding,'visible','off')
    set(handles.nuostatas,'visible','on')
    set(handles.t_nuostatas,'visible','on')
end

function PI_Callback(hObject, eventdata, handles)
open('ecoli_modelis.slx');
open_system('ecoli_modelis/PI reguliatorius')

function nuostatas_Callback(hObject, eventdata, handles)
```

7.7.Priedas Nr. 7. Žinduolių ląstelių augimo greičių išraiškos

```
function [miu, miu_net, k_d, q_Glc, q_Gln, q_Lac, q_NH4]  = fcn(parametrai, Glc, Gln, Lac, NH4,
laikas)

% Reikšmių priskyrimas iš kintamųjų matricos
q_Glcmax = parametrai(1,1);
q_Glnmax = parametrai(1,2);
K_Glc = parametrai(1,3);
K_Gln = parametrai(1,4);
K_Lac = parametrai(2,1);
K_NH4 = parametrai(2,2);
k_dmax = parametrai(2,3);
Y_GlcX = parametrai(2,4);
m_AA = parametrai(3,1);
```

```
Y_GlnX = parametrai(3,2);
Y_LacGlc = parametrai(3,3);
Y_NH4X = parametrai(3,4);
Y_OGlc = parametrai(4,1);
K_o = parametrai(4,2);
Y_OGln = parametrai(4,3);
k1 = parametrai(4,4);
k2 = parametrai(5,1);
m_Gln = parametrai(5,2);
m_0 = parametrai(5,3);

% Gliukozės sunaudojimo greitis [mmol 10^9 cells^-1 h^-1]
q_Glc = q_Glcmax * (Glc/(Glc+K_Glc));
% Glutamino sunaudojimo greitis [mmol 10^9 cells^-1 h^-1]
q_Gln = q_Glnmax * (Gln/(Gln+K_Gln))+m_Gln;
% Ląstelės irimo greitis [h^-1]
k_d = k_dmax * ((Lac/(K_Lac+Lac))+(NH4/(K_NH4+NH4)));
% Santykinis biomasės augimo greitis [h^-1]
miu = ((q_Glc/(Y_GlcX*(Glc/(k1+Glc))))+((q_Gln-m_Gln)/Y_GlnX))*(1-exp(-laikas/10));
% Santykinis gryno kiekio augimo greitis [h^-1]
miu_net = miu - k_d;
% Laktato sunaudojimo greitis [mmol 10^9 cells^-1 h^-1]
q_Lac = Y_LacGlc * (Glc/(Glc+k2))*q_Glc;
% Amonio sulfato sunaudojimo greitis [mmol 10^9 cells^-1 h^-1]
q_NH4 = Y_NH4X*miu+m_AA;
```

7.8.Priedas Nr. 8. Vartotojo sąsajos kodas Žinduolių ląstelių modeliui

```
function varargout = zinduolius_sasaja(varargin)
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @zinduolius_sasaja_OpeningFcn, ...
                   'gui_OutputFcn',  @zinduolius_sasaja_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

function zinduolius_sasaja_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);

function varargout = zinduolius_sasaja_OutputFcn(hObject, eventdata, handles)

varargout{1} = handles.output;

function pradeti_Callback(hObject, eventdata, handles)
procesas = get(handles.popupmenu1,'value');
% aprašomas kintamasis kurio nusakomas srauto itėkjimo būdas
if procesas == 1
    reguliavimas = 0;
    assignin('base', 'reguliavimas', reguliavimas);
else
    reguliavimas = 1;
    assignin('base', 'reguliavimas', reguliavimas);
end

stopas = 0;
q_GLcmax=get(handles.q_GLcmax, 'string'); q_GLcmax=str2double(q_GLcmax);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(q_GLcmax) || isnan(q_GLcmax)
    % Formuojamas pranešimas.
    message = sprintf('q_GLcmax laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
```

```
% užsiduodamas apatinė ir žemutinė ribos
q_GLcmax_apatine = 0.01;
q_GLcmax_virsutine = 0.4;
if (q_GLcmax < q_GLcmax_apatine) || (q_GLcmax > q_GLcmax_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('q_GLcmax laukelyje neleistinas ribos. Veskite nuo %0.1g iki %0.2g.',
q_GLcmax_apatine, q_GLcmax_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end

q_GLnmax=get(handles.q_GLnmax, 'string'); q_GLnmax=str2double(q_GLnmax);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(q_GLnmax) || isnan(q_GLnmax)
    % Formuojamas pranešimas.
    message = sprintf('q_GLnmax laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
q_GLnmax_apatine = 0.01;
q_GLnmax_virsutine = 0.2;
if (q_GLnmax < q_GLnmax_apatine) || (q_GLnmax > q_GLnmax_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('q_GLnmax laukelyje neleistinas ribos. Veskite nuo %0.1g iki %0.2g.',
q_GLnmax_apatine, q_GLnmax_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end

K_Glc=get(handles.K_Glc, 'string'); K_Glc=str2double(K_Glc);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(K_Glc) || isnan(K_Glc)
    % Formuojamas pranešimas.
    message = sprintf('K_Glc laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
K_Glc_apatine = 1;
K_Glc_virsutine = 3.2;
if (K_Glc < K_Glc_apatine) || (K_Glc > K_Glc_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('K_Glc laukelyje neleistinas ribos. Veskite nuo %0.1g iki %0.2g.',
K_Glc_apatine, K_Glc_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end

K_Gln=get(handles.K_Gln, 'string'); K_Gln=str2double(K_Gln);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(K_Gln) || isnan(K_Gln)
    % Formuojamas pranešimas.
    message = sprintf('K_Gln laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
K_Gln_apatine = 0.1;
K_Gln_virsutine = 0.4;
if (K_Gln < K_Gln_apatine) || (K_Gln > K_Gln_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('K_Gln laukelyje neleistinas ribos. Veskite nuo %0.1g iki %0.2g.',
K_Gln_apatine, K_Gln_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end

K_Lac=get(handles.K_Lac, 'string'); K_Lac=str2double(K_Lac);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(K_Lac) || isnan(K_Lac)
    % Formuojamas pranešimas.
    message = sprintf('K_Lac laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
```

```
% užsiduodamas apatinė ir žemutinė ribos
K_Lac_apatine = 26;
K_Lac_virsutine = 104;
if (K_Lac < K_Lac_apatine) || (K_Lac > K_Lac_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('K_Lac laukelyje neleistinas ribos. Veskite nuo %0.2g iki %0.3g.',
K_Lac_apatine, K_Lac_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end

K_NH4=get(handles.K_NH4, 'string'); K_NH4=str2double(K_NH4);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(K_NH4) || isnan(K_NH4)
    % Formuojamas pranešimas.
    message = sprintf('K_NH4 laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
K_NH4_apatine = 2;
K_NH4_virsutine = 10;
if (K_NH4 < K_NH4_apatine) || (K_NH4 > K_NH4_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('K_NH4 laukelyje neleistinas ribos. Veskite nuo %0.1g iki %0.2g.',
K_NH4_apatine, K_NH4_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end

k_dmax=get(handles.k_dmax, 'string'); k_dmax=str2double(k_dmax);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(k_dmax) || isnan(k_dmax)
    % Formuojamas pranešimas.
    message = sprintf('k_dmax laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
k_dmax_apatine = 0.001;
k_dmax_virsutine = 0.1;
if (k_dmax < k_dmax_apatine) || (k_dmax > k_dmax_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('k_dmax laukelyje neleistinas ribos. Veskite nuo %0.1g iki %0.2g.',
k_dmax_apatine, k_dmax_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end

Y_GlcX=get(handles.Y_GlcX, 'string'); Y_GlcX=str2double(Y_GlcX);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(Y_GlcX) || isnan(Y_GlcX)
    % Formuojamas pranešimas.
    message = sprintf('Y_GlcX laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
Y_GlcX_apatine = 3.5;
Y_GlcX_virsutine = 14;
if (Y_GlcX < Y_GlcX_apatine) || (Y_GlcX > Y_GlcX_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('Y_GlcX laukelyje neleistinas ribos. Veskite nuo %0.1g iki %0.2g.',
Y_GlcX_apatine, Y_GlcX_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end

Y_GlnX=get(handles.Y_GlnX, 'string'); Y_GlnX=str2double(Y_GlnX);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(Y_GlnX) || isnan(Y_GlnX)
    % Formuojamas pranešimas.
    message = sprintf('Y_GlnX laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
```

```
% užsiduodamas apatinė ir žemutinė ribos
Y_GlnX_apatine = 0.5;
Y_GlnX_virsutine = 2;
if (Y_GlnX < Y_GlnX_apatine) || (Y_GlnX > Y_GlnX_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('Y_GlnX laukelyje neleistinas ribos. Veskite nuo %0.1g iki %0.2g.',
Y_GlnX_apatine, Y_GlnX_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end

Y_LacGlc=get(handles.Y_LacGlc, 'string'); Y_LacGlc=str2double(Y_LacGlc);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(Y_LacGlc) || isnan(Y_LacGlc)
    % Formuojamas pranešimas.
    message = sprintf('Y_LacGlc laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
Y_LacGlc_apatine = 0.8;
Y_LacGlc_virsutine = 3.2;
if (Y_LacGlc < Y_LacGlc_apatine) || (Y_LacGlc > Y_LacGlc_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('Y_LacGlc laukelyje neleistinas ribos. Veskite nuo %0.1g iki %0.2g.',
Y_LacGlc_apatine, Y_LacGlc_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end

Y_NH4X=get(handles.Y_NH4X, 'string'); Y_NH4X=str2double(Y_NH4X);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(Y_NH4X) || isnan(Y_NH4X)
    % Formuojamas pranešimas.
    message = sprintf('Y_NH4X laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
Y_NH4X_apatine = 0.25;
Y_NH4X_virsutine = 1;
if (Y_NH4X < Y_NH4X_apatine) || (Y_NH4X > Y_NH4X_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('Y_NH4X laukelyje neleistinas ribos. Veskite nuo %0.1g iki %0.2g.',
Y_NH4X_apatine, Y_NH4X_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end

Y_OGlc=get(handles.Y_OGlc, 'string'); Y_OGlc=str2double(Y_OGlc);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(Y_OGlc) || isnan(Y_OGlc)
    % Formuojamas pranešimas.
    message = sprintf('Y_OGlc laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
Y_OGlc_apatine = 1.8;
Y_OGlc_virsutine = 5.2;
if (Y_OGlc < Y_OGlc_apatine) || (Y_OGlc > Y_OGlc_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('Y_OGlc laukelyje neleistinas ribos. Veskite nuo %0.1g iki %0.2g.',
Y_OGlc_apatine, Y_OGlc_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end

Y_OGln=get(handles.Y_OGln, 'string'); Y_OGln=str2double(Y_OGln);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(Y_OGln) || isnan(Y_OGln)
    % Formuojamas pranešimas.
    message = sprintf('Y_OGln laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
```

```
% užsiduodamas apatinė ir žemutinė ribos
Y_OGln_apatine = 5;
Y_OGln_virsutine = 20;
if (Y_OGln < Y_OGln_apatine) || (Y_OGln > Y_OGln_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('Y_OGln laukelyje neleistinas ribos. Veskite nuo %0.1g iki %0.2g.',
Y_OGln_apatine, Y_OGln_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end

k1=get(handles.k1, 'string'); k1=str2double(k1);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(k1) || isnan(k1)
    % Formuojamas pranešimas.
    message = sprintf('k1 laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
k1_apatine = 0.16;
k1_virsutine = 0.64;
if (k1 < k1_apatine) || (k1 > k1_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('k1 laukelyje neleistinas ribos. Veskite nuo %0.1g iki %0.2g.', k1_apatine,
k1_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end

k2=get(handles.k2, 'string'); k2=str2double(k2);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(k2) || isnan(k2)
    % Formuojamas pranešimas.
    message = sprintf('k2 laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
k2_apatine = 1.5;
k2_virsutine = 6;
if (k2 < k2_apatine) || (k2 > k2_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('k2 laukelyje neleistinas ribos. Veskite nuo %0.1g iki %0.2g.', k2_apatine,
k2_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end

m_Gln=get(handles.m_Gln, 'string'); m_Gln=str2double(m_Gln);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(m_Gln) || isnan(m_Gln)
    % Formuojamas pranešimas.
    message = sprintf('m_Gln laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
m_Gln_apatine = 0;
m_Gln_virsutine = 0.1;
if (m_Gln < m_Gln_apatine) || (m_Gln > m_Gln_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('m_Gln laukelyje neleistinas ribos. Veskite nuo %0.1g iki %0.2g.',
m_Gln_apatine, m_Gln_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end

m_AA=get(handles.m_AA, 'string'); m_AA=str2double(m_AA);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(m_AA) || isnan(m_AA)
    % Formuojamas pranešimas.
    message = sprintf('m_AA laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
```

```
% užsiduodamas apatinė ir žemutinė ribos
m_AA_apatine = 0;
m_AA_virsutine = 0.1;
if (m_AA < m_AA_apatine) || (m_AA > m_AA_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('m_AA laukelyje neleistinas ribos. Veskite nuo %0.1g iki %0.2g.',
m_AA_apatine, m_AA_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end

K_o=get(handles.K_o, 'string'); K_o=str2double(K_o);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(K_o) || isnan(K_o)
    % Formuojamas pranešimas.
    message = sprintf('K_o laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
K_o_apatine = 0.5;
K_o_virsutine = 2;
if (K_o < K_o_apatine) || (K_o > K_o_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('K_o laukelyje neleistinas ribos. Veskite nuo %0.1g iki %0.2g.',
K_o_apatine, K_o_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end

m_0=get(handles.m_0, 'string'); m_0=str2double(m_0);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(m_0) || isnan(m_0)
    % Formuojamas pranešimas.
    message = sprintf('m_0 laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
m_0_apatine = 0;
m_0_virsutine = 0.1;
if (m_0 < m_0_apatine) || (m_0 > m_0_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('m_0 laukelyje neleistinas ribos. Veskite nuo %0.1g iki %0.2g.',
m_0_apatine, m_0_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end

Cglc=get(handles.Cglc, 'string'); Cglc=str2double(Cglc);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(Cglc) || isnan(Cglc)
    % Formuojamas pranešimas.
    message = sprintf('Cglc laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
Cglc_apatine = 50;
Cglc_virsutine = 100;
if (Cglc < Cglc_apatine) || (Cglc > Cglc_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('Cglc laukelyje neleistinas ribos. Veskite nuo %0.1g iki %0.2g.',
Cglc_apatine, Cglc_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Parametras siunčiamas į Workspac'ą
assignin('base', 'Cglc', Cglc);

Cgln=get(handles.Cgln, 'string'); Cgln=str2double(Cgln);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(Cgln) || isnan(Cgln)
    % Formuojamas pranešimas.
    message = sprintf('Cgln laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
```

```
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
Cgln_apatine = 5;
Cgln_virsutine = 40;
if (Cgln < Cgln_apatine) || (Cgln > Cgln_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('Cgln laukelyje neleistinas ribos. Veskite nuo %0.1g iki %0.2g.',
Cgln_apatine, Cgln_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Parametras siunčiamas į Workspac'ą
assignin('base', 'Cgln', Cgln);

nuostatas=get(handles.nuostatas, 'string'); nuostatas=str2double(nuostatas);
% Tikrinima ar nėra tuščias laukas ir neleistinų simbolių
if isempty(nuostatas) || isnan(nuostatas)
    % Formuojamas pranešimas.
    message = sprintf('Nuostato laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
nuostatas_apatine = 0.001;
nuostatas_virsutine = 0.1;
if (nuostatas < nuostatas_apatine) || (nuostatas > nuostatas_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('Nuostato laukelyje neleistinas ribos. Veskite nuo %0.1g iki %0.2g.',
nuostatas_apatine, nuostatas_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Parametras siunčiamas į Workspac'ą
assignin('base', 'nuostatas', nuostatas);

% Simulink'o realizacija
% Simuliacijos laiko aprašymas
Ton=get(handles.laikas_pradzia, 'string'); ton=str2double(Ton);
if isempty(ton) || isnan(ton)
    % Formuojamas pranešimas.
    message = sprintf('Simuliacijos pradžios laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
ton_apatine = 0;
ton_virsutine = 12;
if (ton < ton_apatine) || (ton > ton_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('Simuliacijos pradžios laukelyje neleistinas ribos. Veskite nuo %0.2g iki
%0.3g.', ton_apatine, ton_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end

Toff=get(handles.laikas_pabaiga, 'string'); toff=str2double(Toff);
if isempty(toff) || isnan(toff)
    % Formuojamas pranešimas.
    message = sprintf('Simuliacijos pabaigos laukelyje neleistinas simbolis. Veskite skaičių.');
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end
% Tikrinima ar įvesta reikšmė patenka į užduotas ribas
% užsiduodamas apatinė ir žemutinė ribos
toff_apatine = 1;
toff_virsutine = 300;
if (toff < toff_apatine) || (toff > toff_virsutine)
    % Formuojamas pranešimas jei sąlyga tenkinama.
    message = sprintf('Simuliacijos pabaigos laukelyje neleistinas ribos. Veskite nuo %0.2g iki
%0.3g.', toff_apatine, toff_virsutine);
    uiwait(warndlg(message, 'Klaida!'));
    stopas = 1;
end

% parametrai surašomi į matricą, iš kurios vėliau gaunamos kintamųjų vertės
parametrai = [ q_GLcmax, q_GLnmax,   K_Glc,    K_Gln;
               K_Lac,    K_NH4,      k_dmax,   Y_GlcX;
```

```
                m_AA,      Y_GlnX,      Y_LacGlc, Y_NH4X;
                Y_OGlc,    K_o,         Y_OGln,   k1;
                k2,        m_Gln          m_0,    0;
                ];
assignin('base', 'parametrai', parametrai);

% Pradedama simuliacija jeigu visi simboliai yra leistini
if stopas == 0
options=simset('DstWorkspace', 'current',...
                'SrcWorkspace', 'current',...
                'OutputPoints', 'specified',...
                'SaveFormat', 'Array');
% open('zinduoliu_modelis.slx');
sim('zinduoliu_modelis.slx', [ton, toff],options);

% Grafikų piešimas
% Bendro ląstelių skaičiaus kitimo grafikas
axes(handles.g_Xt);
g_Xt=plot(Xt(:,1), Xt(:,2), 'k'); grid on
title('Bendro ląstelių skaičiaus kitimas'); ylabel ('Xv, [10^9 cellsl L^-1]'); xlabel ('Laikas,
[h]');
% Gyvybingų lasteliu kitimo grafikas
axes(handles.g_Xv);
g_Xv=plot(Xv(:,1), Xv(:,2), 'k'); grid on
title('Gyvybingų ląstelių kitimas'); ylabel ('Xv, [10^9 cellsl L^-1]'); xlabel ('Laikas, [h]');
% Gliukozės
axes(handles.g_Glc);
g_Glc=plot(Glc(:,1), Glc(:,2), 'k'); grid on
title('Ištirpusios gliukozės kitimas'); ylabel ('Glc, [mM]'); xlabel ('Laikas, [h]');
% Glutamino kitimo grafikas
axes(handles.g_Gln);
g_Gln=plot(Gln(:,1), Gln(:,2), 'k'); grid on
title('Glutamino kitimas'); ylabel ('Gln, [mM]'); xlabel ('Laikas, [h]');
% Lakatato kitimo grafikas
axes(handles.g_Lac);
g_Lac=plot(Lac(:,1), Lac(:,2), 'k'); grid on
title('Laktato kitimas'); ylabel ('Lac, [mM]'); xlabel ('Laikas, [h]');
% NH4 kitimo grafikas
axes(handles.g_NH4);
g_NH4=plot(NH4(:,1), NH4(:,2), 'k'); grid on
title('NH4 kitimas'); ylabel ('NH4, [mM]'); xlabel ('Laikas, [h]');
% Tūris
axes(handles.g_W);
g_W=plot(W(:,1), W(:,2), 'k'); grid on
title('Tūrio kitimas'); ylabel ('V, [kg]'); xlabel ('Laikas, [h]');
% Pamaitinimo srautas
axes(handles.g_pamaitinimas);
g_pamaitinimas=plot(pamaitinimas(:,1), pamaitinimas(:,2), 'k'); grid on
title('Glutamino pamaitinimo srauto kitimas'); ylabel ('F, [g/h]'); xlabel ('Laikas, [h]');
end

function nomenklatura_Callback(hObject, eventdata, handles)
norm = imread('nomenklatura.jpg');
figure(1);
imshow(norm);

function lygtys_Callback(hObject, eventdata, handles)
form = imread('formules.jpg');
figure(1);
imshow(form);

function F_Glc_Callback(hObject, eventdata, handles)
open_system('zinduoliu_modelis/F_Glukozė')

function F_Gln_Callback(hObject, eventdata, handles)
open_system('zinduoliu_modelis/F_Glutaminas')

function F_base_Callback(hObject, eventdata, handles)
open_system('zinduoliu_modelis/F_bazė')

function grizti_Callback(hObject, eventdata, handles)
set(gcf,'Visible','off')
run('pradinis.m')

function uzdaryti_Callback(hObject, eventdata, handles)
close

function popupmenu1_Callback(hObject, eventdata, handles)
procesas = get(hObject,'value');
if procesas == 1
```

```
%     Jei pasirinktas rankiniu būdu užduodamas srautas tai įjungiamas
%     mygtukas leidžiantis keisti parametrus
    set(handles.F_Gln,'visible','on')
    set(handles.PI,'visible','off')
    set(handles.nuostatas,'visible','off')
    set(handles.t_nuostatas,'visible','off')
else
%     Jei pasirenkamas PI reguliatoriumi užduodamas srautas tai įjungiamas
%     mygtukas leidžiantis keisti reguliatoriaus parametrus
    set(handles.PI,'visible','on')
    set(handles.F_Gln,'visible','off')
    set(handles.nuostatas,'visible','on')
    set(handles.t_nuostatas,'visible','on')
end

function PI_Callback(hObject, eventdata, handles)
open('zinduoliu_modelis.slx');
open_system('zinduoliu_modelis/PI reguliatorius')
```