KAUNO TECHNOLOGIJOS UNIVERSITETAS

FUNDAMENTALIŲJŲ MOKSLŲ FAKULTETAS

MATEMATINĖS SISTEMOTYROS KATEDRA

Aurelija Sakalauskaitė

LĄSTELIŲ PLYŠINĖS JUNGTIES MODELIAVIMAS NAUDOJANT MARKOVO GRANDINES

Magistro darbas

Vadovas
profesorius habilituotas daktaras Henrikas Pranevičius

KAUNAS, 2011

KAUNO TECHNOLOGIJOS UNIVERSITETAS

FUNDAMENTALIŲJŲ MOKSLŲ FAKULTETAS

MATEMATINĖS SISTEMOTYROS KATEDRA

TVIRTINU
Katedros vedėjas
prof. habil.dr. V.Pekarskas
2011 06 02

LĄSTELIŲ PLYŠINĖS JUNGTIES MODELIAVIMAS NAUDOJANT MARKOVO GRANDINES

Taikomosios matematikos magistro baigiamasis darbas

Vadovas
Profesorius habilituotas daktaras
Henrikas Pranevičius
2011 06 01

Recenzentas
docentas Dalius Makackas
2011 06 01

Atliko
FMMM 9 grupės studentė
Aurelija Sakalauskaitė
2011 05 20

KAUNAS, 2011

KVALIFIKACINĖ KOMISIJA

Pirmininkas: Leonas Saulis, profesorius (VGTU)

Sekretorius: Eimutis Valakevičius, docentas (KTU)

Nariai: Algimantas Jonas Aksomaitis, profesorius (KTU)

Vytautas Janilionis, docentas (KTU)

Vidmantas Povilas Pekarskas, profesorius (KTU)

Rimantas Rudzkis, habil. dr., vyriausiasis analitikas (DnB NORD Bankas)

Zenonas Navickas, profesorius (KTU)

Arūnas Barauskas, dr., vice-prezidentas projektams (UAB „Baltic Amadeus“)

Sakalauskaitė A. Ląstelių plyšinės jungties modeliavimas naudojant Markovo grandines: Taikomosios matematikos magistro baigiamasis darbas / mokslinis vadovas profesorius H. Pranevičius; Kauno technologijos universitetas, Fundamentaliųjų mokslų fakultetas, Matematinės sistemotyros katedra. – Kaunas, 2011. – 85 p.

SANTRAUKA

Šiame darbe pateikiama ląstelių plyšinės jungties Markovo modelių sudarymo metodika, apimanti perėjimo tikimybių skaičiavimą panaudojant nepriklausomų J. Bernulio bandymų schemą, stacionariųjų tikimybių skaičiavimą ir plyšinės jungties laidumo priklausomybės nuo įtampos skaičiavimus.

Tariama, kad plyšinė jungtis sudaryta iš daugybės lygiagrečiai sujungtų kanalų (pvz., 1000). Kiekvienas kanalas sudarytas iš 2 nuosekliai sujungtų puskanalių (koneksonų), o kiekvienas koneksonas sudarytas iš 6 lygiagrečiai sujungtų vienetų (koneksinų). Kiekvienas koneksinas gali būti atviroje arba uždaroje būsenoje, kuri priklauso nuo kanalo įtampos.

Modelių, sukurtų naudojant šią metodiką, adekvatumas patikrintas lyginant plyšinės jungties modeliavimo rezultatus su imitacinio modeliavimo (programų, kurias atliko Nerijus Paulauskas ir Saulius Vaičeliūnas (KTU Informatikos fakulteto magistrantai)) rezultatais, kurie patikrinti su eksperimentų rezultatais.

Sukurta Markovo modelių metodika panaudota kuriant plyšinės jungties modelius, kai koneksinai aprašomi 3 būsenomis: uždara, atvira ir visiškai uždara.

Sakalauskaitė A. Modelling of the Gap Junction of Cells Using Markov Chains: Master‘s work in applied mathematics / supervisor habil. dr. prof. H. Pranevičius; Department of Mathematical Research in Systems, Faculty of Fundamental Sciences, Kaunas University of Technology. – Kaunas, 2011. – 85 p.

SUMMARY

In this paper the methodology of composing of Markov models of the gap junction of cells is introduced. This methodology contains of computing of transition probabilities using scheme of independent J. Bernoulli trials, computing of stationary probabilities and computing of the conductance of the gap junction dependence on a voltage.

It is considered that the gap junction consists of a lot of channels (for example, 1000), joined parallel with each other. Each channel consists of two subchannels (connexons), joined in series, and each connexon consists of 6 units (connexins), joined parallel with each other. Each connexin can be in an open or a closed state. State of a connexin depends on a voltage that is going through the channel.

The adequacy of models that were created using this methodology is tested comparing the results of modelling of the gap junction using Markov chains with the results of the imitational modelling (programs that were done by Nerijus Paulauskas and Saulius Vaičeliūnas (postgraduate students from Informatics faculty, KTU)). The latter results were tested with the results of experiments.

In this paper the methodology of created Markov models was used creating the models of gap junction, where a connexin is described being in 3 states: closed, open and deep closed.

TURINYS

LENTELIŲ SĄRAŠAS

PAVEIKSLŲ SĄRAŠAS

ĮVADAS

Daugelyje stuburinių audinių jonų ir mažų molekulių pasikeitimą tarp gretimų ląstelių valdo plyšinės jungtys (sinapsės), tokiu būdu koordinuojančios ląstelių veiklą. Sinapsės tiekia išteklius (jonus, mažas molekules), kad valdytų ląstelių veiklą audiniuose. Dėl to sinapsės dalyvauja daugelyje biologinių procesų, tokių kaip:

- vystymasis;
- augimas;
- sekrecija (išskyrimas);
- impulsyvus dauginimasis.

Struktūriniu požiūriu sinapsės yra tarpląstelinių kanalų (plyšinių jungčių kanalų) agregatai. Dvejopo įtampos-varžos metodo taikymas ląstelių porų mėginiuose leidžia išsiaiškinti elektrines plyšinių jungčių bei jų kanalų savybes. Laidūs ir kinetiniai duomenys, gauti daugiakanaliame ir vienkanaliame lygmenyje, nuvedė prie apibendrintos plyšinių jungčių kanalų veikimo koncepcijos.

Šiame darbe tiriamas tarpląstelinių plyšinių jungčių veikimas. Sudaryti 6 skirtingi modeliai: 4 diskretaus laiko Markovo grandinės (2 būsenų 6 koneksinų, 12 koneksinų ir 3 būsenų 6 koneksinų, 12 koneksinų), 2 tolydaus laiko Markovo grandinės (2 būsenų 6 koneksinų, 12 koneksinų).

Tikslas – sudaryti Markovo grandinių modelius (paminėtus aukščiau) ląstelių plyšinei jungčiai ir ištirti Markovo grandinių adekvatumą (palyginti gautus rezultatus su imitacinių modelių rezultatais, gautais Nerijaus Paulausko ir Sauliaus Vaičeliūno).

Anksčiau buvo atliekami tyrimai su dvejais vartais (kai kanalas sudarytas tik iš dviejų puskanalių, t. y. vartų), su keturiais vartais (kai du vartai užsidaro greitai, o kiti du - lėtai) bei su šešiais vartais (kai kanalo vienas puskanalis sudarytas iš 6 koneksinų (baltymų)). Nerijus Paulauskas (buvęs KTU Informatikos fakulteto magistrantas) ir Saulius Vaičeliūnas (KTU Informatikos fakulteto magistrantas) sukūrė imitacinio modeliavimo programas, kurios parodo laidžio, įtampos bei srovės stiprio priklausomybes nuo laiko t, kai yra kanalas sudarytas iš dviejų, keturių bei šešių vartų. Taip pat programoje atlikta parametrų optimizacija lyginant modeliavimo duomenis su atliktų Niujorko Yeshiva universiteto Alberto Einšteino medicinos kolegijoje (Albert Einstein College of Medicine of Yeshiva University, New York, U.S.A) eksperimentų rezultatais. N. Paulauskas ir S. Vaičeliūnas bendradarbiavo su Niujorko Einšteino kolegijos profesoriumi Feliksu Bukausku, medicinos daktaru Mindaugu Pranevičiumi bei su KTU Informatikos fakulteto Verslo informatikos katedros profesoriumi Henriku Pranevičiumi.

Padedant prof. H. Pranevičiui ir prof. F. Bukauskui šiame darbe nagrinėjami tikimybiniai modeliai, kurių gautais rezultatais galima patikrinti imitacinių modelių adekvatumą.

1. ANALITINĖ DALIS

1.1. TARPLĄSTELINĖS PLYŠINĖS JUNGTIES KANALO (PJK) MODELIAI

Vienas pirmųjų modelių, kuriuo remiasi tolimesni modeliai, yra Rolfo Vogelio ir Roberto Veingarto modelis (Rolf Vogel; Robert Weingart: 2002: 1)

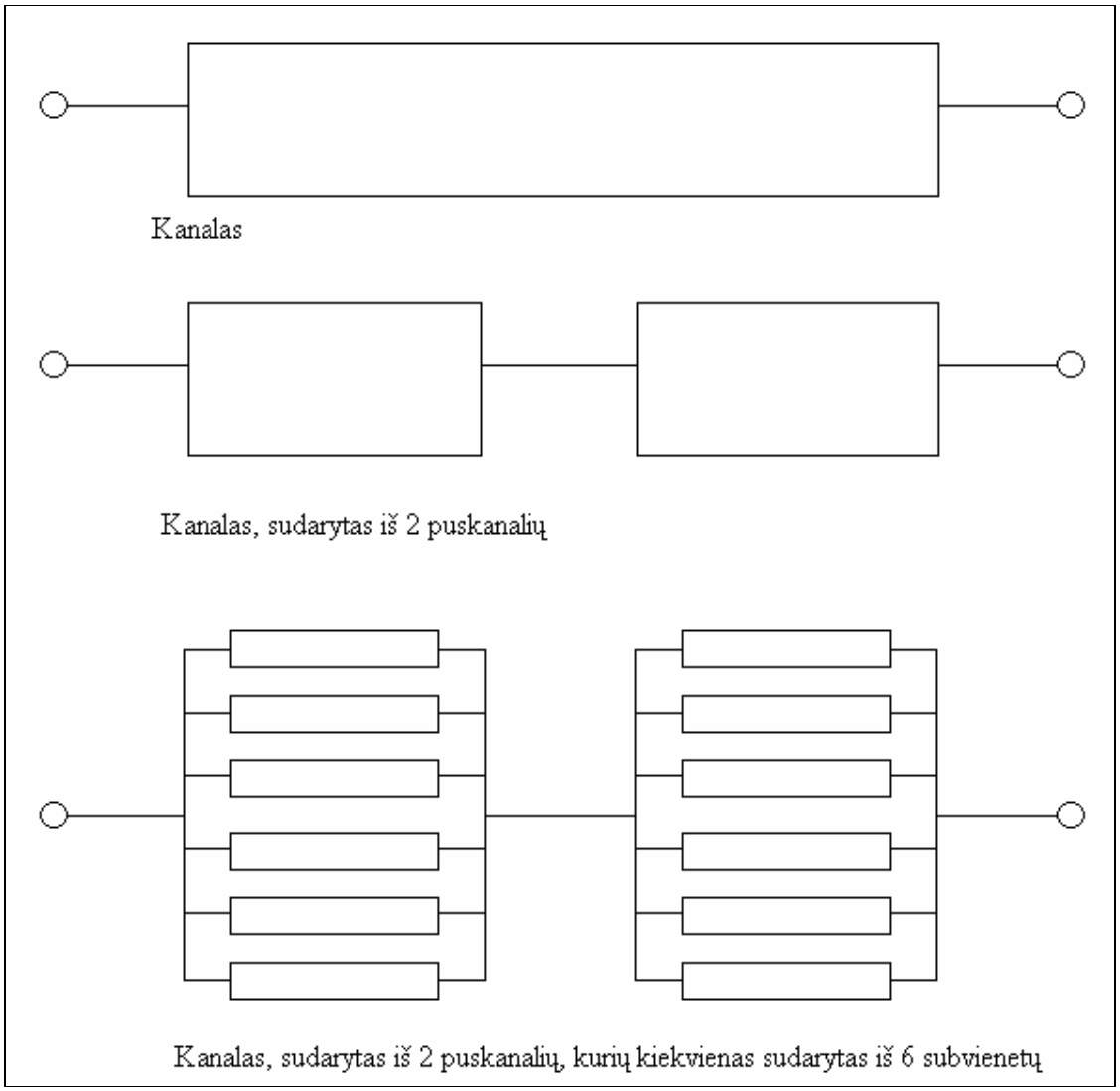
Šis modelis aprašo plyšinių jungčių, jų kanalų ir jų matematinio modeliavimo elektrofiziologiją.

Daugelyje stuburinių audinių jonų ir mažų molekulių pasikeitimą tarp gretimų ląstelių valdo plyšinės jungtys (sinapsės), tokiu būdu koordinuojančios ląstelių veiklą. Dvejopo įtampos-varžos metodo taikymas ląstelių porų mėginiuose leidžia išsiaiškinti elektrines plyšinių jungčių bei jų kanalų savybes. Laidūs ir kinetiniai duomenys, gauti daugiakanaliame ir vienkanaliame lygmenyje, nuvedė prie apibendrintos plyšinių jungčių kanalų veikimo koncepcijos. Remiantis biologiniais duomenimis, gautais šiuo būdu, buvo sukurtas matematinis modelis. Šis modelis yra įvairiapusiškas ir leidžia atkurti skirtingų rūšių stuburinių plyšinių jungčių elektrofiziologinį elgesį.

Sinapsės tiekia išteklius (jonus, mažas molekules), kad valdytų ląstelių veiklą audiniuose. Ši savybė buvo panaudota skirtingų filogeninių rūšių metazojų skirtingose ontogenetinėse stadijose. Dėl to sinapsės dalyvauja daugelyje biologinių procesų, tokių kaip:

- vystymasis;
- augimas;
- sekrecija (išskyrimas);
- impulsyvus dauginimasis.

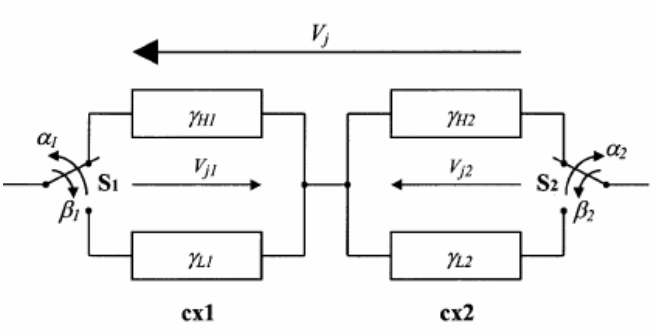
Struktūriniu požiūriu sinapsės yra tarpląstelinių kanalų (plyšinių jungčių kanalų) agregatai. Kiekvienas kanalas sudarytas iš 2 puskanalių (koneksonų), iš kurių kiekvienas sudarytas iš 6 tarpmembraninių proteinų (koneksinų/subvienetų), kurie suformuoja vandeninę porą (žr. 1.1 pav.).



1.1 pav. Plyšinės jungties kanalo koncepcija

Apibendrintas matematinis modelis (Rolf Vogel; Robert Weingart: 1998: 2) buvo pateiktas elektrofizinėms kanalų savybėms išsiaiškinti.

Šis modelis apima tik kanalą, sudarytą iš 2 puskanalių (žr. 1.2 pav.).



1.2 pav. Kanalas, sudarytas iš 2 nuosekliai sujungtų vartų

Kanalas susideda iš 2 nuosekliai sujungtų puskanalių, kurių kiekvienas išskiria 2 būsenas – 2 laidumus: aukštos (H – high) būsenos ir žemos (L – low) būsenos laidžius. Būsena H – tai būsena, kai puskanalis yra visiškai atidarytas, o L – būsena, kai puskanalis dalinai uždarytas. Tų pačių būsenų laidžiai tokie: γ_H, γ_L. Pagrindinė kanalo būsena atitinka 2 visiškai atidarytus puskanalius, t. y. abu puskanaliai yra aukštoje (H – high) būsenoje, kuri atitinka būseną HH. Kanalas veikia pasyvioje būsenoje, kai vienas iš puskanalių yra žemoje (L – low) būsenoje. Priklausomai nuo perėjimo įtampos V_j poliškumo vieno kanalo būsena atitinka būseną LH (Low-High) arba HL (High-Low). Ligi šiol šios būsenos nebuvo atskirtos, t. y. jos buvo laikomos viena būsena. Paaiškėjo, kad tai buvo trūkumas, ypač homomerinių – heterotipinių kanalų, susidedančių iš puskanalių su skirtingomis elektrinėmis savybėmis (kai laidis $\gamma_{j,pasyvioje\_busenoje}$ tampa asimetrinis), atveju.

Kiekvieno puskanalio nuo įtampos priklausančių vartų sąveikavimas, t. y. perėjimas tarp aukštos ir žemos būsenų, sumodeliuotas jungiklių S_1 ir S_2. Kiekvieno jungiklio elgsena aprašoma parametrais α ir β, kurie yra atitinkamai L ir H būsenų gyvavimo trukmės. Buvo pasiūlyta, kad jungikliai veikia nepriklausomai vienas nuo kito ir tarta, kad vartų sąveikavimas – būdinga puskanalių savybė, t. y. kiekvienas puskanalis turi daviklį, kuris aptinka vietinį elektrinį lauką, kuris priklauso nuo įtampos kritimo ant puskanalio. α_1, β_1 - V_{j1} (pirmo puskanalio) funkcijos; α_2, β_2 - V_{j2} (antro puskanalio) funkcijos. Jungiklių nepriklausomumas suvaržytas tuo, kad V_{j1} ir V_{j2} priklauso nuo V_j ir nuo puskanalio laidumų.

Kad galima būtų toliau aptarti laidžias ir kinetines plyšinių jungčių charakteristikas, svarbu išsiaiškinti kanalo struktūros ir funkcijos ryšį. Struktūriškai yra 2 kanalų klasės, kurias reiktų aptarti: homomeriniai – homotipiniai ir homomeriniai – heterotipiniai kanalai. Kiekvienas puskanalis išskiria 2 laidžias būsenas, vedančias prie 4 kanalo būsenų (iš viso 8 atvejai, nes kiekvienas puskanalis gali turėti tik vieną būseną (4x2=8) (2 puskanaliai ir 4 būsenos, tai iš viso gali būti 8 būdai)). Dėl matematinio modeliavimo 8 atvejai gali būti sumažinti net iki 2 skirtingų funkcinių būsenų, pavadintų homotipine ir heterotipine funkcijomis.

Homotipinė funkcija (simetrinė funkcija) tinkama, kai abu puskanaliai gauna vienodas laidžių savybes, kurios galioja homotipinio vieno kanalo būsenoms HH ir LL. Suminis laidis apskaičiuojamas tokia formule (1.1):

$$\gamma_{\text{suminis}} = \frac{\Gamma_0}{\exp\left(\frac{-V_j}{V_0\left(1+\exp\left(\frac{V_j}{V_0}\right)\right)}\right) + \exp\left(\frac{V_j}{V_0\left(1+\exp\left(\frac{-V_j}{V_0}\right)\right)}\right)}; \quad \textbf{(1.1)}$$

čia:

$$\Gamma_0 = \begin{cases} \Gamma_H, kai \quad pagrindinė \quad (HH) \quad būsena; \\ \Gamma_L, kai \quad pasyvioji \quad (LL) \quad būsena; \end{cases}$$

$$V_0 = \begin{cases} V_H, kai \quad pagrindinė \quad (HH) \quad būsena; \\ V_L, kai \quad pasyvioji \quad (LL) \quad būsena. \end{cases}$$

Tai varpo formos kreivė, simetrinė ordinatės atžvilgiu.

Heterotipinė funkcija atitinka suminį laidį 2 puskanalių su skirtingomis laidžių funkcijomis, t. y. Γ_{01}, V_{01} ir Γ_{02}, V_{02}. Ši funkcija apibūdina homotipinio kanalo būsenas LH ir HL ir apskritai visas 4 būsenas heterotipinio kanalo. Kadangi nėra jokio analizinio sprendimo apskaičiuoti suminį laidį, tai šis laidis skaičiuojamas taikant skaitinius metodus (Rolf Vogel; Robert Weingart: 1998: 2). Heterotipinė funkcija yra asimetrinė ordinatės atžvilgiu.

Homomeriniai puskanaliai – puskanaliai, sudaryti iš vieno tipo koneksinų.

Heteromeriniai puskanaliai – puskanaliai, sudaryti iš daugiau negu vieno tipo koneksinų.

Homotipiniai plyšinės jungties kanalai – kanalai, sudaryti iš 2 vienodų puskanalių.

Heterotipiniai plyšinės jungties kanalai – kanalai, sudaryti iš 2 skirtingų puskanalių.

Taigi, plyšinės jungties kanalai gali būti tokie:

- homomeriniai – homotipiniai;
- heteromeriniai – homotipiniai;
- homomeriniai – heterotipiniai;
- heteromeriniai – heterotipiniai.

Ye Chen-Izu, Alonso P. Moreno ir Robert A. Spangler modelis (2001: 3)

Plyšinės jungtys yra tarpląsteliniai kanalai, kurie jungia gretimų ląstelių citoplazmas. Kadangi plyšinės jungties kanalas sudarytas iš dviejų puskanalių (koneksonų), besijungiančių tiesiogiai vienas su kitu, tai įtampai jautrus kanalas yra simetriškas homotipiniams kanalams, sudarytiems iš dviejų

identiškų koneksonų, ir asimetriškas heterotipiniams kanalams, sudarytiems iš dviejų skirtingų koneksonų (t. y. skirtingų koneksinų kompozicijos).

Šiame straipsnyje autoriai aprašo vartų modelį, kuris leidžia kiekybiškai aprašyti homotipinių ir heterotipinių kanalų jautrumą įtampai. Šį vartų modelį dabar galima pritaikyti visai įtampos sričiai, pašalinant duomenų sujungimo ir dviejų puskanalių sintezės apibūdinimus, kurie yra probleminiai susiduriant su heterotipiniais kanalais. Šis modelis taip pat pateikia praktinę formulę perteikti kiekybiškas kelias anksčiau kokybiškas idėjas įtraukiant panašumo principą taikant vartų įtampą priimančiam puskanaliui, vartų poliškumo priskyrimą puskanaliui bei sujungimo sąveiką tarp 2 narių-puskanalių nesugadintame plyšinės jungties kanale.

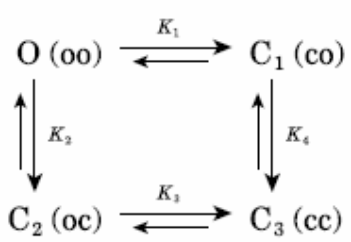
Autoriai pristato paprastą 4 būsenų modelį sujungti 2 narius-puskanalius nesugadintame plyšinės jungties kanale. Šis modelis taiko 3 naujas prielaidas:

1) termodinaminį susiderinimą Gibbso laisvos energijos sistemoje;

2) vieną atviro kanalo laidžio ir vieną uždarojo (pasyviojo) kanalo laidžio reikšmę nesugadintam (sveikam) kanalui;

3) supaprastinimą, pateikiamą priimant nepriklausomą arba priklausomą vartų sąveikavimą (pirmas šią prielaidą iškėlė Spray ir kiti; šitame straipsnyje autoriai šią prielaidą įgyvendina nauja matematine išraiška).

Priešpriešių vartų modelis plyšinės jungties įtampos kitimams. Įtampa krenta ant kiekvieno puskanalio skirtinga, nes puskanaliai sujungti nuosekliai (remiantis nuosekliuoju grandinės jungimu, $U = U_1 + U_2$ (U – visa įtampa, krentanti ant kanalo; U_1 ir U_2 yra atitinkamai pirmojo ir antrojo puskanalių įtampos)). Modelio schema yra tokia (žr. 1.3 pav.):



1.3 pav. Modelio schema

Ši schema rodo, kokia naudojama pusiausvyros konstanta K_i ($i = \overline{1,4}$) keičiantis kanalo būsenoms. Taip pat rodoma, kad kanalas gali būti vienoje iš 4 būsenų:

1) O (oo) – būsena, kai abu vartai (tiek kairysis, tiek dešinysis puskanalis) yra atviri (o – „open“);
2) C_1 (co) – būsena, kai kairysis puskanalis yra uždaras (c – „closed“), o dešinysis – atviras (o – „open“);

3) C_2 (oc) – būsena, kai kairysis puskanalis yra atviras, o dešinysis – uždaras;

4) C_3 (cc) – būsena, kai abu puskanaliai uždaryti.

Pusiausvyros konstantos išreiškiamos Bolcmano sąryšiu:

$$K_i = e^{A_i \cdot (-V - V_{oi})}; \tag{1.2}$$

čia: A_i ($i = \overline{1,4}$) – įtampos jautrio koeficientas; V_{oi} ($i = \overline{1,4}$) – įtampa pusiau maksimaliam laidžiui.

Tikimybė P_o, kad plyšinės jungties kanalas bus atviroj būsenoj, yra apskaičiuojama panaudojant pusiausvyros konstantas (žr. (1.3) formulę).

$$P_o = \frac{O}{O + C_1 + C_2 + C_3} = \frac{1}{1 + K_1 + K_2 + K_1 \cdot K_4}; \tag{1.3}$$

čia: remiantis laisvos energijos keitimusi tarp dviejų būsenų pastovumu, $K_1 \cdot K_4 = K_2 \cdot K_3$. Tikimybė gali būti išreikšta ir laidžiais (žr. (1.4) formulę).

$$P_o = \frac{g_n - g_{res}}{g_{\max} - g_{res}}; \tag{1.4}$$

čia: $g_n = \frac{G_{ss}}{G_i}$ - normalizuotas pastovios būsenos (ss – „steady state“) laidis; G_{ss} – pastovios būsenos laidis; G_i – pradinis/momentinis laidis; $g_{\max} = \frac{G_{\max}}{G_i}$ – normalizuotas maksimalus laidis ($g_{\max} \geq 1$); $g_{res} = \frac{G_{\min}}{G_i}$ – normalizuotas liekamasis (res – „residual“) (uždaros būsenos) laidis ($0 \leq g_{res} \leq 1$).

Iš formulės (1.4) galima išreikšti g_n. Tada gausime (žr. (1.5) formulę)

$$g_n = g_{res} + (g_{\max} - g_{res}) \cdot P_o; \tag{1.5}$$

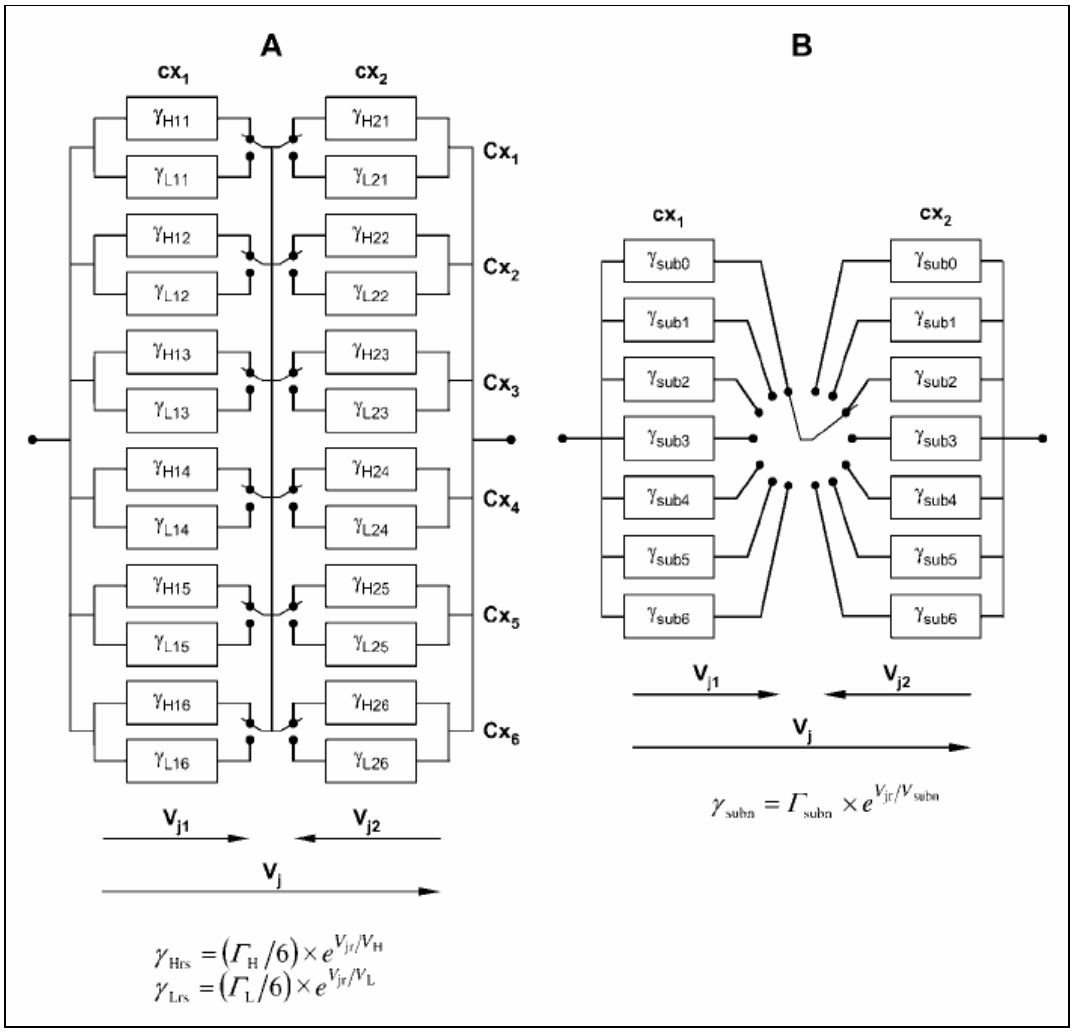
Šia ir (1.3) formulėmis remiasi 2 modeliai, kurie yra suskaidyti iš šio modelio. Tai priklausomas vartų sąveikavimo modelis, kai teigiama, kad kai vienas puskanalis uždaras, tai kitas turi būti atidarytas ir visa įtampa krinta ant uždaro puskanalio, ir nepriklausomas vartų sąveikavimo modelis, kai teigiama, kad abu puskanaliai (vartai) veikia nepriklausomai vienas nuo kito ir įtampa krinta ant abiejų vartų.

Rolf Vogel, Virginijaus Valiūno, Robert Weingart modelis (2006: 4)

Žmogaus HeLa[1] ląsteles atitinkančios pelės ląstelės koneksinas30 buvo panaudotos tirti plyšinės jungties kanalo subbūsenų elektrines savybes. Eksperimentai buvo atlikti su ląstelių poromis naudojant dvigubą įtampos-krūvio kaupimo metodą. Vieno kanalo srovės atskleidė diskrečius lygius, priskirtinus pagrindinei būsenai, uždarajai būsenai ir tarp šių būsenų įsiterpiančioms 5 subbūsenoms, rodančius veikimą 6 subvartų, tiekiamų 6 plyšinės jungties puskanalio koneksinų. Subbūsenų laidžiai $\gamma_{j,subbusena}$ ($j = \overline{1,6}$) buvo nelygiai pasiskirstę tarp pagrindinės būsenos ir pasyvios (uždaros) būsenos laidžių ($\gamma_{j,pagrindine\_busena} = 141\,\text{pS}$, $\gamma_{j,uzdara\_busena} = 21\,\text{pS}$). Pirmojo subvarto sužadinimas kanalo laidumą sumažino $\approx 30\%$, o kitų subvartų sužadinimas sumažino laidį kiekviename subvarte $10-15\%$. Srovės perėjimai tarp būsenų yra greiti (< 2 ms). Subbūsenų įvykiai buvo atskirti nuo perėjimų iš pagrindinės būsenos į pagrindinę būseną; perėjimai tarp subbūsenų buvo reti. Dėl to subvartai komplektuojami vienu metu, o ne iš eilės. Subbūsenų įvykių skaičius buvo didesnis prie didesnės įtampos reikšmės. Subbūsenų įvykių dažnis ir trukmė padidėjo didėjant sinchroniškai sužadintų subvartų skaičių. Matematinis modelis, aprašantis plyšinės jungties kanalų veikimą, buvo išplėstas įtraukiant kanalo subbūsenas. Remiantis laidžių $\gamma_{j,pagrindine\_busena}$ ir $\gamma_{j,uzdara\_busena}$ jautrumu įtampai, imitacinis modelis parodė, kad kiekvienos subbūsenos laidis taip pat priklauso nuo įtampos: $\gamma_{j,subbusena} = f(V_j)$.

Apibendrintas kanalo modelis naudojo įtampai jautrų vartų kitimą pereinant iš pagrindinės būsenos į uždarą. Priklausomai nuo perėjimo įtampos poliškumo, vienas iš 2 puskanalių persijungia akimirksniu iš aukšto laidžio į žemo laidžio būseną ir atv., kai kitas puskanalis sustoja aukšto laidžio būsenoje. Ši elgsena buvo paimta modeliuoti vieno kanalo subbūsenas, t. y. kiekviena vieno kanalo subbūsena vaizduoja vieno puskanalio subvartų sužadinimą arba užsidarymą, kai kito puskanalio subvartai yra nesužadinti arba atviri. Sužadintų subvartų skaičius vaizduoja vieno kanalo subbūsenų skaičių. Vieno kanalo pagrindinė būsena žymima sub0 (sub – „subbstates“ – „subbūsenos“), o uždara būsena – sub6 (žr. 1.4 pav.).

[1] HeLa ląstelės – tai gimdos kaklelio vėžinės ląstelės kultūra, gauta iš vienos vėžinės ląstelės, paimtos iš Henrietta Lacks, kuri mirė nuo vėžio 1951-10-04.



1.4 pav. Vieno kanalo ir jo subbūsenų elektrinės schemos

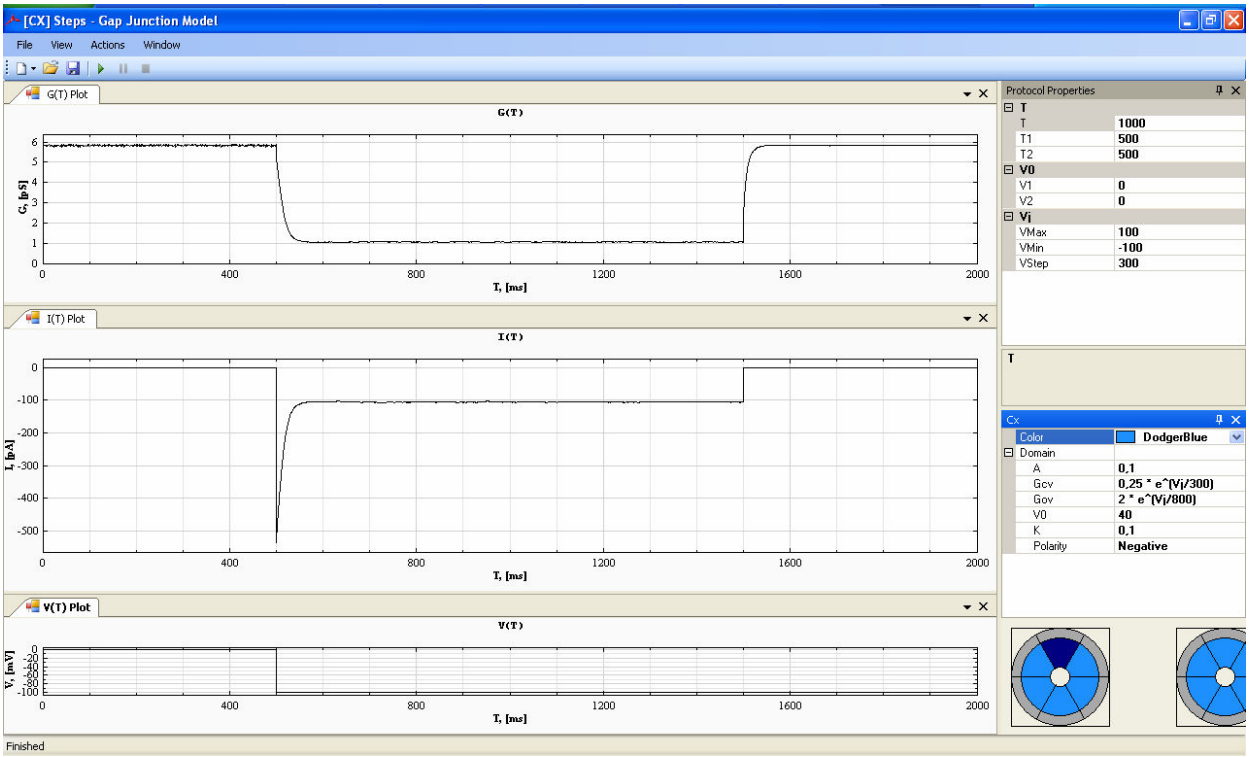
Šiame paveiksle vaizduojami du atvejai: A atveju teigiama, kad koneksinai Cx veikia nepriklausomai vienas nuo kito. γ_{Hrs} ir γ_{Lrs} - aukšto (H – „high“) ir žemo (L – „low“) laidžio koneksino s ($s = \overline{1,6}$) būsenos puskanalio r ($r = \overline{1,2}$), valdomo įtampos V_{jr}. Kanalo pagrindinė būsena sukonstruota visus jungiklius (pažymėti paveiksle lankeliais) įjungiant, o uždara būsena – išjungiant arba 6 kairiuosius, arba 6 dešiniuosius jungiklius. B atveju teigiama, kad koneksinai priklausomi vienas nuo kito. Vieno kanalo subbūsenos sumodeliuotos 7 puskanalio subbūsenomis. Kiekvieną subbūseną (sub0-sub6) atitinkantys parametrai turi būti suskaičiuoti atskirai. Pagrindinė kanalo būsena sumodeliuota iš puskanalių cx_1 ir cx_2 laidžių γ_{sub0}, o uždara būsena – iš puskanalio cx_1 laidžio γ_{sub0} ir iš puskanalio cx_2 laidžio γ_{sub6}. Paveikslėlyje parodytų formulių žymenys: Γ_H ir Γ_L – pastovaus aukšto (H) ir žemo (L) laidžio reikšmės; Γ_{subn} – puskanalio sustojimo n-tojoje subbūsenoje laidžio konstanta; V_H ir V_L – puskanalio pastovios aukštos ir žemos įtampos reikšmės (dažniai); V_{subn} – puskanalio sustojimo n-tojoje subbūsenoje įtampos konstanta.

1.2. PJK MODELIUI TAIKOMŲ MATEMATINIŲ METODŲ IR PROGRAMINIŲ PRIEMONIŲ APŽVALGA

Naudojamas dvigubas įtampos-krūvio kaupimo metodas. Buvo atliekami elektrofiziologiniai tyrimai su plyšinėmis jungtimis, paimtomis iš Xenopus (vandens varlės) oocitų. Šitas metodas skirtas įvertinti plyšinės jungties ląstelių poravimąsi. Ląstelės pora (kairioji ir dešinioji ląstelė) buvo skirtingai įkrauta. Šios poros potencialas buvo lygus -60 mV, artimas ląstelės pastoviam potencialui. Įtampos žingsniai buvo perduodami dešiniajai ląstelei, kai įtampa buvo nekeičiama kairiojoje ląstelėje. Sužadinta srovė kairiojoje ląstelėje tada tapo perėjimo srove.

Duomenų analizei naudota Microsoft Office Excel 2003 programa. Buvo nustatyta, kad geriausiai srovės gesimą dėl vartų priklausymo nuo įtampos aprašo eksponentinės funkcijos. Pradinė srovė gaunama laiko momentu, lygiu nuliui. Pastovios būsenos srovė gauta artėjant eksponentinei funkcijai į begalybę. Įtampa V_j gaunama kaip potencialų skirtumas tarp 2 įtampos elektrodų. Pradinis laidis G_i ir pastovios būsenos laidis G_{ss} suskaičiuoti atitinkamai iš pradinės ir pastovios būsenos srovių (Ye Chen-Izu; Alonso P. Moreno; Robert A. Spangler: 2001: 3).

Nerijus Paulauskas sukūrė imitacinio modeliavimo programą su C# (C Sharp). Programa braižo dviejų vartų, keturių vartų bei šešių vartų modelių grafikus (įtampos, laidžio ir srovės stiprio priklausomybes nuo laiko, laidžio priklausomybę nuo įtampos), atlieka modelio parametrų optimizavimą. Programos langas pavaizduotas 1.5 paveiksle.



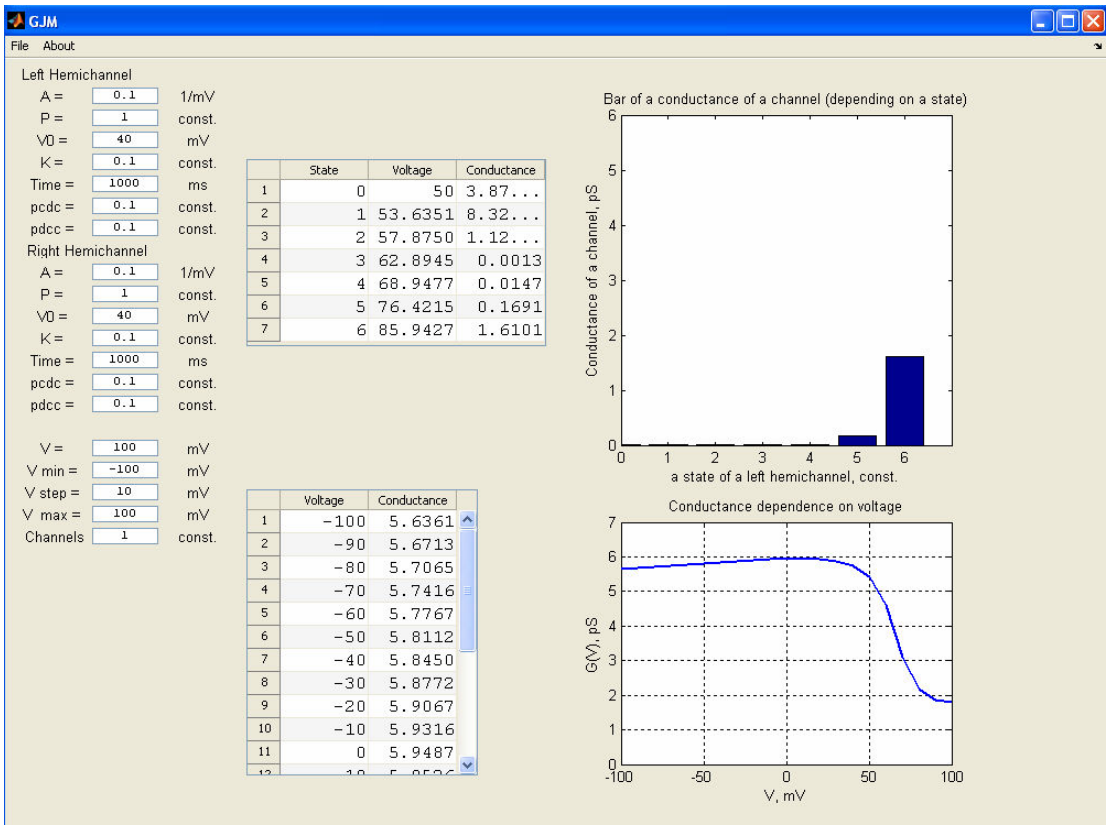
1.5 pav. GJM programos, kurtos C# programavimo kalba, langas

Programoje galima dirbti su dviejų nuosekliai sujungtų vartų, 4 nuosekliai sujungtų vartų (kai yra 2 „lėti“ vartai) ir su 6 lygiagrečiai sujungtų subvartų imitaciniais modeliais. Galima keisti parametrus.

Vaizduojami grafikai aiškūs, matomi (galima išsikelti grafikų langus ir padidinti, galima padidinti kiekvieno grafiko tam tikrą atkarpą).

Galima matyti reikšmes ir pasididinus grafiko tam tikrą vietą. Tai yra patogu, nes rezultatų vertinimas yra kokybinis.

Stacionariems modeliams kurti (Markovo grandinių modeliams) buvo pasirinkta programavimo kalba MatLab, kadangi modeliuose naudojami masyvai ir vektoriai ir atliekami veiksmai su jais. Programa atlikta MatLab 2010 a versija. Programos langas pavaizduotas 1.6 paveiksle.



1.6 pav. GJM Markovo modelių programos langas

Programoje galima dirbti su 6 modeliais: 4 diskretaus laiko Markovo grandinių (2 būsenų 6 ir 12 koneksinų ir 3 būsenų 6 ir 12 koneksinų) ir 2 tolydaus laiko Markovo grandinų (2 būsenų 6 ir 12 koneksinų). Rodomi rezultatai vieno kanalo arba kanalų (priklausomai nuo to, koks nustatytas skaičius „Channels“ laukelyje) prie nustatytos vienos įtampos reikšmės V, taip pat rodomi rezultatai, esant intervalui V (nuo minimalios „V min“ iki maksimalios „V max“ įtampos reikšmės žingsniu „V step“). Grafikai rodomi yra 2: vienas rodo laidžio vertę, esant skirtingai būsenai (užsidariusių koneksinų

skaičius nuo 0 iki 6 užsidariusių koneksinų), kitas rodo laidžio priklausomybės G(V) (pikosimensai) grafiką nuo įtampos V (milivoltai).

Kadangi Nerijaus Paulausko programoje nėra nagrinėjami kanalai, kurių kiekvienas koneksinas gali būti 3 būsenose, tai dar buvo naudotasi Sauliaus Vaičeliūno kurta programa, kuri analogiška Nerijaus Paulausko programai, todėl programos langas nepateikiamas.

1.3. REIKALAVIMAI KURIAMIEMS MODELIAMS, ALGORITMAMS, PROGRAMINEI ĮRANGAI IR ATLIEKAMIEMS TYRIMAMS

Specialių reikalavimų algoritmams ir programinei įrangai nėra. Svarbu, kad būtų išlaikyta modelio struktūra ir grąžinti reikiami rezultatai.

1.4. DARBE SPRENDŽIAMI UŽDAVINIAI

1. Sudaryti plyšinės jungties (PJ) 6 modelius:

a) diskretaus laiko Markovo grandinės 6 koneksinų, kurių kiekvienas gali būti 2 būsenų (arba atviras, arba uždaras), modelį;

b) diskretaus laiko Markovo grandinės 12 koneksinų, kurių kiekvienas gali būti 2 būsenų (arba atviras, arba uždaras), modelį;

c) tolydaus laiko Markovo grandinės 6 koneksinų, kurių kiekvienas gali būti 2 būsenų (arba atviras, arba uždaras), modelį;

d) tolydaus laiko Markovo grandinės 12 koneksinų, kurių kiekvienas gali būti 2 būsenų (arba atviras, arba uždaras), modelį;

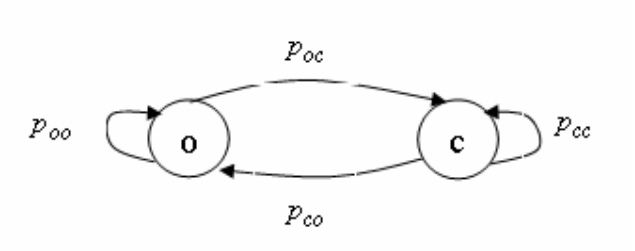
2. a)-b) modelius, kai koneksinas gali būti 3 būsenų (arba atviras, arba uždaras, arba visiškai uždaras).
3. Atlikti 1 punkte sudarytų modelių analizę (palyginti gautus modeliavimo rezultatus su imitacinio modeliavimo rezultatais).

2. METODOLOGINĖ DALIS

2.1. PJK KONCEPTUALUSIS MODELIS

Kanalas sudarytas iš dviejų nuosekliai sujungtų puskanalių. Puskanaliai yra nepriklausomi vienas nuo kito. Kiekvienas puskanalis yra sudarytas iš šešių koneksinų (subvartų) (schema buvo pateikta anksčiau (žr. 1.4 pav. A)), kurie gali būti arba atviroj (angl. *open, high*), arba uždaroj (angl. *closed, low, residual*) būsenoj. Kiekvieno puskanalio koneksinas (subvartas) apibūdinamas laidumu g, kuris priklauso nuo pridedamos įtampos ir nuo būsenos, kuri keičiasi keičiantis įtampai. Perėjimai tarp būsenų gali būti 4 (žr. 2.1 pav.):

- pereis iš atviros (angl. *o - open*) į uždarą (angl. *c - closed*);
- pasiliks toje pačioje būsenoje (*o*) (vadinasi, pereis į atvirą);
- pereis iš uždaros (*c*) į atvirą (*o*);
- pasiliks toje pačioje būsenoje (*c*).



2.1 pav. Perėjimai tarp būsenų

Vadinasi, perėjimų tikimybės gali būti 4 (žr. (2.1) - (2.4) formules).

$$p_{oc} = \frac{K \cdot k}{1+k}; \tag{2.1}$$

$$p_{oo} = 1 - p_{oc}; \tag{2.2}$$

$$p_{co} = \frac{K}{1+k}; \tag{2.3}$$

$$p_{cc} = 1 - p_{co}; \tag{2.4}$$

čia: $k = e^{A\cdot(\Pi\cdot V - V_0)}$

Π – įtampos poliškumas (+1 arba -1);

A – kanalo kairiojo arba dešiniojo puskanalio kiekvieno koneksino jautrio koeficientas;

K – kanalo kairiojo arba dešiniojo puskanalio koneksinų būsenų reguliavimo konstanta (sumažinanti tikimybę koneksinui pasilikti toje pačioje būsenoje).

V_o – kanalo kairiojo arba dešiniojo puskanalio įtampa, priklausanti nuo pusės maksimalaus laidžio.

Šis modelis yra stochastinis ir modelio parametrai (įtampa, laidis, srovės stipris) yra priklausomi nuo laiko.

Įtampa krinta ant viso kanalo. Kadangi kanalas sudarytas iš 2 nuosekliai sujungtų puskanalių, tai visa kanalo įtampa yra $V=V_1+V_2$, o laidis 2 puskanalių yra $g=\frac{g_1 \cdot g_2}{g_1+g_2}$, o kadangi kiekvieno puskanalio 6 koneksinai sujungti vienas su kitu lygiagrečiai, tai kairiojo puskanalio g_1=visų kairiojo puskanalio koneksinų laidžių suma. Tada kanalo laidžio formulė (žr. (2.6) formulę). Kiekvieno koneksino (subvarto) laidis priklauso nuo būsenos funkcijos ir pridedamos įtampos (žr. (2.5) formulę; $u_{ikj}(t_{m+1})$ – kiekvieno kanalo kiekvieno puskanalio koneksino įtampa laiko momentu t_{m+1}).

$$g_{ikj}(t_m)=f\left(s_{ikj}(t_m),u_{ikj}(t_m)\right)=\begin{cases}\frac{\Gamma_H}{6}\cdot\exp\left(\frac{u_{ikj}}{V_H}\right), & jei \quad s_{ikj}(t_m)=o,\\ \frac{\Gamma_L}{6}\cdot\exp\left(\frac{u_{ikj}}{V_L}\right), & jei \quad s_{ikj}(t_m)=c,\end{cases} \qquad \textbf{(2.5)}$$

$$j=\overline{1,6}, k\in\{l,r\},$$

čia: V_H ir V_L – aukštos (H) (800 mV) ir žemos (L) (300 mV) įtampos dažniai;

Γ_H ir Γ_L – aukšto (H) (12) ir žemo (L) (1,5) laidžio daugikliai.

čia: $u_i(t_m)$ – i-tojo kanalo įtampa,

$u_{ikj} := u_{ikj}(t_m)$ – kairiojo ($k=l$)/dešiniojo *($k=r$)* puskanalio koneksinų įtampa,

$s_{ikj}(t_m)\in\{o,c\}$ – kairiojo ($k=l$)/dešiniojo *($k=r$)* puskanalio koneksinų būsenos,

$g_{ikj}(t_m)$ – kairiojo ($k=l$)/dešiniojo *($k=r$)* puskanalio koneksinų laidžiai.

$$g_i(t_m)=\frac{\sum_{j=1}^{6} g_{ilj}(t_m)\cdot\sum_{j=1}^{6} g_{irj}(t_m)}{\sum_{j=1}^{6} g_{ilj}(t_m)+\sum_{j=1}^{6} g_{irj}(t_m)}, \qquad \textbf{(2.6)}$$

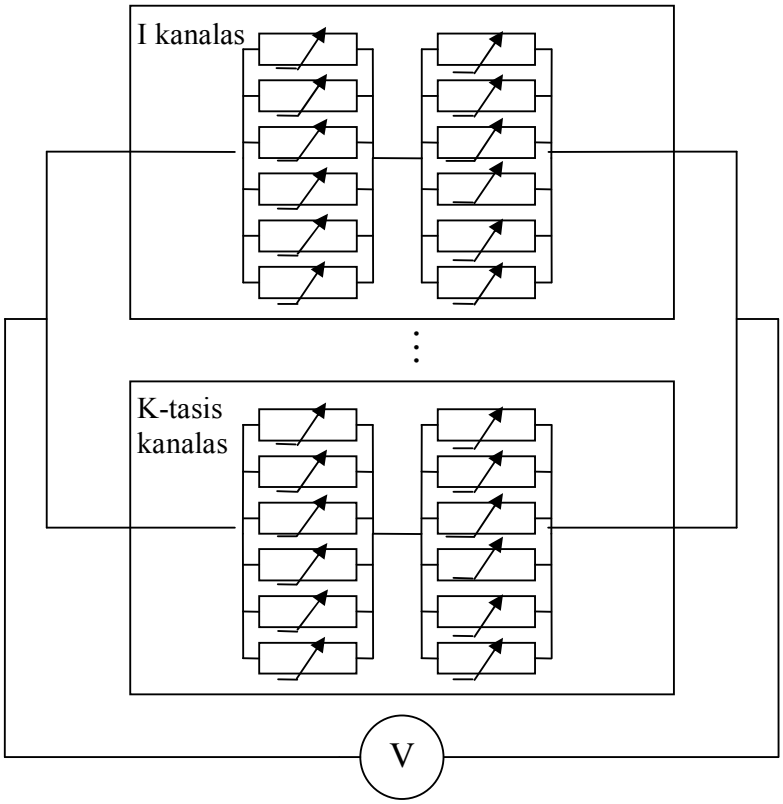
čia:

$g_i(t_m)$ – i-tojo kanalo laidis,

$g_{ilj}(t_m)$ – kairiojo puskanalio koneksinų laidžiai,

$g_{irj}(t_m)$ - dešiniojo puskanalio koneksinų laidis.

Skaičiuojamas laidis $g(t_{m+1})$ laiko momentu t_{m+1}. Kadangi sistemą sudaro ne vienas kanalas (žr. 2.2 paveikslą), tai susumuotų kiekvieno kanalo laidžių reikšmė dalinama iš kanalų skaičiaus. Šis modelis yra imitacinis, kuris buvo atliekamas Nerijaus Paulausko. Remiantis šiuo modeliu, tolimesniuose skyriuose bus nagrinėjami kiti modeliai, kuriami remiantis Markovo grandinių teorija.



2.2 pav. PJ schema

2.2. PJK DISKRETAUS LAIKO MARKOVO GRANDINĖS (DLMG) MODELIAI

Šiame skyriuje bus aptarti diskretaus (2.3 skyriuje tolydaus) laiko Markovo grandinės modeliai, kuriais bus tikrinamas imitacinis modelis, aptartas 2.1 skyriuje.

2.2.1. PJK DLMG 6 KONEKSINŲ 2 BŪSENŲ MODELIS

Šis modelis aprašytas straipsnyje, kuris skelbtas ir internete (Aurelija Sakalauskaitė; Henrikas Pranevičius; Mindaugas Pranevičius; Feliksas Bukauskas: 2011: 5).

PJK sudarytas iš 2 nuosekliai sujungtų puskanalių: kairiojo ir dešiniojo. Kiekvienas puskanalis sudarytas iš 6 porą formuojančių koneksinų. Kiekviens koneksinas gali būti 2 būsenų (žr. 2.1 skyrių). Tariama, kad tik kairiojo puskanalio koneksinai keičia būsenas, o dešiniojo puskanalio koneksinai yra atviri (žr. 1.4 paveikslą). PJK yra veikiamas įtampos V ir dėl to kiekvienas koneksinas keičia būseną *o*↔*c*. Kaip buvo aprašyta ankstesniuose darbuose (Rolf Vogel; Robert Weingart: 2002: 1), (Ye Chen-Izu; Alonso P. Moreno; Robert A. Spangler: 2001: 3), (Rolf Vogel; Virginijus Valiūnas; Robert Weingart: 2006: 4), perėjimų tikimybės tarp atviros ir uždaros būsenų aprašomos tokiais sąryšiais:

$$p_{oc}(A,P,V_{left},V_0)=\frac{K\cdot k\left(A,P,V_{left},V_0\right)}{1+k\left(A,P,V_{left},V_0\right)} \tag{2.7}$$

yra perėjimo *o*→*c* tikimybė;

$$p_{oo}(A,P,V_{left},V_0)=1-p_{oc}(A,P,V_{left},V_0) \tag{2.8}$$

yra tikimybė pasilikti atviroje būsenoje;

$$p_{co}(A,P,V_{left},V_0)=\frac{K}{1+k\left(A,P,V_{left},V_0\right)} \tag{2.9}$$

yra perėjimo c→o tikimybė;

$$p_{cc}(A,P,V_{left},V_0)=1-p_{co}(A,P,V_{left},V_0) \tag{2.10}$$

yra tikimybė pasilikti uždaroje būsenoje (žr. 2.1 paveikslą); čia $k\left(A,P,V_{left},V_0\right)=e^{A\cdot\left(P\cdot V_{left}-V_0\right)}$; P yra įtampos poliškumas (+1 or -1);

$$V_{left}\left(step,n\right)=\frac{V\cdot g_{right}\left(V-V_{left}\left(step-1,n\right),P\right)}{g_{left}\left(V_{left}\left(step-1,n\right),P\right)+g_{right}\left(V-V_{left}\left(step-1,n\right),P\right)}, \tag{2.11}$$

čia:

$$g_{right}\left(V - V_{left}\left(step-1,n\right),P\right) = 6 \cdot g_o\left(V - V_{left}\left(step-1,n\right),P\right) \quad \textbf{(2.12)}$$

yra dešiniojo puskanalio laidis ir

$$g_{left}\left(V_{left}\left(step-1,n\right),P\right) = \left(6-n\right)\cdot g_o\left(V_{left}\left(step-1,n\right),P\right) + n \cdot g_c\left(V_{left}\left(step-1,n\right),P\right) \quad \textbf{(2.13)}$$

yra kairiojo puskanalio laidis; $n = \overline{0,6}$; $step = \overline{1,100}$; $p_{ij}\left(A,P,V_{steady},V_0\right)$ $\left(i \in \{o,c\}; j \in \{o,c\}\right)$ – tikimybė $n = \overline{0,6}$ uždarų koneksinų pereiti iš atviros (o)/uždaros (c) būsenos į atvirą (o)/uždarą (c) būseną su kanalo poliškumu P ir įtampa V; A – koeficientas, apibūdinantis įtampos intensyvumą (1/mV); K – kanalo kairiojo puskanalio koneksinų būsenų reguliavimo konstanta (sumažinanti tikimybę koneksinui pasilikti toje pačioje būsenoje); V_o – puskanalio įtampa, priklausanti nuo pusės maksimalaus laidžio (mV); V_{left} – kairiojo puskanalio įtampa (mV); $V_{left}\left(step,n\right)$ – įtampa, kai n koneksinų yra uždari ir $\left(6-n\right)$ yra atviri; $step$ – žingsnių skaičius, su kuriuo pasiekiamas nusistovėjusios įtampos V_{steady} vektorius, kurio reikšmės yra 7 galimų būsenų įtampų vertės.

Tariama, kad pradinės sąlygos tokios: $step = 0$, o $V = 0$ mV.

Kiekvieno koneksino laidis g, priklausantis nuo įtampos $V_{left/right}$, gali kisti keičiantis koneksino atviros būsenos laidžiui $g_o = 2$ su pasirenkamais vienetais pikosimensais (pS) su uždaroje būsenoje išsiskiriančiu liekamuoju (angl. *residual*) laidžiu $g_r = 0.25$ pS. Be to, tariama, kad g_o ir g_c reikšmės reguliuojamos, t.y., priklauso nuo $V_{left/right}$ eksponentiškai:

$$g_o\left(V_{left/right},P\right) = 2 \cdot e^{\frac{P \cdot V_{left/right}}{800}}, \quad \textbf{(2.14)}$$

$$g_c\left(V_{left},P\right) = 0.25 \cdot e^{\frac{P \cdot V_{left}}{300}}, \quad \textbf{(2.15)}$$

čia: $V_{left/right}$ – kairiojo arba dešiniojo puskanalio įtampa.

Kai kiekvieno koneksino laidžiai yra $g_o\left(V_{left/right},P\right)$ ir $g_c\left(V_{left},P\right)$, tai PJK laidis randamas, naudojantis Markovo grandine.

Markovo grandinė – tai stochastinis diskretaus laiko procesas $\left\{X_n, n = 0,1,...\right\}$ su reikšmėmis (būsenomis) $i \in Z^+$, kai bet kokiam būsenų rinkiniui $i_0,...,i_{n-1};i,j$ galioja sąlyga $P(X_{n+1} = j \mid X_0 = i_0,...,X_{n-1} = i_{n-1}, X_n = i) = P(X_{n+1} = j \mid X_n = i)$.

Kadangi nagrinėjami kanalo kiekvieno puskanalio 6 koneksinai yra nuo laiko nepriklausomi, tai Markovo grandinė yra homogeninė ir perėjimo tikimybės nepriklauso nuo laiko ir žymimos taip:

$p_{ij} = P(X_{n+1} = j \mid X_n = i)$; čia p_{ij} – tikimybė, kad iš būsenos i per vieną žingsnį pereinama į būseną j. Šios tikimybės apibrėžia perėjimo tikimybių matricą (žr. konkretų atvejį – (2.16) formulę)

$perejimo\_matrica = \left((p_{ij})\right)$ (Eugenijus Manstavičius, 2007: 6).

$$perejimu\_matrica = \begin{pmatrix} p_{00} & p_{01} & p_{02} & p_{03} & p_{04} & p_{05} & p_{06} \\ p_{10} & p_{11} & p_{12} & p_{13} & p_{14} & p_{15} & p_{16} \\ p_{20} & p_{21} & p_{22} & p_{23} & p_{24} & p_{25} & p_{26} \\ p_{30} & p_{31} & p_{32} & p_{33} & p_{34} & p_{35} & p_{36} \\ p_{40} & p_{41} & p_{42} & p_{43} & p_{44} & p_{45} & p_{46} \\ p_{50} & p_{51} & p_{52} & p_{53} & p_{54} & p_{55} & p_{56} \\ p_{60} & p_{61} & p_{62} & p_{63} & p_{64} & p_{65} & p_{66} \end{pmatrix} \quad \textbf{(2.16)}$$

Yra 7 būsenos, t.y., būsena $n = \overline{0,6}$:

0 – būsena, kad 0 koneksinų uždaroje būsenoje, o 6 koneksinai bus atviroje būsenoje;

1 – būsena, kad 1 koneksinas uždaroje būsenoje, o 5 koneksinai atviroje būsenoje;

2 – būsena, kad 2 koneksinai uždaroje būsenoje, o 4 koneksinai atviroje būsenoje;

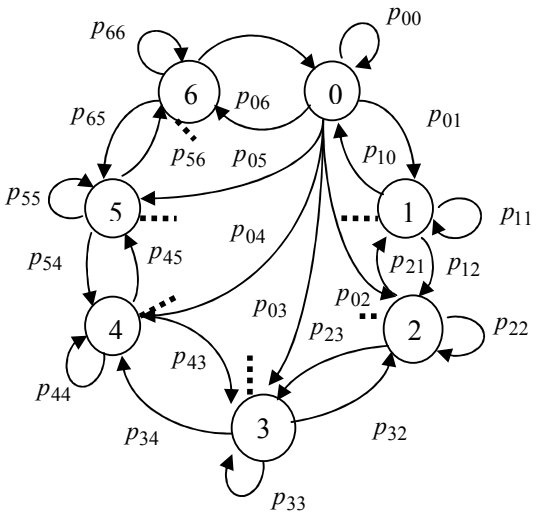
3 – būsena, kad 3 koneksinai uždaroje būsenoje, o 3 koneksinai atviroje būsenoje;

4 – būsena, kad 4 koneksinai uždaroje būsenoje, o 2 koneksinai atviroje būsenoje;

5 – būsena, kad 5 koneksinai uždaroje būsenoje, o 1 koneksinas atviroje būsenoje;

6 – būsena, kad 6 koneksinai uždaroje būsenoje, o 0 koneksinų atviroje būsenoje.

Sudarome būsenų grafą:



2.3 pav. PJK kairiojo puskanalio būsenų grafas

Čia pavaizduotas ne visas būsenų grafas (žr. 2.3 paveikslą). Ne tik būsena 0 gali pasikeisti į būsenas 1, 2, 3, 4, 5, 6, bet ir būsenos 1, 2, 3, 4, 5, 6 gali pereiti į kitas būsenas (pavaizduota daugtaškiais). Šiame paveiksle pavaizduotos tik 25 perėjimų tikimybės. Jų iš viso yra 49.

Pradinėje matricoje „perejimu_matrica“ (žr. (2.16) formulę) perėjimo tikimybė p_{00} reiškia, kad iš 0 koneksinų, esančių uždaroje būsenoje, ir 6 koneksinų, esančių atviroje būsenoje, pereinama į tą pačią būseną 0 (t.y. į 0 koneksinų, esančių uždaroje būsenoje, ir 6 koneksinų, esančių atviroje būsenoje). Analogiškas aiškinimas ir su kitais matricos „perejimu_matrica“ elementais.

Teigiama, jog vieno kanalo puskanaliai yra vienodi. Tai skaičiuoti belieka tik vieno puskanalio koneksinų užsidarymų tikimybes, kai kito puskanalio koneksinai yra atviri.

Perėjimų tikimybės skaičiuotos, remiantis šveicarų matematiko J. Bernulio formule. Tariama, kad puskanaliai veikia nepriklausomai vienas nuo kito. Kadangi tiriame tik vieną puskanalį, tai to puskanalio 6 koneksinai veikia taip pat nepriklausomai vienas nuo kito. Tarkim, turim vieną koneksiną iš 6, kuris užsidaro. Tai nesvarbu, kuris iš jų užsidarys. 6 koneksinai yra sujungti lygiagrečiai vienas su kitu.

$$\Pr_n(k) = C_n^k \cdot p^k \cdot q^{n-k}, \qquad \textbf{(2.17)}$$

čia: $C_n^k = \frac{n!}{k!(n-k)!}$ – derinių formulė; $n!$ – skaičiaus n faktorialas; $k = \overline{0,n}$, $n = \overline{0,6}$; k – skaičius kartų, kai įvyksta įvykis; p – perėjimo tikimybė $o \to c$ ($c \to o$); q – perėjimo tikimybė $(1-p)$, t.y., $o \to o$ ($c \to c$).

Kiekvienai perėjimo tikimybei skaičiuoti atliekami J. Bernulio eksperimentai. J. Bernulio eksperimentai – tai nepriklausomi ekperimentai, kurių kiekvieno metu gali įvykti tik įvykis M (apibūdinantis koneksino perėjimą $o \to c$) arba jam priešingas įvykis $\overline{M}$ (apibūdinantis koneksino perėjimą $o \to o$) su nekintančiomis visuose eksperimentuose tikimybėmis (Algimantas Aksomaitis, 2002: 7) $p_{oc}(A,P,V_{steady},V_0) = \Pr(M)$, $p_{oo}(A,P,V_{steady},V_0) = 1 - \Pr(M) = \Pr(\overline{M})$ ir įvykis N (apibūdinantis koneksino perėjimą $c \to o$) arba jam priešingas įvykis $\overline{N}$ (apibūdinantis koneksino perėjimą $c \to c$) su nekintačiomis visuose eksperimentuose tikimybėmis $p_{co}(A,P,V_{steady},V_0) = \Pr(N)$ ir $p_{cc}(A,P,V_{steady},V_0) = 1 - \Pr(N) = \Pr(\overline{N})$.

Kairįjį puskanalį, sudarytą iš 6 koneksinų, skaidome į dvi koneksinų grupes: atvirus koneksinus (I grupė) ir uždarus koneksinus (II grupė). Nagrinėjamos būsenos, į kurias pereina I ir II grupės koneksinai. Taigi, yra *pradinė būsena* (n_c, n_o); čia n_c – uždarų koneksinų skaičius ir n_o – atvirų koneksinų skaičius. Iš pradinės būsenos sudaromos dvi naujos būsenos: viena nauja būsena suformuota iš I grupės koneksinų ir vadinama *kita busena* $(n_c - k, k)$; čia $k = \overline{0,n_c}$; ir kita nauja būsena

suformuota iš II grupės koneksinų ir vadinama *kita busena* $(l, n_o - l)$; čia $l = \overline{0, n_o}$. I ir II grupės koneksinai keičia būsenas nepriklausomai vieni nuo kitų ir sudaro visas galimas būsenas $(n_c - k, k \quad l, n_o - l) = (n_c - k + l, k + n_o - l)$ ir, naudojantis (2.7)-(2.10) formulėmis, gaunamos perėjimų tikimybės

$$p_{n_c,(n_c-k+l)} = C_{n_c}^{n_c-k} p_{n_c cc}^{n_c-k} p_{n_c co}^{k} \cdot C_{n_o}^{n_o-l} p_{n_c oc}^{n_o-l} p_{n_c oo}^{l} \qquad \textbf{(2.18)}$$

čia: $n = n_c = n_o = \overline{0,6}$; $p_{n_c mk} := p_{mk}(P, V_{steady}(n_c))$; $m \in \{o, c\}; k \in \{o, c\}$; P – poliškumas.

Kai kurios būsenos yra vienodos, tačiau jų kombinacija, sudaryta iš I ir II grupių, yra skirtinga. Dėl to naudojama adityvumo tikimybinė aksioma (Algimantas Aksomaitis, 2002: 8), kuri teigia

$$\Pr(aibe1 \cup aibe2) = \Pr(aibe1) + \Pr(aibe2), \qquad \textbf{(2.19)}$$

čia: $aibe1 \cap aibe2 = \emptyset$; *aibė1* – viena aibė, sudaryta iš I ir II grupės koneksinų; *aibė2* – kita aibė, sudaryta iš I ir II grupės koneksinų; $aibė1 \cup aibė2$ – I ir II grupių aibių sąjunga; *Pr* – tikimybė.

Pavyzdžiui, kairiojo puskanalio koneksinai išsidėstę taip: *busena* $(1,5)$. I grupės koneksinai tuomet pereis į būseną *kita busena* $(1-k, k)$; čia: $k = \overline{0,1}$, ir II grupės koneksinai pereis į būseną *kita busena* $(l, 5-l)$; čia: $l = \overline{0,5}$. Taigi, sujungus Dekarto sandauga šias dviejų grupių būsenas gaunama $(1-k, k \quad l, 5-l) = (1-k+l, k+5-l)$ (žr. (2.18)-(2.19) formules).

Sudarius perėjimų matricą stacionariosios tikimybės p_j; $j = \overline{0,6}$ randamos iš tiesinių lygčių sistemos:

$$\begin{cases} p_j = \sum_{i=0}^{6} p_{i,j} \cdot p_i; j = \overline{0,6}, \\ \sum_{j=0}^{6} p_j = 1. \end{cases} \qquad \textbf{(2.20)}$$

čia: $p_{i,j}$ – matrica, kai $i = \overline{0,6}$, $j = \overline{0,6}$.

Radus stacionariąsias tikimybes skaičiuojamas PJK laidis $g_i(V, V_{left}, n)$:

$$g_i(V, V_{left}, n) = \frac{p_n \cdot g_{left}(V_{left}, P) \cdot g_{right}(V_{right}, P)}{g_{left}(V_{left}, P) + g_{right}(V_{right}, P)}, \qquad \textbf{(2.21)}$$

čia: p_n – stacionarioji tikimybė būsenos $n = \overline{0,6}$, kuri rodo, kiek koneksinų yra uždarytų;

$$g_{left}\left(V_{left}, P\right) = (6-n)g_o(V_{left}, P) + n \cdot g_c(V_{left}, P) \quad \textbf{(2.22)}$$

yra *i*-tojo PJK kairiojo puskanalio laidis;

$$g_{right}\left(V_{right}, P\right) = 6 \cdot g_o(V_{right}, P) \quad \textbf{(2.23)}$$

yra *i*-tojo PJK dešiniojo puskanalio laidis; $g_i(V, V_{left}, n)$ – *i*-tojo PJK $n = \overline{0,6}$ būsenos laidis; $V_{left} = V_{steady}$; $V_{right} = V - V_{steady}$.

Kadangi PJ sudaryta iš daugiau nei vieno kanalo, tai esant nusistovėjusiai būsenai ir nustatytai įtampai V laidis

$$g(V, V_{left}, P) = \sum_{i=1}^{j} g_i(V, V_{left}, P), \quad \textbf{(2.24)}$$

čia:

$$g_i(V, V_{left}, P) = \sum_{n=0}^{6} g_i(V, V_{left}, P, n) \quad \textbf{(2.25)}$$

ir *j* – PJ kanalų skaičius.

2.2.2. PJK DLMG 12 KONEKSINŲ 2 BŪSENŲ MODELIS

Šis modelis remiasi 2.2.1 skyriuje aprašytu modeliu. Šiame modelyje tiek kairiojo, tiek dešiniojo puskanalių įtampa yra kintanti, t.y., keičia būsenas tiek kairiojo, tiek dešiniojo puskanalio koneksinai, nepriklausomai vieni nuo kitų. Todėl ieškoma nusistovėjusi įtampa (žr. (2.26) formulę). Pradiniame žingsnyje (*i=0*) tariama, kad tiek kairiojo, tiek dešiniojo puskanalių koneksinai yra atviri, t.y. $V_{left(right),0} = 0$.

$$V_{left(right),i} = \frac{V \cdot g_{right(left)}\left(V_{(right)left,i-1}\right)}{g_{left}\left(V_{left,i-1}\right) + g_{right}\left(V_{right,i-1}\right)}; \quad \textbf{(2.26)}$$

čia:

V – kanalo įtampa (mV);

$g_{left}\left(V_{left,i}\right)=(6-n_{c\_left})\cdot g_{o\_left}\left(V_{left,i}\right)+n_{c\_left}\cdot g_{c\_left}\left(V_{left,i}\right)$ – kairiojo puskanalio laidis, priklausantis nuo kairiojo puskanalio įtampos; $n_{c\_left}=\overline{0,6}$ – kairiojo puskanalio užsidariusių koneksinų skaičius;

$g_{right}\left(V_{right,i}\right)=(6-n_{c\_right})\cdot g_{o\_right}\left(V_{right,i}\right)+n_{c\_right}\cdot g_{c\_right}\left(V_{right,i}\right)$ – dešiniojo puskanalio laidis, priklausantis nuo dešiniojo puskanalio įtampos; $n_{c\_right}=\overline{0,6}$ – dešiniojo puskanalio užsidariusių koneksinų skaičius;

$g_{o\_left(right)}\left(V_{left(right),i}\right)=2\cdot e^{\frac{P\cdot V_{left(right),i}}{800}}$ – kairiojo (dešiniojo) puskanalių koneksino laidis, kai koneksinas yra atviras;

$g_{c\_left(right)}\left(V_{left(right),i}\right)=0.25\cdot e^{\frac{P\cdot V_{left(right),i}}{300}}$ – dešiniojo (kairiojo) puskanalių koneksino laidis, kai koneksinas yra uždaras;

$i=\overline{1,i_{steady}}$; i_{steady} – žingsnis, kai $V_{left(right),i}=V_{left(right),i-1}$.

Kai $V_{left(right),i}=V_{left(right),i-1}$, tai žymima $V_{left(right)}:=V_{left(right),i_{steady}}$.

Markovo modelis sudaromas norint rasti stacionariųjų tikimybių vektorių **p**. Sudaroma perėjimų matrica $\left(p_{i,j}\right)$; $i=\overline{0,48}$; $j=\overline{0,48}$. Kad būtų gauta tokia matrica, nesudaroma būsenų aibė taip, kaip tai buvo atlikta 2.2.1 skyriuje, o tiesiog pasinaudojama formulėmis (2.18)-(2.19) ir pavadinama pirmoji matrica *kairiojo puskanalio perėjimų matrica* (*perejimu_matrica_kp*), ir analogiškai sudaroma *dešiniojo puskanalio perėjimų matrica* (*perejimu_matrica_dp*). Tai abiejų puskanalių bendra perėjimų matrica (*perejimu_matrica_bendra*) bus tokia:

$$perejimu\_matrica\_bendra = perejimu\_matrica\_kp(i,j)\cdot \\ \cdot perejimu\_matrica\_dp \qquad \textbf{(2.27)}$$

čia: $i=\overline{0,48}$ ir $j=\overline{0,48}$ reprezentuoja būsenų $n_{c\_left(right)}=\overline{0,6}$; $n_{o\_left(right)}=\overline{0,6}$ Dekarto sandaugas.

Kadangi kairiojo ir dešiniojo puskanalių koneksinai keičia būsenas nepriklausomai vieni nuo kitų, tai kiekvienas kairiojo puskanalio perėjimų matricos elementas yra dauginamas iš visos dešiniojo puskanalio perėjimų matricos.

Šios matricos (žr. (2.27) formulę) kiekvienos eilutės elementai yra normuojami, t.y.,

$$perejimu\_matrica\_bendra_{i,j} = \frac{perejimu\_matrica\_bendra_{i,j}}{\sum_{k=0}^{48} perejimu\_matrica\_bendra_{i,k}}; \quad \textbf{(2.28)}$$

čia: $i = \overline{0,48}; j = \overline{0,48}$.

Normuojama dėl to, kad matricos kai kurių eilučių elementų suma nėra lygi vienetui (o artima vienetui) dėl perėjimų matricos dirbtinio formavimo (t.y. sudarymo iš kairiojo ir dešiniojo puskanalių perėjimų matricų).

Remiantis gautąja perėjimų matrica, randamas stacionariųjų tikimybių vektorius **p**. Radus vektorių suskaičiuojamas *i*-tojo kanalo laidis

$$g_i = \sum_{indeksas=0}^{48} \frac{p_{indeksas} \cdot g_{left}(V_{left}) \cdot g_{right}(V_{right})}{g_{left}(V_{left}) + g_{right}(V_{right})}; \quad \textbf{(2.29)}$$

čia:

$p_{indeksas}$ – stacionarioji $indeksas = \overline{0,48}$ būsenos tikimybė;

$g_{left(right)}\left(V_{left(right)}\right)$ – kairiojo arba dešiniojo puskanalio laidis, gautas, esant nusistovėjusiai įtampai $V_{left(right)}$.

2.2.3. PJK DLMG 6 KONEKSINŲ 3 BŪSENŲ MODELIS

Šis modelis remiasi 2.2.1 skyriuje aprašytu modeliu. Kiekvienas koneksinas gali būti 3 būsenų: atviras (angl. *open* (žymima *o*)), uždaras (angl. *closed* (žymima *c*)) ir visiškai uždaras (angl. *deep closed* (žymima *dc*)). Tariama, kad tik kairiojo puskanalio koneksinai keičia būsenas, o dešiniojo puskanalio koneksinai yra atviri (žr. 1.4 paveikslą). PJK yra veikiamas įtampos V ir dėl to kiekvienas koneksinas keičia būseną *o*↔*c* ir *c*↔*dc*. Kaip buvo aprašyta ankstesniuose darbuose (Rolf Vogel; Robert Weingart: 2002: 1), (Ye Chen-Izu; Alonso P. Moreno; Robert A. Spangler: 2001: 3), (Rolf Vogel; Virginijus Valiūnas; Robert Weingart: 2006: 4), perėjimų tikimybės tarp atviros ir uždaros būsenų aprašomos sąryšiais (2.7)-(2.8), o sąryšiai (2.9)-(2.10) yra šiek tiek keičiami (žr. (2.30)-(2.31) formules) ir dar papildomai įvedamos tikimybės

p_{cdc} – tikimybė koneksinui pereiti iš uždaros „c“ būsenos į visiškai uždarą „dc“;

ir

p_{dcc} – tikimybė koneksinui pereiti iš visiškai uždaros „dc“ būsenos į uždarą „dc“;

$p_{dcdc} = 1 - p_{dcc}$.

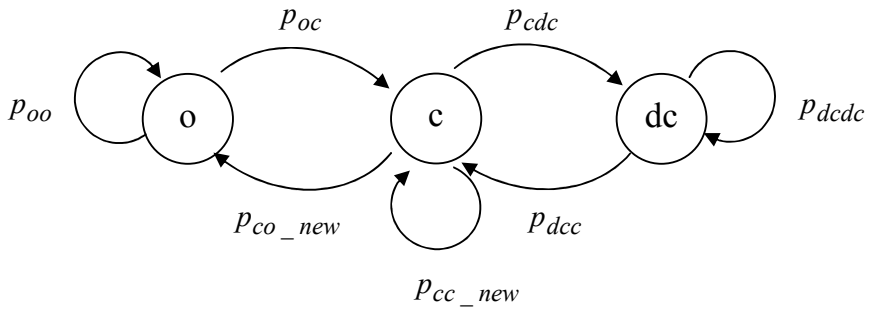
Tikimybės p_{cdc} ir p_{dcc} yra pasirenkamos laisvai, t.y., jos kinta intervale [0;1]. Iš buvusių tikimybių p_{co} ir p_{oc} (žr. (2.9)-(2.10) formules) sudaromos naujos tikimybės p_{co_new} ir p_{cc_new} (žr. (2.30)-(2.31) formules), kurios susietos tikimybe p_{cdc}.

$$p_{co\_new}(A,P,V_{left},V_0,p_{cdc}) = p_{co}(A,P,V_{left},V_0)\cdot(1-p_{cdc}) \quad \textbf{(2.30)}$$

yra nauja perėjimo c→o tikimybė;

$$p_{cc\_new}(A,P,V_{left},V_0,p_{cdc}) = p_{cc}(A,P,V_{left},V_0)\cdot(1-p_{cdc}) \quad \textbf{(2.31)}$$

yra nauja tikimybė pasilikti uždaroje būsenoje (žr. 2.4 paveikslą); *P* yra įtampos poliškumas (+1 or -1).



2.4 pav. Perėjimai tarp 3 būsenų

Keičiama kairiojo puskanalio laidžio formulė (žr. (2.13) formulę), nes pridedama būsena „visiškai uždara“ (žr. (2.32) formulę).

$$\begin{aligned} g_{left}\left(V_{left}(step-1,n),P,n_{dc},n_c,n_o\right) &= n_{dc}\cdot g_c\left(V_{left}(step-1,n),P\right)+ \\ &+ n_c\cdot g_c\left(V_{left}(step-1,n),P\right)+ n_o\cdot g_o\left(V_{left}(step-1,n),P\right) \end{aligned} \quad \textbf{(2.32)}$$

čia: $n=\overline{1,28}$, nes tiek yra galimų susidarančių būsenų, t.y., $6 = n_{dc}+n_c+n_o$, kai

$n_{dc}=\overline{0,6}$ – visiškai uždarų koneksinų skaičius,

$n_c=\overline{0,6}$ – uždarų koneksinų skaičius,

$n_o=\overline{0,6}$ – atvirų koneksinų skaičius.

Taigi, n yra galimų kombinacijų, sudarančių sumą 6, skaičius.

Markovo grandinės perėjimų matrica šiuo atveju bus 28x28 dydžio, nes yra 28 galimos būsenos. Taip pat pakinta perėjimų matricos sudarymo formulė, t.y. formulė (2.18) keičiama į (2.33) formulę

$$p_{n_{dc},n_c,n_o} = C_{n_{dc}}^{n_{dc}-x} p_{dcc}^{n_{dc}-x} p_{dcdc}^{x} \cdot \\ \cdot \frac{n_c!}{(n_{dc}-x)!(n_c-y)!(n_o-z)!} \cdot p_{cdc}^{n_{dc}-x} p_{cc\_new}^{n_c-y} p_{co\_new}^{n_o-z} \cdot \\ \cdot C_{n_o}^{n_o-z} p_{oc}^{n_o-z} p_{oo}^{z} \qquad \textbf{(2.33)}$$

čia: $n = n_{dc} = n_c = n_o = \overline{0,6}$; $p_{ab}(P, V_{steady}(n))$ – perėjimų tikimybės, minėtos anksčiau; $a \in \{o,c\}; b \in \{o,c\}$; $a,b \in \{c,dc\}$, $x = \overline{0,n_{dc}}, y = \overline{0,n_c}, z = \overline{0,n_o}$; P – poliškumas.

Formulė (2.33) savo struktūra panaši į formulę (2.18), tačiau (2.33) koneksinų grupės yra 3, o ne 2: n_{dc}, n_c, n_o. Turint 28 kombinacijas, kurios sudaro sumą 6, kiekviena tokia kombinacija skaidoma į 3 grupes: 1) $\{n_{dc},0,0\}$; 2) $\{0,n_c,0\}$; ir 3) $\{0,0,n_o\}$. Kiekviena iš šių grupių taip pat sudaro kombinacijas, kurių sumos atitinkamai lygios n_{dc}, n_c ir n_o. Formulei (2.33) panaudota ne tik binominis skirstinys, bet ir bendresnis atvejis – multinominis skirstinys. Šis skirstinys reikalingas, nes antroje grupėje $\{0,n_c,0\}$ iš uždaros būsenos koneksinas gali pereiti ne tik į atvirą būseną arba pasilikti toje pačioje uždarytoje būsenoje, bet ir pereiti į visiškai uždarą būseną. Todėl šiuo atveju yra jau trys galimos tikimybės: tikimybė pereiti koneksinuo iš uždaros būsenos į visiškai uždarą būseną, tikimybė pasilikti koneksinui uždaroje būsenoje, tikimybė pereiti koneksinui iš uždaros būsenos į atvirą būseną.

Sudarius perėjimų matricą spredžiama Kolmogorovo-Čapmeno lygčių sistema ir gautos tikimybės panaudojamos plyšinės jungties kanalo laidžiui skaičiuoti (formulės analogiškos (2.20)-(2.25) formulėms, išskyrus tai, kad šiuo atveju $n = \overline{1,28}$ arba kitaip tariant n yra išskaidyta į tris grupes: $n_{dc} + n_c + n_o = 6$; $n_{dc} = \overline{0,6}, n_c = \overline{0,6}, n_o = \overline{0,6}$).

2.2.4. PJK DLMG 12 KONEKSINŲ 3 BŪSENŲ MODELIS

Šis modelis savo struktūra panašus į 2.2.2 skyriuje aprašytą modelį, tačiau fomulės tokios, kaip pateiktos 2.2.3 skyriuje, t.y. perėjimų matrica, laidžių formulės yra tokios, kaip pateiktos 2.2.3 skyriuje.

Kiekviens koneksinas gali būti 3 būsenų: atviras (angl. *open* (žymima *o*)), uždaras (angl. *closed* (žymima *c*)) ir visiškai uždaras (angl. *deep closed* (žymima *dc*)). Šiame modelyje tiek kairiojo, tiek dešiniojo puskanalių įtampa yra kintanti, t.y., keičia būsenas tiek kairiojo, tiek dešiniojo puskanalio koneksinai, nepriklausomai vieni nuo kitų.

Iš esmės šis skyrelis analogiškas 2.2.2, išskyrus tai, kad indeksacija yra $\overline{0,(28 \cdot 28)-1}$, o ne $\overline{0,(7 \cdot 7)-1}$ ir, kaip minėta aukščiau, perėjimų matrica ir laidžių formulės 2.2.3 skyrelio.

2.3. PJK TOLYDAUS LAIKO MARKOVO GRANDINĖS (TLMG) MODELIAI

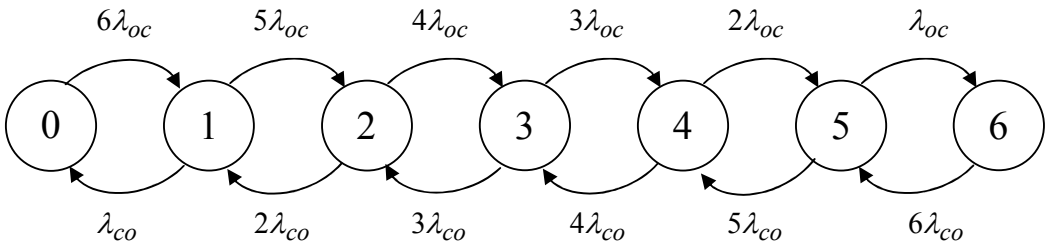
2.3.1. PJK TLMG 6 KONEKSINŲ 2 BŪSENŲ MODELIS

Šis modelis yra kuriamas panašiai, kaip ir modelis, aprašytas 2.2.1 skyriuje, išskyrus tai, kad čia bus kuriama tolydaus laiko Markovo grandinė.

Tolydaus laiko Markovo grandinė – tai stochastinis procesas $\{X(t), t \geq 0\}$, kurio būsenų aibė yra baigtinė $\{0,1,2,...\}$ ir turintis Markovo savybę

$\Pr\left(X_{s+t} = j \middle| X_s = i, X_{s_n} = i_n, ..., X_{s_1} = i_1\right) = \Pr\left(X_{s+t} = j \middle| X_s = i\right)$ $\forall t > 0, s > s_n > ... > s_1 \geq 0$ ir $i, j, i_k \in Z^+$.

Apibrėžiama dažnių matrica Q, kuri yra išvestinė tikimybių įvykių, kintančių per be galo mažą laiko intervalą. Tariama, kad koneksinai gali keisti būseną per vieną žingsnį dažniu $n \cdot \lambda_{oc}(V_{left})$ (čia: $n = \overline{0,6}$, $\lambda_{oc}(V_{left}) = \frac{p_{oc}(A,P,V_{left})}{time\_of\_a\_connexin}$) ir $n \cdot \lambda_{co}(V_{left})$ (čia: $n = \overline{0,6}$, $\lambda_{co}(V_{left}) = \frac{p_{co}(A,P,V_{left})}{time\_of\_a\_connexin}$; *time_of_a_connexin* – koneksino buvimo toje pačioje būsenoje laikas (įvedama vartotojo reikšmė programoje)). Taigi, naudojamasi tik 2 tikimybėmis, nes, remiantis tolydaus laiko Markovo grandinės teorija, koneksinai pasilikti toje pačioje būsenoje negali (6 koneksinų būsenų grafas parodytas žemiau).



2.5 pav. Kairiojo puskanalio būsenų grafas su dažniais

Remiantis 2.5 paveiksle pavaizduotu grafu, sudaroma dažnių matrica:

$$Q=\begin{pmatrix} -6\lambda_{oc} & 6\lambda_{oc} & 0 & 0 & 0 & 0 & 0 \\ \lambda_{co} & -(\lambda_{co}+5\lambda_{oc}) & 5\lambda_{oc} & 0 & 0 & 0 & 0 \\ 0 & 2\lambda_{co} & -(2\lambda_{co}+4\lambda_{oc}) & 4\lambda_{oc} & 0 & 0 & 0 \\ 0 & 0 & 3\lambda_{co} & -(3\lambda_{co}+3\lambda_{oc}) & 3\lambda_{oc} & 0 & 0 \\ 0 & 0 & 0 & 4\lambda_{co} & -(4\lambda_{co}+2\lambda_{oc}) & 2\lambda_{oc} & 0 \\ 0 & 0 & 0 & 0 & 5\lambda_{co} & -(5\lambda_{co}+\lambda_{oc}) & \lambda_{oc} \\ 0 & 0 & 0 & 0 & 0 & 6\lambda_{co} & -6\lambda_{co} \end{pmatrix} \quad \textbf{(2.34)}$$

Kitaip tariant, Q reikšmės gaunamos taip: $q_{ij}=\begin{cases} (6-i)\cdot\lambda_{oc}, \text{ jei } i-j=-1; \\ i\cdot\lambda_{co}, \text{ jei } i-j=1; \\ -\sum\limits_{k\neq i} q_{ik}, jei\ i=j; \\ 0, \text{ kitu atveju.} \end{cases}$

Kadangi norima gauti stacionarų PJK laidį, tai, remiantis Kolmogorovo balanso lygtimi ir normavimo sąlyga, gaunama lygčių sistema (žr. (2.35) formulę), kurią išsprendus gaunamas stacionariųjų tikimybių vektorius Π, naudojamas laidžio formulėje (žr. (2.21)-(2.25) formules).

$$\begin{cases} \sum\limits_{i=0}^{6} \Pi_i \cdot q_{ij} = 0; \\ \sum\limits_{i=0}^{6} \Pi_i = 1 \end{cases} \quad \textbf{(2.35)}$$

2.3.2. PJK TLMG 12 KONEKSINŲ 2 BŪSENŲ MODELIS

Šis modelis sudaromas panašiai, kaip ir aprašytas 2.3.1 skyriuje, tačiau struktūra (matricos formavimas, laidžio skaičiavimas) analogiškas modeliui, aprašytam 2.2.2 skyriuje. Pradžia (įtampos nusistovėjimas) aprašoma lygiai taip pat, kaip ir 2.2.2 skyriuje. Šiuo atveju ieškoma ne perėjimų matrica, bet dažnių (intesyvumų) matrica, tačiau bendros matricos sudarymo formulė panaši į (2.27) formulę, išskyrus tai, kad kairiojo puskanalio intensyvumų matricos Q (žr. (2.34) formulę) (žymima *intensyvumu_matrica_kp*) kiekvienas elmentas yra pridedamas prie dešiniojo puskanalio intensyvumų matricos (*intensyvumu_matrica_dp*) (žr. (2.36) formulę). Be to, diagonalės elementai kairiosios ir dešiniosios intensyvumų matricų nesumuojami. Kai sudaroma bendra (abiejų puskanalių) matrica (*intensyvumu_matrica_bendra*), tai diagonalės elementams priskiriamos kiekvienos bendros perėjimų matricos eilutės sumos su neigiamais ženklais, kad matricos kiekvienos eilutės suma būtų lygi 0.

$$\begin{aligned} &inltensyvumu\_matrica\_bendra = inltensyvumu\_matrica\_kp(i,j) + \\ &+ inltensyvumu\_matrica\_dp \end{aligned} \quad \textbf{(2.36)}$$

čia: $i = \overline{0,48}$ ir $j = \overline{0,48}$ reprezentuoja būsenų $n_{c\_left(right)} = \overline{0,6}$; $n_{o\_left(right)} = \overline{0,6}$ Dekarto sandaugas.

Kadangi norima gauti stacionarų PJK laidį, tai, remiantis Kolmogorovo balanso lygtimi ir normavimo sąlyga, gaunama lygčių sistema (žr. (2.37) formulę), kurią išsprendus gaunamas stacionariųjų tikimybių vektorius Π, naudojamas laidžio formulėje (žr. (2.21)-(2.25) formules).

$$\begin{cases} \sum_{i=0}^{48} \Pi_i \cdot q_{ij} = 0; \\ \sum_{i=0}^{48} \Pi_i = 1 \end{cases} \quad \textbf{(2.37)}$$

3. TIRIAMOJI DALIS

3.1. PJK DLMG MODELIŲ ANALIZĖ

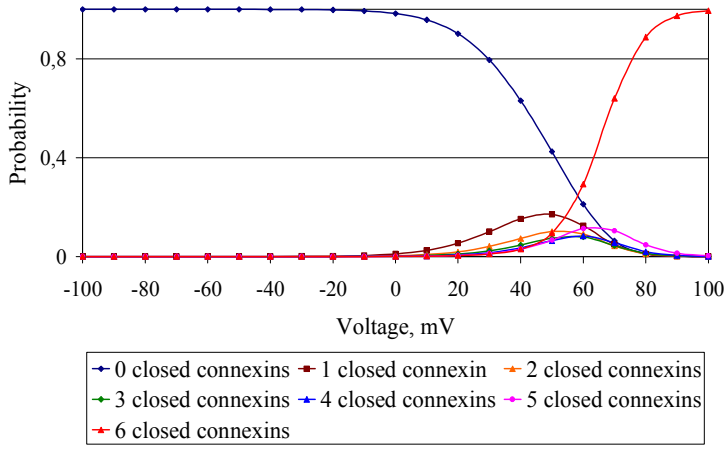
3.1.1. PJK DLMG 6 KONEKSINŲ 2 BŪSENŲ MODELIO TYRIMAS

Kad galima būtų atlikti laidžių skaičiavimus, remiantis (2.21)-(2.25) formulėmis, pasirenkamos reikšmės parametrų, apibūdinančių koneksinų būsenų kitimo savybes (žr. 3.1 lentelę).

3.1 lentelė. Pasirinktų parametrų reikšmės

Parameterai	Reikšmės (vienetai)
A	0.1 (1/mV)
P	1 (const.)
V	-100:20:100 (mV)
V_0	40 (mV)
$g_o\left(V_{left/right}, P\right)$	$2 \cdot e^{\frac{P \cdot V_{left/right}}{800}}$ (pS)
$g_c\left(V_{left}, P\right)$	$0.25 \cdot e^{\frac{P \cdot V_{left}}{300}}$ (pS)
K	0.1 (const.)

Formulėmis (2.18)-(2.20) suskaičiuojamos PJK stacionariosios tikimybės, keičiant PJK įtampą ($V = -100 \div 100$ mV). Šių tikimybių kitimas, priklausomai nuo įtampos, parodytas 3.1 paveiksle.



3.1 pav. PJK kairiojo puskanalio 7 būsenų (t.y. 6 koneksinų 2 (open-closed) būsenų) stacionariųjų tikimybių priklausomybės nuo įtampos V grafikas

Kai $V = -100$ mV, tikimybė, kad visi koneksinai bus atviri, lygi vienam. Kai $V = 100$ mV, tikimybė, kad visi koneksinai bus uždari, taip pat lygi vienam. Neįmanoma, kad užsidarys daugiau nei vienas, bet mažiau nei 6 koneksinai, atitinkamai kai $V = -100$ mV ir $V = 100$ mV. Kai $V = -100 \div 100$ mV, likusios būsenos visos tikėtinos.

Laidžio reikšmės, gautos keičiant įtampą V, yra palygintos su reikšmėmis, gautomis atlikus simuliaciją (Nerijus Paulauskas; Mindaugas Pranevičius; Henrikas Pranevičius; Feliksas Bukauskas: 2009: 9). Simuliacijos rezultatai buvo patikrinti su eksperimentiniais rezultatais ir yra adekvatūs. Modelių reikšmės skiriasi iki 10 % (žr. 3.2 lentelę).

3.2 lentelė. Simuliacijos ir diskretaus laiko Markovo 6 koneksinų 2 būsenų modelių rezultatai, naudojantis parametrais, pasirinktais 3.1 lentelėje.

Įtampa, mV	**Laidis, pS**		**Santykinė paklaida (%)**
	Simuliacijos rezultatai (t=1000 ms)	**Markovo modelio rezultatai**	
-100	5,5736	5,6365	1,12
-80	5,643	5,7074	1,13
-60	5,7131	5,7791	1,14
-40	5,7583	5,8515	1,59
-20	5,8018	5,9228	2,04
0	5,8276	5,9814	2,57
20	5,7916	5,9585	2,8
40	5,2061	5,5769	6,65

60	3,987	3,942	1,14
80	1,8875	1,906	0,97
100	1,5811	1,7277	8,49

Reikšmės skiriasi 0,1-0,2 pikosimenso. Galima teigti, kad Markovo modelio rezultatai yra adekvatūs simuliacijos rezultatams.

3.1.2. PJK DLMG 12 KONEKSINŲ 2 BŪSENŲ MODELIO TYRIMAS

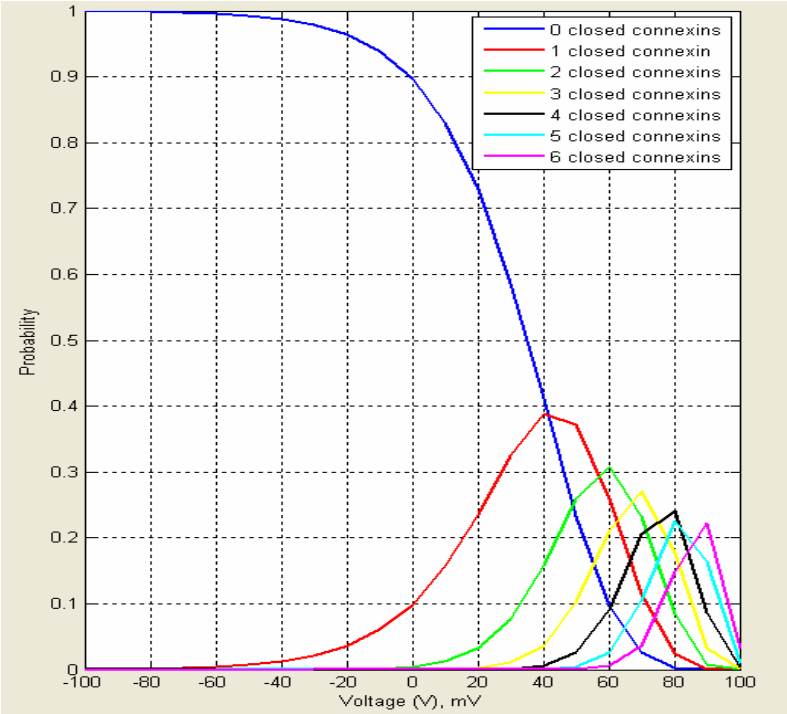
Tiriant šį modelį naudojamasi taip pat 3.1 lentele ir gauti tokie Markovo modelio rezultatai:

3.3 lentelė. Simuliacijos ir diskretaus laiko Markovo 12 koneksinų 2 būsenų modelių rezultatai, naudojantis parametrais, pasirinktais 3.1 lentelėje.

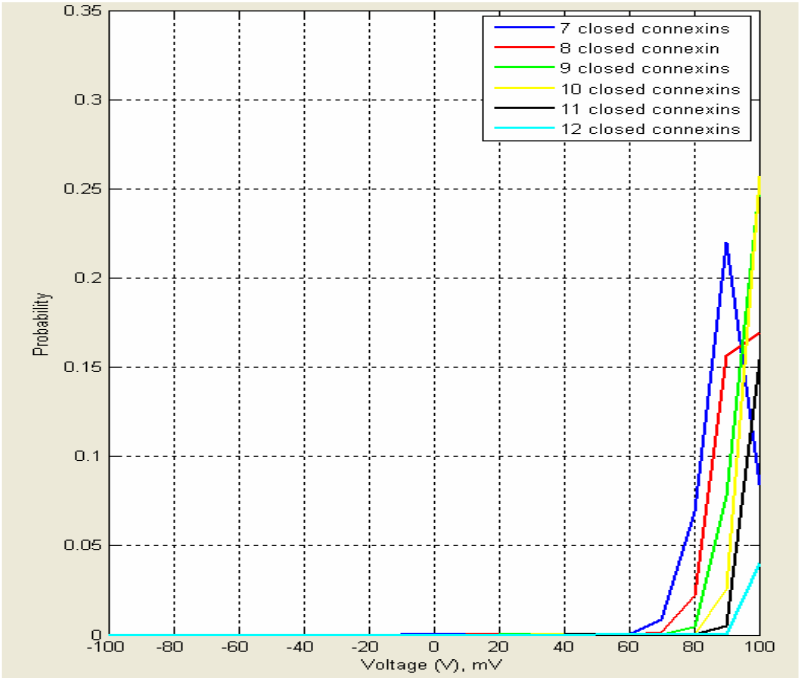
Įtampa, mV	**Laidis, pS**		**Santykinė paklaida (%)**
	Simuliacijos rezultatai (t=1000 ms)	**Markovo modelio rezultatai**	
-100	5,1712	5,2844	2,14
-80	5,1778	5,3860	3,87
-60	5,1930	5,4885	5,38
-40	5,2082	5,5902	6,83
-20	5,2232	5,6863	8,14
0	4,9405	5,7621	14,26
20	4,9600	5,7761	14,13
40	4,9796	5,6183	11,37
60	4,9988	5,0694	1,39
80	4,7245	4,0063	17,93
100	1,7444	1,8641	6,42

Matyti, kad reikšmės artimos, kaip ir 6 koneksinų atveju, tačiau skirtumai didesni.

Kadangi šiame modelyje nagrinėjama 12 koneksinų, tai ir nagrinėjama užsidariusių 0-12 koneksinų tikimybių priklausomybė nuo įtampos (žr. 3.2-3.3 paveikslus).



3.2 pav. 12 (nuo 0 iki 6 užsidariusių) koneksinų 2 būsenų tikimybių priklausomybės nuo įtampos grafikas



3.3 pav. 12 (nuo 7 iki 12 užsidariusių) koneksinų 2 būsenų tikimybių priklausomybės nuo įtampos grafikas

Žvelgiant į aukščiau nubraižytus grafikus matyti, kad įtampai didėjant tikimybė, kad užsidarys 0 koneksinų, t.y. kad visi koneksinai bus atviri, mažėja nuo 1, kai įtampa yra -100 mV, iki 0, kai įtampa yra 100 mV. Įtampai didėjant tikimybė, kad užsidarys 12 koneksinų, didėja nuo 0, kai įtampa yra 0

mV, iki ~0,05, kai įtampa yra 100 mV. Tikimybės, kad užsidarys nuo 1 iki 6 koneksinų, yra atitinkamai lygios apie 0,4 (kai įtampa yra 40 mV) ir 0,2 (kai įtampa yra 90 mV). Tikimybės, kad užsidarys nuo 7 iki 11 koneksinų (taip pat ir aukščiau minėtų 12 koneksinų), yra nedidelė, esant 90-100 mV įtampai.

3.1.3. PJK DLMG 6 KONEKSINŲ 3 BŪSENŲ MODELIO TYRIMAS

Tiriant šį modelį naudojamasi taip pat 3.1 lentele ir gauti tokie Markovo modelio rezultatai:

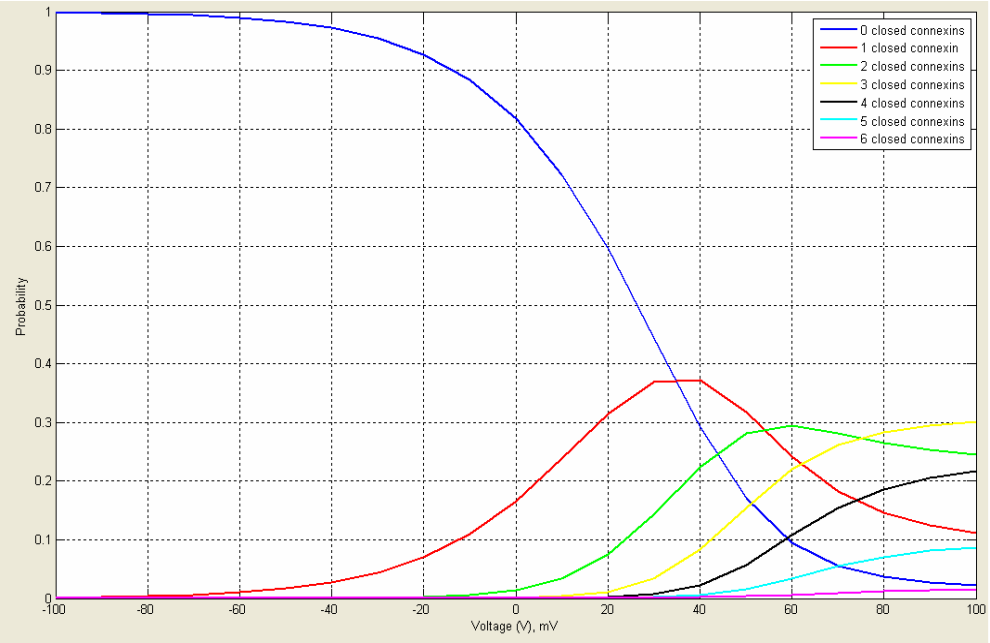
3.4 lentelė. Simuliacijos ir diskretaus laiko Markovo 6 koneksinų 3 būsenų modelių rezultatai, naudojantis parametrais, pasirinktais 3.1 lentelėje.

Įtampa, mV	Laidis, pS		Santykinė paklaida (%)
	Simuliacijos rezultatai (t=1000 ms)	Markovo modelio rezultatai	
-100	5,0234	5,6358	10,87
-80	5,0339	5,7056	11,77
-60	5,0645	5,7742	12,29
-40	5,0951	5,8383	12,73
-20	5,1251	5,8888	12,97
0	5,1558	5,9020	12,64
20	5,1859	5,8186	10,87
40	5,2160	5,5183	5,48
60	5,2460	5,0111	4,69
80	4,0077	4,7033	14,79
100	4,0301	4,6364	13,08

Trijų būsenų modelio parametruose (žr. 3.1 lentelę) įvedamos tikimybių reikšmės $p_{cdc} = 0{,}1, p_{dcc} = 0{,}1$, kurios nepriklauso nuo įtampos (programoje vartotojas pats pasirenka, kokią reikšmę iš intervalo [0;1] įvesti).

Matyti, kad šio modelio rezultatai skiriasi daugiau kaip 10 % (maksimali santykinė paklaida yra ~15 %), kaip 2 būsenų modelių. Todėl 3 būsenų modeliams vertinti negalima pasakyti, ar Markovo modelis tinkamas.

Tikimybės priklausomybė nuo įtampos panaši į 6 koneksinų 2 būsenų modelio (žr. 3.1 paveikslą):



3.4 pav. 6 koneksinų 3 būsenų stacionariųjų tikimybių priklausomybės nuo įtampos V grafikas

Lyginant 3.4 paveikslą su 3.1 paveikslu matyti, kad tikimybė, kad užsidarys 6 koneksinai („6 closed connexins“ (rožinė linija)), yra nedidelė (artima nuliui), o tikimybė, kad neužsidarys nė vienas, t.y. kad bus 6 atviri koneksinai, įtampai didėjant mažėja nuo 1 iki ~0. Kai įtampa teigiama, tai užsidarančių nuo 1 iki 5 koneksinų tikimybė yra nuo 0,38 iki 0,1.

3.1.4. PJK DLMG 12 KONEKSINŲ 3 BŪSENŲ MODELIO TYRIMAS

Tiriant šį modelį naudojamasi taip pat 3.1 lentele ir gauti tokie Markovo modelio rezultatai:

3.5 lentelė. Simuliacijos ir diskretaus laiko Markovo 12 koneksinų 3 būsenų modelių rezultatai, naudojantis parametrais, pasirinktais 3.1 lentelėje.

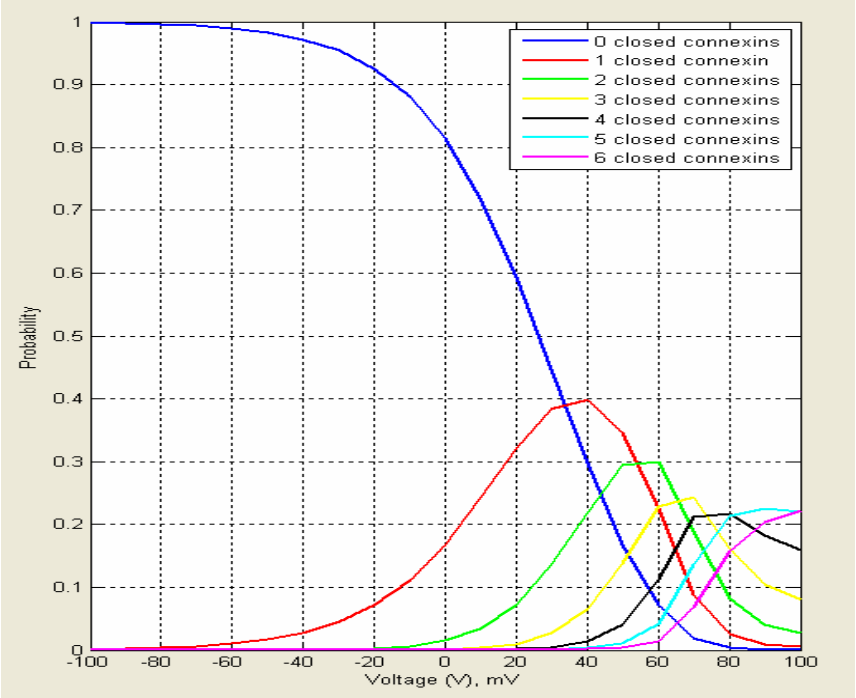
Įtampa, mV	**Laidis, pS**		**Santykinė paklaida (%)**
	Simuliacijos rezultatai (t=1000 ms)	**Markovo modelio rezultatai**	
-100	5,0234	5,2840	4,93
-80	5,0339	5,3849	6,52
-60	5,0645	5,4856	7,68
-40	5,0951	5,5826	8,73
-20	5,1251	5,6667	9,56

0	5,1558	5,7145	9,78
20	5,1859	5,6747	8,61
40	5,2160	5,4733	4,7
60	5,2460	5,0005	4,91
80	4,0077	4,0335	0,64
100	4,0301	3,6913	9,18

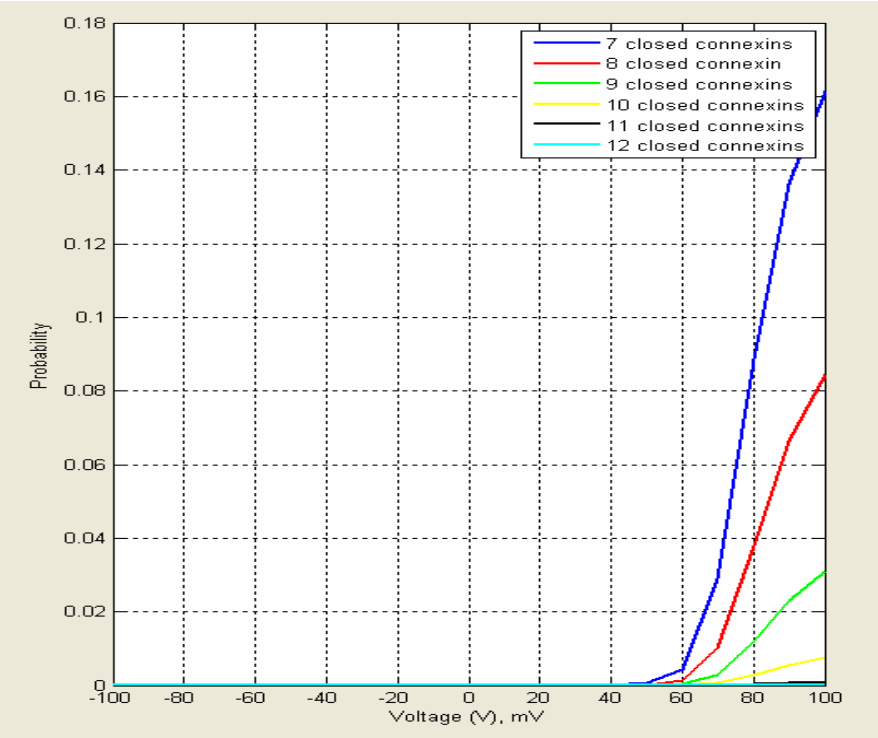
Šio modelio rezultatai jau yra panašesni negu 6 koneksinų 3 būsenų modelio (santykinė paklaida neviršija 10 %). Tai gali būti dėl to, kad kintant abiems puskanaliams gaunamos Markovo modelio reikšmės yra dauginamos, t.y. dauginama kairiojo puskanalio perėjimo tikimybių kiekviena reikšmė iš dešiniojo puskanalio matricos. Todėl tikimybės tikslesnės ir laidžių reikšmės artimesnės simuliacijai.

Modelis yra tikslesnis negu 12 koneksinų 2 būsenų modelis, kurio santykinė paklaida, esant teigiamoms įtampos reikšmėms santykinė paklaida viršija 10 %.

Kadangi šiame modelyje nagrinėjama 12 koneksinų, tai ir nagrinėjama užsidariusių 0-12 koneksinų tikimybių priklausomybė nuo įtampos (žr. 3.5-3.6 paveikslus).



3.5 pav. 12 (nuo 0 iki 6 užsidariusių) koneksinų 3 būsenų tikimybių priklausomybės nuo įtampos grafikas



3.6 pav. 12 (nuo 7 iki 12 užsidariusių) koneksinų 3 būsenų tikimybių priklausomybės nuo įtampos grafikas

Lyginant 3.2-3.3 grafikus su 3.5-3.6 grafikais matyti, kad 3 būsenų grafikuose tikimybė, kad užsidarys 12 koneksinų, lygi 0, o 2 būsenų grafikuose tikimybė, kad užsidarys 12 koneksinų, nėra lygi 0, kai įtampa yra 100 mV. Grafike 3.2 tikimybės, kad užsidarys 2-5 koneksinai, lygios 0, kai įtampa yra 100 mV, o 3.5 paveiksle atitinkamos tikimybės yra lygios ~0,02-0,22. Grafike 3.6 matyti, kad tikimybės, kad užsidarys 7-11 koneksinų, kai įtampa yra 100 mV, yra ~1,5 karto mažesnės už tikimybes, pavaizduotas 3.3 paveiksle. Taigi, 12 koneksinų 3 būsenų modelis kiek tikslesnis, parodantis, kad pridėta trečioji visiškai uždara (angl. *deep-closed*) būsena didina tikimybę užsidaryti koneksinams.

3.2. PJK TLMG MODELIŲ ANALIZĖ

3.2.1. PJK TLMG 6 KONEKSINŲ 2 BŪSENŲ MODELIO TYRIMAS

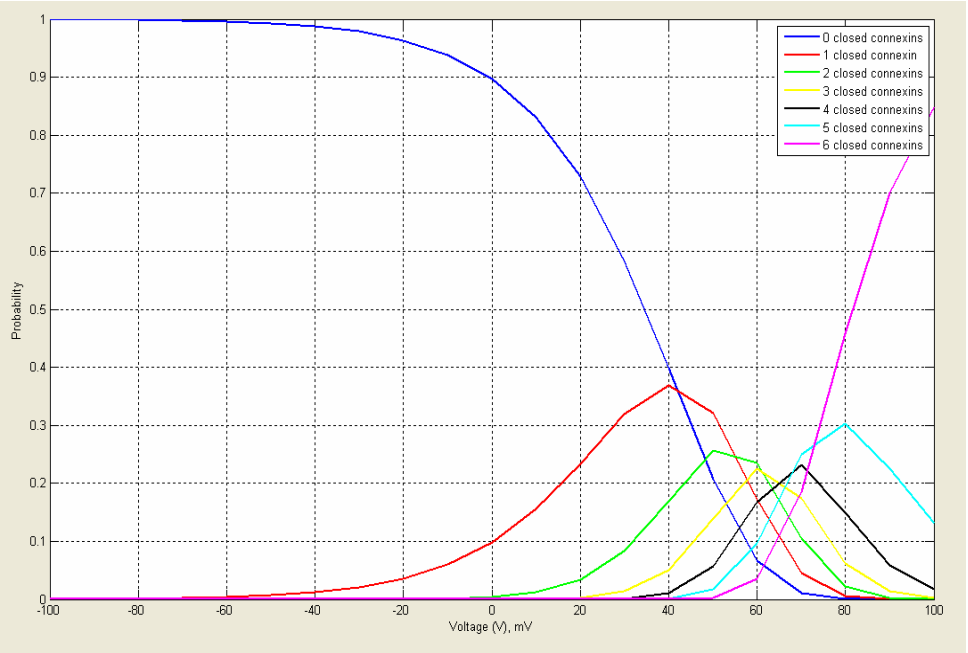
Tiriant šį modelį naudojamasi taip pat 3.1 lentele ir nustatytu koneksino laiku, kuris nurodo, kiek laiko koneksinas būna toje pačioje būsenoje (nustatytas laikas yra 1000), ir gauti tokie Markovo modelio rezultatai:

3.6 lentelė. Simuliacijos ir tolydaus laiko Markovo 6 koneksinų ir 2 būsenų modelių rezultatai, naudojantis parametrais, pasirinktais 3.1 lentelėje.

Įtampa, mV	**Laidis, pS**		**Santykinė paklaida (%)**
	Simuliacijos rezultatai (t=1000 ms)	**Markovo modelio rezultatai**	
-100	5,5736	5,6363	1,11
-80	5,643	5,7067	1,12
-60	5,7131	5,7772	1,11
-40	5,7583	5,8460	1,5
-20	5,8018	5,9081	1,8
0	5,8276	5,9487	2,04
20	5,7916	5,9232	2,22
40	5,2061	5,6797	8,34
60	3,987	4,5948	13,23
80	1,8875	2,6133	27,77
100	1,5811	1,9198	17,64

Remiantis lentele, reikšmės artimos (santykinė paklaida 1-2 procentų), kai įtampos reikšmės yra neigiamos, ir reikšmės nėra artimos, kai įtampos reikšmės teigiamos. Tokiems pokyčiams galėjo daryti įtaką parinkti intensyvumai, aprašyti 2.3.1 skyriuje. Tačiau kokį beparinktume koneksino buvimo toje pačioje būsenoje laiką, reikšmes, gautas Markovo modeliu, gauname vis vien panašias.

Grafikas tikimybių priklausomybės nuo įtampos (žr. 3.7 paveikslą) panašus į grafiką, gautą naudojant diskretaus laiko modelį (žr. 3.1 paveikslą).



3.7 pav. Tolydaus laiko 6 koneksinų 2 būsenų tikimybių priklausomybės nuo įtampos V grafikas

Šio modelio tikimybės, kad užsidarys 2-5 koneksinai, yra didesnės 2 kartus negu pavaizduotos 3.1 paveiksle, tikimybė, kad užsidarys 6 koneksinai, yra 2 kartus mažesnės negu pavaizduotos 3.1 paveiksle. Taigi, tolydaus laiko Markovo grandinės modelio rezultatai nėra adekvatūs tikrinant simuliacijos rezultatus, kai įtampa yra 60-100 mV.

3.2.2. PJK TLMG 12 KONEKSINŲ 2 BŪSENŲ MODELIO TYRIMAS

Tiriant šį modelį naudojamasi taip pat 3.1 lentele ir nustatytu koneksino laiku, kuris nurodo, kiek laiko koneksinas būna toje pačioje būsenoje (nustatytas laikas yra 1000), gauti tokie Markovo modelio rezultatai:

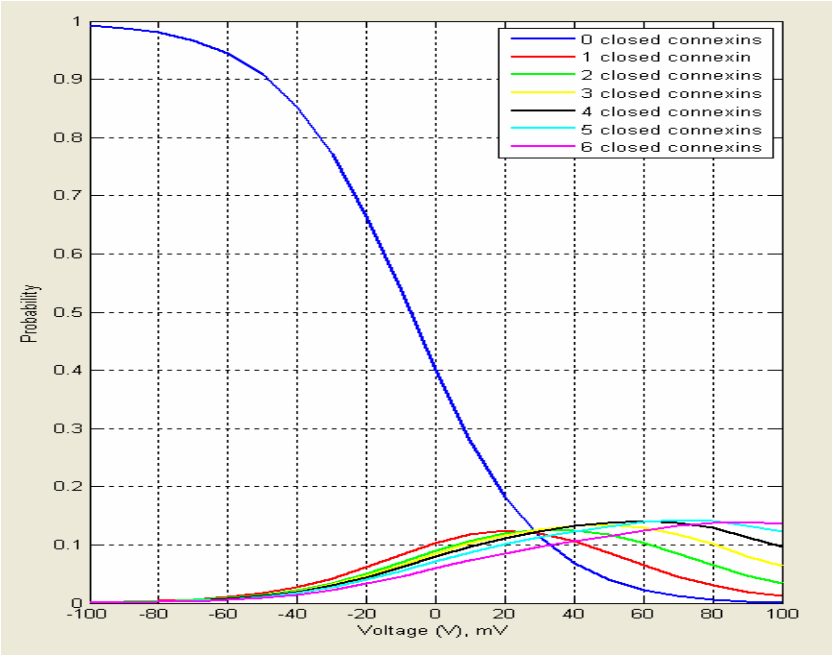
3.7 lentelė. Simuliacijos ir tolydaus laiko Markovo modelių 12 koneksinų 2 būsenų rezultatai, naudojantis parametrais, pasirinktais 3.1 lentelėje.

Įtampa, mV	**Laidis, pS**		**Santykinė paklaida (%)**
	Simuliacijos rezultatai (t=1000 ms)	**Markovo modelio rezultatai**	
-100	5,1712	5,2720	1,91
-80	5,1778	5,3495	3,21
-60	5,1930	5,3841	3,55

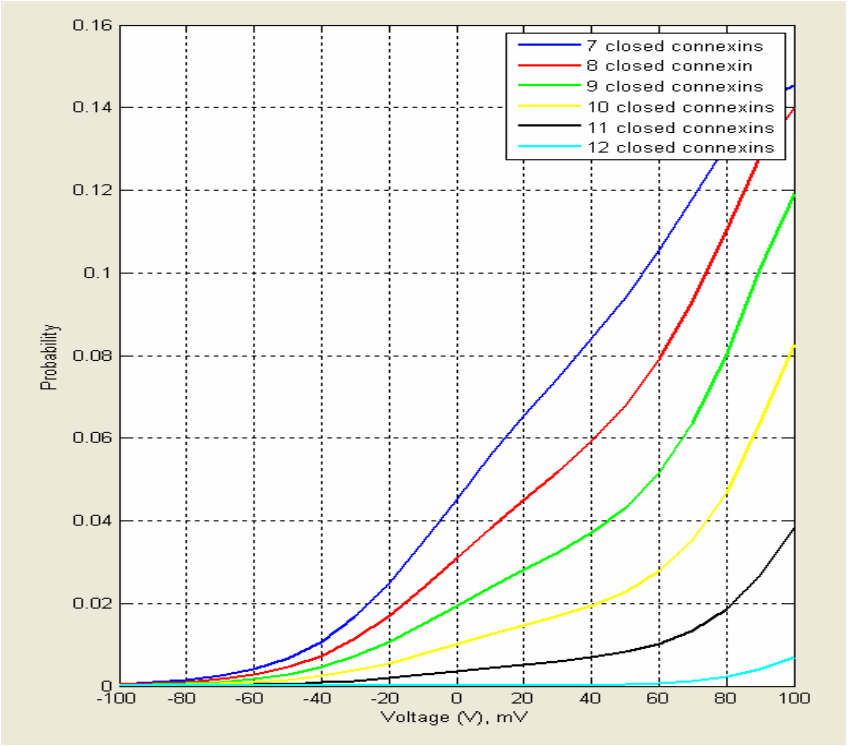
-40	5,2082	5,3102	1,92
-20	5,2232	5,0385	3,67
0	4,9405	4,5983	7,44
20	4,9600	4,1967	18,19
40	4,9796	3,9148	27,2
60	4,9988	3,6609	36,55
80	4,7245	3,3395	41,47
100	3,7444	3,0131	24,27

Remiantis lentele, reikšmės panašios (santykinė paklaida 1-4 %), kai įtampos reikšmės neigiamos, ir reikšmės skiriasi (santykinė paklaida > 10 %), kai įtampos reikšmės teigiamos. Tačiau kokį beparinktume koneksino buvimo toje pačioje būsenoje laiką (žr. 3.2.1 skyrių), reikšmes, gautas Markovo modeliu, gauname vis vien panašias.

Lyginant 12 koneksinų 2 būsenų diskretaus laiko Markovo grandinės modelio grafiką (žr. 3.2-3.3 paveikslus) su tolydaus laiko Markovo grandinės modelio grafiku (žr. 3.8-3.9 paveikslus) matyti, kad tolydaus laiko Markovo modelio tikimybės, kad užsidarys 0-12 koneksinų, 2 kartus mažesnės už diskretaus laiko modelio tikimybes (žr. 3.2-3.3 paveikslus).



3.8 pav. Tolydaus laiko 12 (nuo 0 iki 6 užsidariusių) koneksinų 2 būsenų tikimybių priklausomybės nuo įtampos grafikas



3.9 pav. Tolydaus laiko 12 (nuo 7 iki 12 užsidariusių) koneksinų 2 būsenų tikimybių priklausomybės nuo įtampos grafikas

4. PROGRAMINĖ REALIZACIJA IR INSTRUKCIJA VARTOTOJUI

Programa kurta Matlab R2010a. Pasirinkta ši programa dėl to, kad atliekami veiksmai su matricomis ir Matlab programoje kai kurios procedūros, kaip antai: lygčių sistemų sprendimas taip pat sprendžiamas matricine forma.

Sukurta grafinė vartotojo sąsaja, kurios instrukcija aprašoma šiame skyriuje.

Grafinė vartotojo sąsaja (angl. *Graphical User Interface*) veikia Windows operacinėje sistemoje (sukurtas paleidžiamasis (angl. *executable (.exe)*) failas). Programa paleidžiama iš failo „Program_GJM.exe“. Ši programa į operacinę sistemą nėra diegiama, o tiesiog iš karto paleidžiama, tačiau reikalingas Matlab kompiliatorius „Matlab Compiler Runtime“, kurį reikia įdiegti į operacinę sistemą prieš paleidžiant failą „Program_GJM.exe“. Šis kompiliatorius („MCRInstaller.exe“) įrašytas kompaktiniame diskelyje, prisegtame prie šio magistrinio darbo.

Programos langas buvo parodytas skyriuje 1.2. (žr. 1.6 paveikslą).

Paleidus programą pirmiausia reikia nustatyti norimas parametrų reikšmes ir tuomet pereiti prie meniu skilties „File“.

Programos lange yra dvi meniu skiltys: „File“ ir „About“. Meniu skiltyje „File“ galima pasirinkti „Discrete Time Markov Model“ – tai diskretaus laiko Markovo modelis. Iš šios skilties reikia pasirinkti 2 arba 3 būsenų modelius („2 states“ arba „3 states“). Pasirinkus kažkurią iš 2 skilčių reikia pasirinkti arba 6 koneksinų arba 12 koneksinų modelius („6 connexins“ arba „12 connexins“).
„Continuous Time Markov Model“ – tai tolydaus laiko Markovo modelis. Iš šios skilties reikia pasirinkti 2 („2 states“) būsenų modelius. Tuomet galima rintis arba 6 arba 12 koneksinų modelius („6 connexins“ arba „12 connexins“).

Skiltyje „About“ trumpai parašyta apie programą ir įrašyta programą kūrusio studento vardas, pavardė ir grupė.

Nustačius pasirinktas parametrų reikšmes ir pasirinkus vieną iš modelių programos lange vaizduojami tokie grafikai:

1) jei tai 6 koneksinų modelis, tai bus vaizduojama kairiojo puskanalio laidžio priklausomybė nuo būsenos; jei tai 12 koneksinų modelis, tai bus vaizduojama abiejų kanalų laidžio priklausomybė nuo būsenos.
2) laidžio priklausomybės nuo pasirinkto įtampos intervalo grafikas.

IŠVADOS

Magistriniame darbe pateikiama ląstelių plyšinės jungties Markovo modelių metodika, leidžianti skaičiuoti plyšinės jungties laidumo priklausomybę nuo įtampos. Markovo grandinių perėjimo tikimybių skaičiavimams naudojama nepriklausomų J. Bernulio bandymų schema. Sukurta plyšinės jungties Markovo modelių metodika, kai koneksinas aprašomas trimis būsenomis.

Lyginant Markovo modeliais gautus rezultatus su imitacinio modeliavimo rezultatais parodyta, kad abiejų modeliavimų rezultatai skiriasi ne daugiau kaip 10 %, tačiau tokia nedidelė paklaida buvo ne visų modelių. Pateikiami pavadinimai keturių modelių, kurių rezultatai neviršija 10 %:

1. diskretaus laiko 6 koneksinų 2 būsenų Markovo modelio;
2. diskretaus laiko 12 koneksinų 3 būsenų Markovo modelio;
3. tolydaus laiko 6 koneksinų 2 būsenų Markovo modelio (iki 60 mV, t.y. įtampos intervale $\left[-100;60\right)$ mV);
4. tolydaus laiko 12 koneksinų 2 būsenų Markovo modelio (kai įtampa yra neigiama, santykinė paklaida neviršija 8 %).

Likusių modelių (taip pat keturių) rezultatų santykinė paklaida viršija nuo 15 iki 41 %:

1. diskretaus laiko 12 koneksinų 2 būsenų Markovo modelio rezultatų santykinė paklaida yra iki 18 %;
2. diskretaus laiko 6 koneksinų 3 būsenų Markovo modelio rezultatų santykinė paklaida yra iki 15 %;
3. tolydaus laiko 6 koneksinų 2 būsenų Markovo modelio, kai įtampos intervalas $\left[60;100\right]$ mV, rezultatų santykinė paklaida yra iki 28 %;
4. tolydaus laiko 12 koneksinų 2 būsenų Markovo modelio rezultatų santykinė paklaida yra nuo 18 % iki 41 %.

Taigi, simuliacijos rezultatams adekvatūs rezultatai šių modelių: diskretaus laiko 6 koneksinų 2 būsenų ir 3 būsenų Markovo modelių ir diskretaus laiko 12 koneksinų 2 ir 3 būsenų Markovo modelių (jeigu santykinė paklaida yra iki 20 %). Tačiau tolydaus laiko Markovo modelių rezultatai nėra tikslūs (Markovo modelių ir simuliacijos rezultatų rezultatų reikšmių skirtumai didinant įtampą didėja), todėl nėra tinkami (todėl ir nebuvo kuriami tolydaus laiko 3 būsenų modeliai).

Tyrimus magistrinio darbo tematika numatoma tęsti, kai koneksinas aprašomas 3 būsenomis. Taip pat numatoma sukurtus plyšinės jungties Markovo modelius panaudoti parametrų optimizavimui bei modeliuojant elektrinių signalų perdavimą ląstelių tinkluose.

LITERATŪRA

1. The electrophysiology of gap junctions and gap junction channels and their mathematical modelling. Rolf Vogel, Robert Weingart. Biology of the Cell, No. 94, p. 501-510, 2002.

2. Mathematical model of vertebrate gap junctions derived from electrical measurements on homotypic and heterotypic channels. Rolf Vogel and Robert Weingart. Journal of Physiology, No. 510.1, p. 177-189, 1998 (žiūrėta 2011-04-28). Internetinė nuoroda: http://jp.physoc.org/content/510/1/177.full.pdf.

3. Opposing gates model for voltage gating of gap junction channels. Ye Chen-Izu, Alonso P. Moreno, Robert A. Spangler. Am J Physiol Cell Physiol, No. 281, p. C1604-C1613, 2001.

4. Subconductance States of Cx30 Gap Junction Channels: Data from Transfected HeLa Cells versus Data from a Mathematical Model. Rolf Vogel, Virginijus Valiūnas, Robert Weingart. Biophysical Journal, Volume 91, p. 2337-2348, 2006.

5. Markovian Model of the Voltage Gating of Connexin-based Gap Junction Channels. Aurelija Sakalauskaitė, Henrikas Pranevičius, Feliksas Bukauskas, Mindaugas Pranevičius. Electronics and Electrotechnics, No. 5 (111), p. 103-106, 2011. Internetinė nuoroda: http://www.ee.ktu.lt/journal/2011/05/24__ISSN_1392-1215_Markovian%20Model%20of%20the%20Voltage%20Gating%20of%20Connexin.pdf.

6. Rinktiniai procesų teorijos skyriai. Eugenijus Manstavičius, 2007, p. 61 (žiūrėta 2009-05-25). Internetinė nuoroda: http://www.mif.vu.lt/katedros/ttsk/bylos/man/files/konsp-2006.pdf.

7. Algimantas Aksomaitis. Tikimybių teorija ir statistika. Kaunas, Technologija, 2002, p. 46-47.

8. Algimantas Aksomaitis. Tikimybių teorija ir statistika. Kaunas, Technologija, 2002, p. 18.

9. A Stochastic Four-State Model of Contingent Gating of Gap Junction. Nerijus Paulauskas, Mindaugas Pranevičius, Henrikas Pranevičius, Feliksas Bukauskas. Biophysical Journal, Elsevier, No. 96, p. 3936–3948, 2009.

1 PRIEDAS. PROGRAMOS KODAS (MATLAB)

```
function program_GJM
% Graphical User Interface
figure_handle = figure('Visible','off','Position',[200,300,1000,700]);
% Menu
menu_handle=uimenu(figure_handle,'Label','File');
menu_File_handle_1=uimenu(menu_handle,'Label','Discrete Time Markov Model');
submenu_File_handle_11=uimenu(menu_File_handle_1,'Label','2 states');
submenu_File_handle_111=uimenu(submenu_File_handle_11,'Label','6
connexins','Callback',@six_connexins_button_Callback);
submenu_File_handle_112=uimenu(submenu_File_handle_11,'Label','12
connexins','Callback',@twelve_connexins_button_Callback);
submenu_File_handle_12=uimenu(menu_File_handle_1,'Label','3 states');
submenu_File_handle_121=uimenu(submenu_File_handle_12,'Label','6
connexins','Callback',@six_connexins_button2_Callback);
submenu_File_handle_122=uimenu(submenu_File_handle_12,'Label','12
connexins','Callback',@twelve_connexins_button2_Callback);
menu_File_handle_2=uimenu(menu_handle,'Label','Continuous Time Markov Model');
submenu_File_handle_21=uimenu(menu_File_handle_2,'Label','2 states');
submenu_File_handle_211=uimenu(submenu_File_handle_21,'Label','6
connexins','Callback',@six_connexins_continuous_button_Callback);
submenu_File_handle_212=uimenu(submenu_File_handle_21,'Label','12
connexins','Callback',@twelve_connexins_continuous_button_Callback);

menu_File_handle_3=uimenu(menu_handle,'Label','Exit','Callback',@exitbutton_Callba
ck);
set(menu_File_handle_3,'Separator','on');
menu_handle_2=uimenu(figure_handle,'Label','About','Callback',@aboutbutton_Callbac
k);
% end of Menu

description1_text=uicontrol('Style','text','String','Left
Hemichannel','Position',[15,680,100,15]);
A1_text = uicontrol('Style','text','String','A = ','Position',[15,660,60,15]);
P1_text = uicontrol('Style','text','String','P = ','Position',[15,640,60,15]);
V01_text = uicontrol('Style','text','String','V0 = ','Position',[15,620,60,15]);
K1_text = uicontrol('Style','text','String','K = ','Position',[15,600,60,15]);
time1_text = uicontrol('Style','text','String','Time =
','Position',[15,580,60,15]);
pcdc1_text = uicontrol('Style','text','String','pcdc =
','Position',[15,560,60,15]);
pdcc1_text = uicontrol('Style','text','String','pdcc =
','Position',[15,540,60,15]);

description2_text=uicontrol('Style','text','String','Right
Hemichannel','Position',[15,510,120,20]);
A2_text = uicontrol('Style','text','String','A = ','Position',[15,490,60,15]);
P2_text = uicontrol('Style','text','String','P = ','Position',[15,470,60,15]);
V02_text = uicontrol('Style','text','String','V0 = ','Position',[15,450,60,15]);
K2_text = uicontrol('Style','text','String','K = ','Position',[15,430,60,15]);
time2_text = uicontrol('Style','text','String','Time =
','Position',[15,410,60,15]);
pcdc2_text = uicontrol('Style','text','String','pcdc =
','Position',[15,390,60,15]);
pdcc2_text = uicontrol('Style','text','String','pdcc =
','Position',[15,370,60,15]);

V_text = uicontrol('Style','text','String','V = ','Position',[15,340,60,15]);
V_min_text = uicontrol('Style','text','String','V min =
','Position',[15,320,60,15]);
```

```
V_step_text = uicontrol('Style','text','String','V step =
','Position',[15,300,60,15]);
V_max_text = uicontrol('Style','text','String','V_max =
','Position',[15,280,60,15]);
Channels_text =
uicontrol('Style','text','String','Channels','Position',[15,260,60,15]);

A1_handle = uicontrol('Style','edit','String','0.1','Position',[80,660,60,15]);
P1_handle = uicontrol('Style','edit','String','1','Position',[80,640,60,15]);
V01_handle = uicontrol('Style','edit','String','40','Position',[80,620,60,15]);
K1_handle = uicontrol('Style','edit','String','0.1','Position',[80,600,60,15]);
time1_handle =
uicontrol('Style','edit','String','1000','Position',[80,580,60,15]);
pcdc1_handle = uicontrol('Style','edit','String','0.1','Position',[80,560,60,15]);
pdcc1_handle = uicontrol('Style','edit','String','0.1','Position',[80,540,60,15]);
A2_handle = uicontrol('Style','edit','String','0.1','Position',[80,490,60,15]);
P2_handle = uicontrol('Style','edit','String','1','Position',[80,470,60,15]);
V02_handle = uicontrol('Style','edit','String','40','Position',[80,450,60,15]);
K2_handle = uicontrol('Style','edit','String','0.1','Position',[80,430,60,15]);
time2_handle =
uicontrol('Style','edit','String','1000','Position',[80,410,60,15]);
pcdc2_handle = uicontrol('Style','edit','String','0.1','Position',[80,390,60,15]);
pdcc2_handle = uicontrol('Style','edit','String','0.1','Position',[80,370,60,15]);

V_handle = uicontrol('Style','edit','String','100','Position',[80,340,60,15]);
V_min_handle = uicontrol('Style','edit','String','-
100','Position',[80,320,60,15]);
V_step_handle = uicontrol('Style','edit','String','10','Position',[80,300,60,15]);
V_max_handle = uicontrol('Style','edit','String','100','Position',[80,280,60,15]);
Channels_handle =
uicontrol('Style','edit','String','10','Position',[80,260,60,15]);

A1_text_2 = uicontrol('Style','text','String','1/mV','Position',[145,660,60,15]);
P1_text_2 =
uicontrol('Style','text','String','const.','Position',[145,640,60,15]);
V01_text_2 = uicontrol('Style','text','String','mV','Position',[145,620,60,15]);
K1_text_2 =
uicontrol('Style','text','String','const.','Position',[145,600,60,15]);
time1_text_2 = uicontrol('Style','text','String','ms','Position',[145,580,60,15]);
pcdc1_text_2 =
uicontrol('Style','text','String','const.','Position',[145,560,60,15]);
pdcc1_text_2 =
uicontrol('Style','text','String','const.','Position',[145,540,60,15]);
A2_text_2 = uicontrol('Style','text','String','1/mV','Position',[145,490,60,15]);
P2_text_2 =
uicontrol('Style','text','String','const.','Position',[145,470,60,15]);
V02_text_2 = uicontrol('Style','text','String','mV','Position',[145,450,60,15]);
K2_text_2 =
uicontrol('Style','text','String','const.','Position',[145,430,60,15]);
time2_text_2 = uicontrol('Style','text','String','ms','Position',[145,410,60,15]);
pcdc2_text_2 =
uicontrol('Style','text','String','const.','Position',[145,390,60,15]);
pdcc2_text_2 =
uicontrol('Style','text','String','const.','Position',[145,370,60,15]);

V_text_2 = uicontrol('Style','text','String','mV','Position',[145,340,60,15]);
V_min_text_2 = uicontrol('Style','text','String','mV','Position',[145,320,60,15]);
V_step_text_2 =
uicontrol('Style','text','String','mV','Position',[145,300,60,15]);
V_max_text_2 = uicontrol('Style','text','String','mV','Position',[145,280,60,15]);
Channels_text_2 =
uicontrol('Style','text','String','const.','Position',[145,260,60,15]);
```

```
table_initial=[[0:6]' zeros(7,2)];
table_initial_handle = uitable('Position',
[220,440,272,170],'Data',table_initial,...
                            'ColumnName', {'State','Voltage','Conductance'},...
                            'ColumnFormat', {'numeric','numeric','numeric'},...
                            'ColumnWidth', {80 80 80});
bar_initial_handle =
bar(axes('Units','Pixels','Position',[560,350,300,300]),table_initial(:,1),table_i
nitial(:,3));
title 'Bar of a conductance of a channel (depending on a state)';
xlabel('a state of a left hemichannel, const.')
ylabel('Conductance of a channel, pS')
axis([0 7 0 6]);
set(table_initial_handle,'FontName','Courier','FontSize',12);
set(bar_initial_handle);
table_initial_2=zeros(10,2);
table_initial_handle_2 = uitable('Position',
[220,50,210,260],'Data',table_initial_2,...
                         'ColumnName', {'Voltage','Conductance'},...
                         'ColumnFormat', {'numeric','numeric'},...
                         'ColumnWidth', {80 80});
plot_initial_handle =
plot(axes('Units','Pixels','Position',[560,60,300,220]),table_initial_2(:,1),table
_initial_2(:,2),'LineWidth',2);
set(table_initial_handle_2,'FontName','Courier','FontSize',12);
title 'Conductance dependence on voltage';
xlabel('V, mV')
ylabel('G(V), pS')
axis([-100 100 0 7]);
set(plot_initial_handle);
grid on;
set([description1_text,A1_text,P1_text,V01_text,K1_text,time1_text,pcdc1_text,pdcc
1_text,...

description2_text,A2_text,P2_text,V02_text,K2_text,time2_text,pcdc2_text,pdcc2_tex
t,...
    V_text,V_min_text,V_step_text,V_max_text,Channels_text,...

A1_text_2,P1_text_2,V01_text_2,K1_text_2,time1_text_2,pcdc1_text_2,pdcc1_text_2,..
.

A2_text_2,P2_text_2,V02_text_2,K2_text_2,time2_text_2,pcdc2_text_2,pdcc2_text_2,..
.
    V_text_2,V_min_text_2,V_step_text_2,V_max_text_2,Channels_text_2],...
    'FontName','default','FontSize',10);
set([A1_handle,P1_handle,V01_handle,K1_handle,time1_handle,pcdc1_handle,pdcc1_hand
le,...

A2_handle,P2_handle,V02_handle,K2_handle,time2_handle,pcdc2_handle,pdcc2_handle,..
.

V_handle,V_min_handle,V_step_handle,V_max_handle,Channels_handle],'FontName','Cour
ier','BackgroundColor','white');
set(figure_handle,'Units','normalized','NumberTitle', 'off','Menubar','none');
defaultBackground = get(0,'defaultUicontrolBackgroundColor');
set(figure_handle,'Name','GJM','Color',defaultBackground),
movegui(figure_handle,'center')
set(figure_handle,'Visible','on');

% methods that are the same in every model
function g_open = g_open_connexin(P,voltage)
  g_open = 2*exp(P*voltage/800);
end
function g_closed = g_closed_connexin(P,voltage)
```

```
  g_closed = 0.25*exp(P*voltage/300);
end
function g_left = g_left_hemichannel(P,voltage,n)
  g_left=(6-(n-1))*g_open_connexin(P,voltage)+(n-1)*g_closed_connexin(P,voltage);
end
function g_right = g_right_hemichannel(P,voltage,n)
  g_right=(6-(n-1))*g_open_connexin(P,voltage)+(n-1)*g_closed_connexin(P,voltage);
end
function k_value=k(A,P,V0,voltage)
  k_value=exp(A*(P*voltage-V0));
end
function poc_value=poc(A,P,V0,K,voltage)
  poc_value=K*k(A,P,V0,voltage)/(1+k(A,P,V0,voltage));
end
function poo_value=poo(A,P,V0,K,voltage)
  poo_value=1-poc(A,P,V0,K,voltage);
end
 function pco_value=pco(A,P,V0,K,voltage)
  pco_value=K/(1+k(A,P,V0,voltage));
 end
 function pcc_value=pcc(A,P,V0,K,voltage)
  pcc_value=1-pco(A,P,V0,K,voltage);
 end
% for model of 6 connexins only
 function voltage_s=voltage_left_stationary(P1,P2,V)
     for n=1:1:7

V_left_hemichannel(1,n)=(V*g_right_hemichannel(P2,0,1))/(g_left_hemichannel(P1,0,n
)+g_right_hemichannel(P2,0,1));
    end
    for t=2:1:5
       for n=1:1:7
       V_left_hemichannel(t,n)=(V*g_right_hemichannel(P2,V-V_left_hemichannel(t-
1,n),1))/...
      (g_left_hemichannel(P1,V_left_hemichannel(t-
1,n),n)+g_right_hemichannel(P2,V-V_left_hemichannel(t-1,n),1));
       end
    end
    voltage_s=V_left_hemichannel(5,:)';
 end

% for Continuous Time Models
function lambda=lambda_oc(A,P,V0,K,voltage,t)
  lambda=poc(A,P,V0,K,voltage)/t;
 end
 function lambda=lambda_co(A,P,V0,K,voltage,t)
  lambda=pco(A,P,V0,K,voltage)/t;
 end
% end of for Continuous Time Models

% for 3 States Models
function g_left = g_left_hemichannel_3(P,voltage,n1,n2,n3)

g_left=n1*g_closed_connexin(P,voltage)+n2*g_closed_connexin(P,voltage)+n3*g_open_c
onnexin(P,voltage);
end
function g_right = g_right_hemichannel_3(P,voltage,n1,n2,n3)

g_right=n1*g_closed_connexin(P,voltage)+n2*g_closed_connexin(P,voltage)+n3*g_open_
connexin(P,voltage);
end

 function pco_value_new=pco_new(A,P,V0,K,pcdc,voltage)
```

```
  pco_value_new=pco(A,P,V0,K,voltage)*(1-pcdc);
 end
 function pcc_value_new=pcc_new(A,P,V0,K,pcdc,voltage)
  pcc_value_new=pcc(A,P,V0,K,voltage)*(1-pcdc);
 end
% end of for 3 States Models

function stationary = stationary_probabilities(A)
% Compute the stationary distribution of an ergodic Markov chain
% caracterized by its transition probabilities matrix A such that
% A = Pr(x_{k} | x_{k - 1}) , sum(A , 2) = 1.
%
% PI = stationary(A);
%
[V,D]=eig(A');
[foo,tp]=sort(diag(D));
stationary=V(:,tp(end))/sum(V(:,tp(end)));
end

% 2 states models
function six_connexins_button_Callback(source,data)
% 6 connexins model
% variables
A1_variable = str2num(get(A1_handle,'String'));
P1_variable = str2num(get(P1_handle,'String'));
V01_variable = str2num(get(V01_handle,'String'));
K1_variable = str2num(get(K1_handle,'String'));
A2_variable = str2num(get(A2_handle,'String'));
P2_variable = str2num(get(P2_handle,'String'));
V02_variable = str2num(get(V02_handle,'String'));
K2_variable = str2num(get(K2_handle,'String'));

V_variable = str2num(get(V_handle,'String'));
V_min_variable = str2num(get(V_min_handle,'String'));
V_step_variable = str2num(get(V_step_handle,'String'));
V_max_variable = str2num(get(V_max_handle,'String'));
number_of_channels=str2num(get(Channels_handle,'String'));
% end of variables
if P1_variable < -1 || (P1_variable >-1 && P1_variable < 1) || P1_variable >1
||...
   P2_variable < -1 || (P2_variable >-1 && P2_variable < 1) || P2_variable >1
 message_handle=msgbox('P1/P2 has to be +1 or -1!','Error');
 set(message_handle,'Units','normalized','Color',defaultBackground);
end
if (V_max_variable<V_min_variable) || (V_step_variable > V_max_variable) ||...
   (V_max_variable<V_min_variable) && (V_step_variable > V_max_variable)
 message_handle=msgbox('V max is smaller than V min or V step is greater than V
max.','Error');
 set(message_handle,'Units','normalized','Color',defaultBackground);
end
if round(number_of_channels) ~= number_of_channels || number_of_channels<=0
 message_handle=msgbox('Number of channels must be positive integer!','Error');
 set(message_handle,'Units','normalized','Color',defaultBackground);
end
%      formation of a matrix
     function matrix=formationOfMatrix(A,P,V0,K,voltage_steady)
       states=zeros(7,2);
       for n=1:1:7
        states(n,1)=n-1; states(n,2)=6-(n-1);
       end
       function Cartesian=setCartesian(states_1,states_2)
       one=sortrows(repmat(states_1,[size(states_2,1) 1]));
       second=repmat(states_2,[size(states_1,1) 1]);
```

```
        Cartesian=[one second];
        end
        matrix=zeros(7);
        o=1;
        while o<8
             states1=zeros(states(o,1)+1,2);
             for l=1:1:states(o,1)+1
                states1(l,1)=states(o,1)-(l-1);
                states1(l,2)=l-1;
             end
             states2=zeros(states(o,2)+1,2);
             for l=1:1:states(o,2)+1
                states2(l,1)=l-1;
                states2(l,2)=states(o,2)-(l-1);
             end
             Cartesian_product=setCartesian(states1,states2);
             sum_1=sum(Cartesian_product(:,1:2:3),2);
             sum_2=sum(Cartesian_product(:,2:2:4),2);
             indices=[Cartesian_product sum_1 sum_2];
             number=size(indices,1);
             probabilities=zeros(number,1);
             for number1=1:1:number
                probabilities(number1)=...
                nchoosek(o-
1,indices(number1,1))*pcc(A,P,V0,K,voltage_steady(o))^(indices(number1,1))*...

pco(A,P,V0,K,voltage_steady(o))^(indices(number1,2))*...
                nchoosek(6-(o-
1),indices(number1,3))*poc(A,P,V0,K,voltage_steady(o))^(indices(number1,3))*...

poo(A,P,V0,K,voltage_steady(o))^(indices(number1,4));

             end
             probabilities2=zeros(7,1);
             for number1=1:1:number
                 for l=1:1:7
                    if indices(number1,5)==(l-1)
                      probabilities2(l)=probabilities2(l)+probabilities(number1);
                    end
                 end
             end
             for j=1:1:7
               matrix(o,j)=probabilities2(j);
             end
             o=o+1;
          end
      end
  V=voltage_left_stationary(P1_variable,P2_variable,V_variable);
  matrix_a=formationOfMatrix(A1_variable,P1_variable,V01_variable,K1_variable,V);

    function normalised=normalised_matrix(matrix)
        normalised=zeros(size(matrix,1));
        sum_of_p_rows=sum(matrix,2);
        for i=1:1:length(sum_of_p_rows)
         normalised(i,:)=matrix(i,:)/sum_of_p_rows(i);
        end
    end
  matrix=normalised_matrix(matrix_a);
  PI=stationary_probabilities(matrix);
  A=[matrix' - eye(size(matrix,1)); ones(1,size(matrix,1))];
  B=[zeros(size(matrix,1),1);1];
  X=A\B;
 initial_distribution=[1 0 0 0 0 0 0];
```

```
 p(1,:)=initial_distribution(1,:);
 for i=1:1:200
    p(i+1,:)=p(1,:)*(matrix^i);
 end
 conductance=zeros(7,1);
 for n=1:1:7

conductance(n)=number_of_channels*X(n)*(g_left_hemichannel(P1_variable,V(n),n)*g_r
ight_hemichannel(P2_variable,V_variable-V(n),1))/...

(g_left_hemichannel(P1_variable,V(n),n)+g_right_hemichannel(P2_variable,V_variable
-V(n),1));
 end
 n=0:1:6; n=n';
 table = [n V conductance];
% % % % For drawing the graphic conductance(V), when V_variable change
% values from the specified interval
    min=V_min_variable; max=V_max_variable; step=V_step_variable;
    count_voltages=numel(min:step:max);
      V_left_hemichannel2=zeros(5,7);
      conductance_interval=zeros(count_voltages,1);
      index=1;
      for V_variable_variates=min:step:max
        for n=1:1:7

V_left_hemichannel2(1,n)=(V_variable_variates*g_right_hemichannel(P2_variable,0,1)
)/...

(g_left_hemichannel(P1_variable,0,n)+g_right_hemichannel(P2_variable,0,1));
        end

        for t=2:1:5
           for n=1:1:7

V_left_hemichannel2(t,n)=(V_variable_variates*g_right_hemichannel(P2_variable,V_va
riable_variates-V_left_hemichannel2(t-1,n),1))/...
            (g_left_hemichannel(P1_variable,V_left_hemichannel2(t-
1,n),n)+g_right_hemichannel(P2_variable,V_variable_variates-V_left_hemichannel2(t-
1,n),1));
           end
        end
       V2=V_left_hemichannel2(5,:)';

matrix2=formationOfMatrix(A1_variable,P1_variable,V01_variable,K1_variable,V2);
      X2=[matrix2' - eye(size(matrix2)); ones(1,length(matrix2))] \...
          [zeros(length(matrix2),1);1];
      statmatrix(:,index)=abs(X2);
     conductance2=zeros(7,1);
     for n=1:1:7

conductance2(n)=number_of_channels*X2(n)*(g_left_hemichannel(P1_variable,V2(n),n)*
g_right_hemichannel(P2_variable,V_variable_variates-V2(n),1))/...

(g_left_hemichannel(P1_variable,V2(n),n)+g_right_hemichannel(P2_variable,V_variabl
e_variates-V2(n),1));
     end
     conductance_interval(index)=sum(conductance2);
     index=index+1;
      end

  table_handle = uitable('Position', [220,440,272,170],'Data',table,...
                         'ColumnName', {'State','Voltage','Conductance'},...
                         'ColumnFormat', {'numeric','numeric','numeric'},...
```

```
                        'ColumnWidth', {80 80 80});
  bar_handle =
bar(axes('Units','Pixels','Position',[560,350,300,300]),table(:,1),table(:,3));
  title 'Bar of a conductance of a channel (depending on a state)';
  xlabel('a state of a left hemichannel, const.')
  ylabel('Conductance of a channel, pS')
  axis([0 7 0 6]);
  set(bar_handle);
  set(table_handle,'FontName','Courier','FontSize',12);
  voltage_interval=min:step:max; voltage_interval=voltage_interval';
  table_handle_2 = uitable('Position', [220,50,210,260],'Data',[voltage_interval
conductance_interval],...
                        'ColumnName', {'Voltage','Conductance'},...
                        'ColumnFormat', {'numeric','numeric'},...
                        'ColumnWidth', {80 80});
  plot_handle =
plot(axes('Units','Pixels','Position',[560,60,300,220]),voltage_interval,conductan
ce_interval,'LineWidth',2);
  set(table_handle_2,'FontName','Courier','FontSize',12);
  title 'Conductance dependence on voltage';
  xlabel('V, mV')
  ylabel('G(V), pS')
  axis([-100 100 0 7]);
  set(plot_handle);
  grid on;
   figure_handle_0=figure();
   i=1:1:201;
   plot(i',p(:,1),'b',i',p(:,2),'r',i',p(:,3),'g',i',p(:,4),'y',...
     i',p(:,5),'k',i',p(:,6),'c',i',p(:,7),'m','LineWidth',2);
   legend('p_0','p_1','p_2','p_3','p_4','p_5','p_6');
   grid on;
   set(figure_handle_0,'Units','normalized','NumberTitle',
'off','Menubar','none','Name','Simulation of
Probabilities','Color',defaultBackground);
   figure_handle_1=figure();

plot(voltage_interval,statmatrix(1,:)','b',voltage_interval,statmatrix(2,:)','r',.
..

voltage_interval,statmatrix(3,:)','g',voltage_interval,statmatrix(4,:)','y',...

voltage_interval,statmatrix(5,:)','k',voltage_interval,statmatrix(6,:)','c',...
       voltage_interval,statmatrix(7,:)','m','LineWidth',2);
   xlabel('Voltage (V), mV')
   ylabel('Probability')
   legend('0 closed connexins','1 closed connexin','2 closed connexins',...
          '3 closed connexins','4 closed connexins','5 closed connexins',...
          '6 closed connexins');
   grid on;
   set(figure_handle_1,'Units','normalized','NumberTitle',
'off','Menubar','none','Name','Dependency of
Probabilities','Color',defaultBackground);
end

function twelve_connexins_button_Callback(source,data)
% 12 connexins model
% variables
A1_variable = str2num(get(A1_handle,'String'));
P1_variable = str2num(get(P1_handle,'String'));
V01_variable = str2num(get(V01_handle,'String'));
K1_variable = str2num(get(K1_handle,'String'));
A2_variable = str2num(get(A2_handle,'String'));
P2_variable = str2num(get(P2_handle,'String'));
V02_variable = str2num(get(V02_handle,'String'));
```

```
K2_variable = str2num(get(K2_handle,'String'));

V_variable = str2num(get(V_handle,'String'));
V_min_variable = str2num(get(V_min_handle,'String'));
V_step_variable = str2num(get(V_step_handle,'String'));
V_max_variable = str2num(get(V_max_handle,'String'));
number_of_channels=str2num(get(Channels_handle,'String'));
% end of variables
if P1_variable < -1 || (P1_variable >-1 && P1_variable < 1) || P1_variable >1
||...
   P2_variable < -1 || (P2_variable >-1 && P2_variable < 1) || P2_variable >1
 message_handle=msgbox('P1/P2 has to be +1 or -1!','Error');
 set(message_handle,'Units','normalized','Color',defaultBackground);
end
if (V_max_variable<V_min_variable) || (V_step_variable > V_max_variable) ||...
   (V_max_variable<V_min_variable) && (V_step_variable > V_max_variable)
 message_handle=msgbox('V max is smaller than V min or V step is greater than V
max.','Error');
 set(message_handle,'Units','normalized','Color',defaultBackground);
end
if round(number_of_channels) ~= number_of_channels || number_of_channels<=0
 message_handle=msgbox('Number of channels must be positive integer!','Error');
 set(message_handle,'Units','normalized','Color',defaultBackground);
end
   right=1;
   number=7;
   counter=1;
    while right<=number
         for left=1:1:number

V_left_hemichannel(1,counter)=(V_variable*g_right_hemichannel(P2_variable,0,right)
)/...

(g_left_hemichannel(P1_variable,0,left)+g_right_hemichannel(P2_variable,0,right));
         V_right_hemichannel(1,counter)=V_variable-V_left_hemichannel(1,counter);
         counter=counter+1;
         end
    right=right+1;
    end
    for t=2:1:5
       right=1;
      number=7;
      counter=1;
       while right<=number
         for left=1:1:number

V_left_hemichannel(t,counter)=(V_variable*g_right_hemichannel(P2_variable,V_right_
hemichannel(t-1,right),right))/...
      (g_left_hemichannel(P1_variable,V_left_hemichannel(t-
1,left),left)+g_right_hemichannel(P2_variable,V_right_hemichannel(t-
1,right),right));
       V_right_hemichannel(t,counter)=V_variable-V_left_hemichannel(t,counter);
       counter=counter+1;
         end
       right=right+1;
       end
    end
    V_left_steady=V_left_hemichannel(5,1:1:49);
    V_right_steady=V_right_hemichannel(5,1:1:49);
%      formation of a matrix
     function matrix=formationOfMatrix(A,P,V0,K,voltage_steady)
       states=zeros(7,2);
       for s=1:1:7
        states(s,1)=s-1; states(s,2)=6-(s-1);
```

```
        end
        function Cartesian=setCartesian(states_1,states_2)
        one=sortrows(repmat(states_1,[size(states_2,1) 1]));
        second=repmat(states_2,[size(states_1,1) 1]);
        Cartesian=[one second];
        end
        matrix=zeros(7);
        o=1; number_1=1;
        while o<8
             states1=zeros(states(o,1)+1,2);
             for l=1:1:states(o,1)+1
                states1(l,1)=states(o,1)-(l-1);
                states1(l,2)=l-1;
             end
             states2=zeros(states(o,2)+1,2);
             for l=1:1:states(o,2)+1
                states2(l,1)=l-1;
                states2(l,2)=states(o,2)-(l-1);
             end
             Cartesian_product=setCartesian(states1,states2);
             sum_1=sum(Cartesian_product(:,1:2:3),2);
             sum_2=sum(Cartesian_product(:,2:2:4),2);
             indices=[Cartesian_product sum_1 sum_2];
             number=size(indices,1);
             probabilities=zeros(number,1);
             for number1=1:1:number
                probabilities(number1)=...
                nchoosek(o-
1,indices(number1,1))*pcc(A,P,V0,K,voltage_steady(indices(number1,5)+number_1))^(i
ndices(number1,1))*...

pco(A,P,V0,K,voltage_steady(indices(number1,5)+number_1))^(indices(number1,2))*...
                nchoosek(6-(o-
1),indices(number1,3))*poc(A,P,V0,K,voltage_steady(indices(number1,5)+number_1))^(
indices(number1,3))*...

poo(A,P,V0,K,voltage_steady(indices(number1,5)+number_1))^(indices(number1,4));
             end
             probabilities2=zeros(7,1);
             for number1=1:1:number
                 for l=1:1:7
                    if indices(number1,5)==(l-1)
                      probabilities2(l)=probabilities2(l)+probabilities(number1);
                    end
                 end
             end
             for j=1:1:7
               matrix(o,j)=probabilities2(j);
             end
             number_1=number_1+7;
             o=o+1;
         end
      end

matrix_of_a_left_hemichannel=formationOfMatrix(A1_variable,P1_variable,V01_variabl
e,K1_variable,V_left_steady');

matrix_of_a_right_hemichannel=formationOfMatrix(A2_variable,P2_variable,V02_variab
le,K2_variable,V_right_steady');
  matrix_of_both_hemichannels=zeros(49);
  size_of_left_matrix=size(matrix_of_a_left_hemichannel,1);
  size_of_left_right_matrix=size(matrix_of_both_hemichannels,1);
  for i=1:size_of_left_matrix:size_of_left_right_matrix
    for j=1:size_of_left_matrix:size_of_left_right_matrix
```

```
     matrix_of_both_hemichannels(i:i+(size_of_left_matrix-
1),j:j+(size_of_left_matrix-1))=...
     matrix_of_a_left_hemichannel((i+(size_of_left_matrix-
1))/size_of_left_matrix,...
     (j+(size_of_left_matrix-
1))/size_of_left_matrix)*matrix_of_a_right_hemichannel;
    end
  end
  matrix_of_both_a=matrix_of_both_hemichannels;
     function normalised=normalised_matrix(matrix)
        normalised=zeros(size(matrix,1));
        sum_of_p_rows=sum(matrix,2);
        for i=1:1:length(sum_of_p_rows)
         normalised(i,:)=matrix(i,:)/sum_of_p_rows(i);
        end
     end
  matrix_of_both=normalised_matrix(matrix_of_both_a);
  X=stationary_probabilities(matrix_of_both);
  sum_of_p_rows=sum(matrix_of_both_hemichannels,2);
  initial_distribution=zeros(49,1);
  initial_distribution(1)=1;
 p(1,:)=initial_distribution';
 for i=1:1:200
    p(i+1,:)=p(1,:)*expm(matrix_of_both*i);
 end
 conductance=zeros(49,1);
   right=1;
   number=7;
   counter=1;
    while right<=number
         for left=1:1:number

conductance(counter)=number_of_channels*X(counter)*(g_left_hemichannel(P1_variable
,V_left_steady(1,counter),left)*g_right_hemichannel(P2_variable,V_right_steady(1,c
ounter),right))/...

(g_left_hemichannel(P1_variable,V_left_steady(1,counter),left)+g_right_hemichannel
(P2_variable,V_right_steady(1,counter),right));
         counter=counter+1;
         end
    right=right+1;
    end
 n=0:1:48; n=n';
 table=[n [V_left_steady' V_right_steady'] conductance];
% % % % % For drawing the graphic conductance(V), when V_variable change
% % values from the specified interval
function Cartesian=setCartesian(states_1,states_2)
       one=sortrows(repmat(states_1,[size(states_2,1) 1]));
       second=repmat(states_2,[size(states_1,1) 1]);
       Cartesian=[one second];
end
minimum=V_min_variable; max=V_max_variable; step=V_step_variable;
    count_voltages=numel(minimum:step:max);
    conductance_interval=zeros(count_voltages,1);
    index=1;
for V_variable_variates=minimum:step:max
 right=1;
   number=7;
   counter=1;
    while right<=number
         for left=1:1:number

V_left_hemichannel_2(1,counter)=(V_variable_variates*g_right_hemichannel(P2_variab
le,0,right))/...
```

```
(g_left_hemichannel(P1_variable,0,left)+g_right_hemichannel(P2_variable,0,right));
         V_right_hemichannel_2(1,counter)=V_variable_variates-
V_left_hemichannel(1,counter);
         counter=counter+1;
         end
    right=right+1;
    end
    for t=2:1:5
       right=1;
      number=7;
      counter=1;
       while right<=number
         for left=1:1:number

V_left_hemichannel_2(t,counter)=(V_variable_variates*g_right_hemichannel(P2_variab
le,V_right_hemichannel(t-1,right),right))/...
      (g_left_hemichannel(P1_variable,V_left_hemichannel(t-
1,left),left)+g_right_hemichannel(P2_variable,V_right_hemichannel(t-
1,right),right));
       V_right_hemichannel_2(t,counter)=V_variable_variates-
V_left_hemichannel(t,counter);
       counter=counter+1;
         end
       right=right+1;
       end
    end
    V_left_steady_2=V_left_hemichannel_2(5,1:1:49);
    V_right_steady_2=V_right_hemichannel_2(5,1:1:49);

matrix_of_a_left_hemichannel_2=formationOfMatrix(A1_variable,P1_variable,V01_varia
ble,K1_variable,V_left_steady_2');

matrix_of_a_right_hemichannel_2=formationOfMatrix(A2_variable,P2_variable,V02_vari
able,K2_variable,V_right_steady_2');
  matrix_of_both_hemichannels_2=zeros(49);
  size_of_left_matrix_2=size(matrix_of_a_left_hemichannel_2,1);
  size_of_left_right_matrix_2=size(matrix_of_both_hemichannels_2,1);
  for i=1:size_of_left_matrix_2:size_of_left_right_matrix_2
    for j=1:size_of_left_matrix_2:size_of_left_right_matrix_2
     matrix_of_both_hemichannels_2(i:i+(size_of_left_matrix_2-
1),j:j+(size_of_left_matrix_2-1))=...
     matrix_of_a_left_hemichannel_2((i+(size_of_left_matrix_2-
1))/size_of_left_matrix_2,...
      (j+(size_of_left_matrix_2-
1))/size_of_left_matrix_2)*matrix_of_a_right_hemichannel_2;
    end
  end
  X_2=stationary_probabilities(matrix_of_both_hemichannels_2);

  for s=1:1:7
    state(s)=s-1;
  end
   Cartesian_product_1=setCartesian(state',state');
   sum_1=sum(Cartesian_product_1(:,1:2),2);
   statmatrix_0=[Cartesian_product_1 sum_1 X_2];
   number=size(statmatrix_0,1);
   X_20=zeros(13,1);
       for number1=1:1:number
          for l=1:1:13
            if statmatrix_0(number1,3)==(l-1)
              X_20(l)=X_20(l)+statmatrix_0(number1,4);
            end
          end
```

```
        end
        for j=1:1:13
           statmatrix(j,index)=abs(X_20(j));
        end

  conductance_2=zeros(49,1);
   right=1;
   number=7;
   counter=1;
    while right<=number
         for left=1:1:number

conductance_2(counter)=number_of_channels*X_2(counter)*(g_left_hemichannel(P1_vari
able,V_left_steady_2(1,counter),left)*...
         g_right_hemichannel(P2_variable,V_right_steady_2(1,counter),right))/...
        (g_left_hemichannel(P1_variable,V_left_steady_2(1,counter),left)+...
        g_right_hemichannel(P2_variable,V_right_steady_2(1,counter),right));
         counter=counter+1;
         end
    right=right+1;
    end
 conductance_interval(index)=sum(conductance_2);
 index=index+1;
end

 figure_handle_0=figure();
  table_handle = uitable('Position', [100,340,450,300],'Data',table,...
                         'ColumnName', {'State','Voltage of the Left
Hemichannel','Voltage of the Right Hemichannel' , 'Conductance'},...
                         'ColumnFormat', {'numeric','numeric','numeric'},...
                         'ColumnWidth', {80 150 150 80});
  bar_handle =
bar(axes('Units','Pixels','Position',[600,350,600,300]),table(:,1),table(:,4));
  title 'Bar of a conductance of a channel (depending on a state)';
  xlabel('a state of left and right hemichannels, const.')
  ylabel('Conductance of a channel, pS')
  axis([0 48 0 2]);
  set(bar_handle);
  set(table_handle,'FontName','Courier','FontSize',12);
  voltage_interval=minimum:step:max; voltage_interval=voltage_interval';
  table_handle_2 = uitable('Position', [100,50,210,260],'Data',[voltage_interval
conductance_interval],...
                       'ColumnName', {'Voltage','Conductance'},...
                       'ColumnFormat', {'numeric','numeric'},...
                       'ColumnWidth', {80 80});
  plot_handle =
plot(axes('Units','Pixels','Position',[600,60,300,220]),voltage_interval,conductan
ce_interval,'LineWidth',2);
  set(table_handle_2,'FontName','Courier','FontSize',12);
  title 'Conductance dependence on voltage';
  xlabel('V, mV')
  ylabel('G(V), pS')
  axis([-100 100 0 7]);
  set(plot_handle);
  grid on;
  set(figure_handle_0,'Units','normalized','NumberTitle',
'off','Menubar','none','Name','Model of 12 Connexins','Color',defaultBackground);
  figure_handle_1=figure();
  i=1:1:201;
  subplot(2,4,1), plot(i',p(:,1),'b',i',p(:,2),'r',i',p(:,3),'g',i',p(:,4),'y',...
     i',p(:,5),'k',i',p(:,6),'c',i',p(:,7),'m','LineWidth',2);
  legend('p_0','p_1','p_2','p_3','p_4','p_5','p_6'); grid on;
```

```
  subplot(2,4,2),
plot(i',p(:,8),'b',i',p(:,9),'r',i',p(:,10),'g',i',p(:,11),'y',...
     i',p(:,12),'k',i',p(:,13),'c',i',p(:,14),'m','LineWidth',2);
  legend('p_7','p_8','p_9','p_1_0','p_1_1','p_1_2','p_1_3'); grid on;
  subplot(2,4,3),
plot(i',p(:,15),'b',i',p(:,16),'r',i',p(:,17),'g',i',p(:,18),'y',...
     i',p(:,19),'k',i',p(:,20),'c',i',p(:,21),'m','LineWidth',2);
  legend('p_1_4','p_1_5','p_1_6','p_1_7','p_1_8','p_1_9','p_2_0'); grid on;
  subplot(2,4,4),
plot(i',p(:,22),'b',i',p(:,23),'r',i',p(:,24),'g',i',p(:,25),'y',...
     i',p(:,26),'k',i',p(:,27),'c',i',p(:,28),'m','LineWidth',2);
  legend('p_2_1','p_2_2','p_2_3','p_2_4','p_2_5','p_2_6','p_2_7'); grid on;
  subplot(2,4,5),
plot(i',p(:,29),'b',i',p(:,30),'r',i',p(:,31),'g',i',p(:,32),'y',...
     i',p(:,33),'k',i',p(:,34),'c',i',p(:,35),'m','LineWidth',2);
  legend('p_2_8','p_2_9','p_3_0','p_3_1','p_3_2','p_3_3','p_3_4'); grid on;
  subplot(2,4,6),
plot(i',p(:,36),'b',i',p(:,37),'r',i',p(:,38),'g',i',p(:,39),'y',...
     i',p(:,40),'k',i',p(:,41),'c',i',p(:,42),'m','LineWidth',2);
  legend('p_3_5','p_3_6','p_3_7','p_3_8','p_3_9','p_4_0','p_4_1'); grid on;
  subplot(2,4,7),
plot(i',p(:,43),'b',i',p(:,44),'r',i',p(:,45),'g',i',p(:,46),'y',...
     i',p(:,47),'k',i',p(:,48),'c',i',p(:,49),'m','LineWidth',2);
  legend('p_4_2','p_4_3','p_4_4','p_4_5','p_4_6','p_4_7','p_4_8'); grid on;
  set(figure_handle_1,'Units','normalized','NumberTitle',
'off','Menubar','none','Name','Simulation of
Probabilities','Color',defaultBackground);
  figure_handle_2=figure();
  subplot(1,2,1),
plot(voltage_interval,statmatrix(1,:)','b',voltage_interval,statmatrix(2,:)','r',.
..

voltage_interval,statmatrix(3,:)','g',voltage_interval,statmatrix(4,:)','y',...

voltage_interval,statmatrix(5,:)','k',voltage_interval,statmatrix(6,:)','c',...
                      voltage_interval,statmatrix(7,:)','m','LineWidth',2);
  xlabel('Voltage (V), mV')
  ylabel('Probability')
  legend('0 closed connexins','1 closed connexin','2 closed connexins',...
         '3 closed connexins','4 closed connexins','5 closed connexins','6 closed
connexins'); grid on;
  subplot(1,2,2),
plot(voltage_interval,statmatrix(8,:)','b',voltage_interval,statmatrix(9,:)','r',.
..

voltage_interval,statmatrix(10,:)','g',voltage_interval,statmatrix(11,:)','y',...

voltage_interval,statmatrix(12,:)','k',voltage_interval,statmatrix(13,:)','c','Lin
eWidth',2);
  xlabel('Voltage (V), mV')
  ylabel('Probability')
  legend('7 closed connexins','8 closed connexin','9 closed connexins',...
         '10 closed connexins','11 closed connexins','12 closed connexins'); grid
on;
  set(figure_handle_2,'Units','normalized','NumberTitle',
'off','Menubar','none','Name','Dependency of
Probabilities','Color',defaultBackground);
end

function six_connexins_continuous_button_Callback(source,data)
% variables
A1_variable = str2num(get(A1_handle,'String'));
P1_variable = str2num(get(P1_handle,'String'));
V01_variable = str2num(get(V01_handle,'String'));
```

```
K1_variable = str2num(get(K1_handle,'String'));
A2_variable = str2num(get(A2_handle,'String'));
P2_variable = str2num(get(P2_handle,'String'));
V02_variable = str2num(get(V02_handle,'String'));
K2_variable = str2num(get(K2_handle,'String'));

V_variable = str2num(get(V_handle,'String'));
V_min_variable = str2num(get(V_min_handle,'String'));
V_step_variable = str2num(get(V_step_handle,'String'));
V_max_variable = str2num(get(V_max_handle,'String'));

number_of_channels=str2num(get(Channels_handle,'String'));
time_of_a_connexin=str2num(get(time1_handle,'String'));
% end of variables
if P1_variable < -1 || (P1_variable >-1 && P1_variable < 1) || P1_variable >1
||...
   P2_variable < -1 || (P2_variable >-1 && P2_variable < 1) || P2_variable >1
 message_handle=msgbox('P1/P2 has to be +1 or -1!','Error');
 set(message_handle,'Units','normalized','Color',defaultBackground);
end
if (V_max_variable<V_min_variable) || (V_step_variable > V_max_variable) ||...
   (V_max_variable<V_min_variable) && (V_step_variable > V_max_variable)
 message_handle=msgbox('V max is smaller than V min or V step is greater than V
max.','Error');
 set(message_handle,'Units','normalized','Color',defaultBackground);
end
if round(number_of_channels) ~= number_of_channels || number_of_channels<=0
 message_handle=msgbox('Number of channels must be positive integer!','Error');
 set(message_handle,'Units','normalized','Color',defaultBackground);
end
if time_of_a_connexin <= 0
 message_handle=msgbox('Time of a connexin being in the same state is non-
negative!','Error');
 set(message_handle,'Units','normalized','Color',defaultBackground);
end
% format shortE;
 function [matrix,
probability]=matrix_and_probability(A,P,V0,K,voltage_steady,moment)
    Rate=zeros(7);
    for i=1:1:7
     for j=1:1:7
       if i-j==-1
         Rate(i,j)=(7-i)*lambda_oc(A,P,V0,K,voltage_steady(j,1),moment);
       elseif i-j==1
         Rate(i,j)=j*lambda_co(A,P,V0,K,voltage_steady(j,1),moment);
       end
     end
    end
    sum_of_rate_rows=sum(Rate,2);
    Diagonal=zeros(7);
    for i=1:1:7
        for j=1:1:7
         if i==j
          Diagonal(i,j)=sum_of_rate_rows(j);
         end
        end
    end
    matrix=Rate-Diagonal;
    A=[matrix'; ones(1,size(matrix,1))];
    B=[zeros(size(matrix,1),1);1];
    probability=A\B;
 end
V=voltage_left_stationary(P1_variable,P2_variable,V_variable);
```

```
[Q,
PI]=matrix_and_probability(A1_variable,P1_variable,V01_variable,K1_variable,V,time
_of_a_connexin);

initial_distribution=[1 0 0 0 0 0 0];
 p(1,:)=initial_distribution(1,:);
 for step_i=1:1:9
    p(step_i+1,:)=p(1,:)*expm(Q*step_i);
 end
 conductance=zeros(7,1);
 for n=1:1:7

conductance(n)=number_of_channels*PI(n)*(g_left_hemichannel(P1_variable,V(n),n)*g_
right_hemichannel(P2_variable,V_variable-V(n),1))/...

(g_left_hemichannel(P1_variable,V(n),n)+g_right_hemichannel(P2_variable,V_variable
-V(n),1));
 end

 n=0:1:6; n=n';
 table = [n V conductance];
% % % % For drawing the graphic conductance(V), when V_variable change
% values from the specified interval
    minimum=V_min_variable; max=V_max_variable; step=V_step_variable;
    count_voltages=numel(minimum:step:max);
      V_left_hemichannel2=zeros(5,7);
      conductance_interval=zeros(count_voltages,1);
      index=1;
      for V_variable_variates=minimum:step:max
        for n=1:1:7

V_left_hemichannel2(1,n)=(V_variable_variates*g_right_hemichannel(P2_variable,0,1)
)/...

(g_left_hemichannel(P1_variable,0,n)+g_right_hemichannel(P2_variable,0,1));
        end
        for t=2:1:5
           for n=1:1:7

V_left_hemichannel2(t,n)=(V_variable_variates*g_right_hemichannel(P2_variable,V_va
riable_variates-V_left_hemichannel2(t-1,n),1))/...
            (g_left_hemichannel(P1_variable,V_left_hemichannel2(t-1,n),n)+...
            g_right_hemichannel(P2_variable,V_variable_variates-
V_left_hemichannel2(t-1,n),1));
          end
        end
       V2=V_left_hemichannel2(5,:)';

[matrix2,X2]=matrix_and_probability(A1_variable,P1_variable,V01_variable,K1_variab
le,V2,time_of_a_connexin);
      statmatrix(:,index)=abs(X2);
     conductance2=zeros(7,1);
     for n=1:1:7

conductance2(n)=number_of_channels*X2(n)*(g_left_hemichannel(P1_variable,V2(n),n)*
...
     g_right_hemichannel(P2_variable,V_variable_variates-V2(n),1))/...

(g_left_hemichannel(P1_variable,V2(n),n)+g_right_hemichannel(P2_variable,V_variabl
e_variates-V2(n),1));
     end
     conductance_interval(index)=sum(conductance2);
     index=index+1;
```

```
      end

  table_handle = uitable('Position', [220,440,272,170],'Data',table,...
                         'ColumnName', {'State','Voltage','Conductance'},...
                         'ColumnFormat', {'numeric','numeric','numeric'},...
                         'ColumnWidth', {80 80 80});
  bar_handle =
bar(axes('Units','Pixels','Position',[560,350,300,300]),table(:,1),table(:,3));
  title 'Bar of a conductance of a channel (depending on a state)';
  xlabel('a state of a left hemichannel, const.')
  ylabel('Conductance of a channel, pS')
  axis([0 7 0 6]);
  set(bar_handle);
  set(table_handle,'FontName','Courier','FontSize',12);
  voltage_interval=minimum:step:max; voltage_interval=voltage_interval';
  table_handle_2 = uitable('Position', [220,50,210,260],'Data',[voltage_interval
conductance_interval],...
                         'ColumnName', {'Voltage','Conductance'},...
                         'ColumnFormat', {'numeric','numeric'},...
                         'ColumnWidth', {80 80});
  plot_handle =
plot(axes('Units','Pixels','Position',[560,60,300,220]),voltage_interval,conductan
ce_interval,'LineWidth',2);
  set(table_handle_2,'FontName','Courier','FontSize',12);
  title 'Conductance dependence on voltage';
  xlabel('V, mV')
  ylabel('G(V), pS')
  axis([-100 100 0 7]);
  set(plot_handle);
  grid on;
   figure_handle_0=figure();
   i=1:1:10;
   plot(i',p(:,1),'b',i',p(:,2),'r',i',p(:,3),'g',i',p(:,4),'y',...
     i',p(:,5),'k',i',p(:,6),'c',i',p(:,7),'m','LineWidth',2);
   legend('p_0','p_1','p_2','p_3','p_4','p_5','p_6');
   grid on;
   set(figure_handle_0,'Units','normalized','NumberTitle',
'off','Menubar','none','Name','Simulation of
Probabilities','Color',defaultBackground);
   figure_handle_1=figure();

plot(voltage_interval,statmatrix(1,:)','b',voltage_interval,statmatrix(2,:)','r',.
..

voltage_interval,statmatrix(3,:)','g',voltage_interval,statmatrix(4,:)','y',...

voltage_interval,statmatrix(5,:)','k',voltage_interval,statmatrix(6,:)','c',...
        voltage_interval,statmatrix(7,:)','m','LineWidth',2);
   xlabel('Voltage (V), mV')
   ylabel('Probability')
   legend('0 closed connexins','1 closed connexin','2 closed connexins',...
          '3 closed connexins','4 closed connexins','5 closed connexins',...
          '6 closed connexins');
   grid on;
   set(figure_handle_1,'Units','normalized','NumberTitle',
'off','Menubar','none','Name','Dependency of
Probabilities','Color',defaultBackground);
end

function twelve_connexins_continuous_button_Callback(source,data)
% 12 connexins model
% variables
A1_variable = str2num(get(A1_handle,'String'));
```

```
P1_variable = str2num(get(P1_handle,'String'));
V01_variable = str2num(get(V01_handle,'String'));
K1_variable = str2num(get(K1_handle,'String'));
time1_of_a_connexin=str2num(get(time1_handle,'String'));
A2_variable = str2num(get(A2_handle,'String'));
P2_variable = str2num(get(P2_handle,'String'));
V02_variable = str2num(get(V02_handle,'String'));
K2_variable = str2num(get(K2_handle,'String'));
time2_of_a_connexin=str2num(get(time2_handle,'String'));

V_variable = str2num(get(V_handle,'String'));
V_min_variable = str2num(get(V_min_handle,'String'));
V_step_variable = str2num(get(V_step_handle,'String'));
V_max_variable = str2num(get(V_max_handle,'String'));

number_of_channels=str2num(get(Channels_handle,'String'));
% end of variables
if P1_variable < -1 || (P1_variable >-1 && P1_variable < 1) || P1_variable >1
||...
   P2_variable < -1 || (P2_variable >-1 && P2_variable < 1) || P2_variable >1
 message_handle=msgbox('P1/P2 has to be +1 or -1!','Error');
 set(message_handle,'Units','normalized','Color',defaultBackground);
end
if (V_max_variable<V_min_variable) || (V_step_variable > V_max_variable) ||...
   (V_max_variable<V_min_variable) && (V_step_variable > V_max_variable)
 message_handle=msgbox('V max is smaller than V min or V step is greater than V
max.','Error');
 set(message_handle,'Units','normalized','Color',defaultBackground);
end
if round(number_of_channels) ~= number_of_channels || number_of_channels<=0
 message_handle=msgbox('Number of channels must be positive integer!','Error');
 set(message_handle,'Units','normalized','Color',defaultBackground);
end
if time1_of_a_connexin <= 0 || time2_of_a_connexin <=0
 message_handle=msgbox('Time of a connexin being in the same state is non-
negative!','Error');
 set(message_handle,'Units','normalized','Color',defaultBackground);
end

   right=1;
   number=7;
   counter=1;
    while right<=number
         for left=1:1:number

V_left_hemichannel(1,counter)=(V_variable*g_right_hemichannel(P2_variable,0,right)
)/...

(g_left_hemichannel(P1_variable,0,left)+g_right_hemichannel(P2_variable,0,right));
         V_right_hemichannel(1,counter)=V_variable-V_left_hemichannel(1,counter);
         counter=counter+1;
         end
    right=right+1;
    end
    for t=2:1:5
       right=1;
      number=7;
      counter=1;
       while right<=number
         for left=1:1:number

V_left_hemichannel(t,counter)=(V_variable*g_right_hemichannel(P2_variable,V_right_
hemichannel(t-1,right),right))/...
```

```
      (g_left_hemichannel(P1_variable,V_left_hemichannel(t-
1,left),left)+g_right_hemichannel(P2_variable,V_right_hemichannel(t-
1,right),right));
       V_right_hemichannel(t,counter)=V_variable-V_left_hemichannel(t,counter);
       counter=counter+1;
         end
       right=right+1;
       end
    end
    V_left_steady=V_left_hemichannel(5,1:1:49);
    V_right_steady=V_right_hemichannel(5,1:1:49);
%      formation of a matrix
  function Rate=formationOfMatrix(A,P,V0,K,voltage_steady,moment)
   Rate=zeros(7);
   ind_voltage_oc=2;
   ind_voltage_co=8;
    for i=1:1:7
     for j=1:1:7
       if i-j==-1
         Rate(i,j)=(7-
i)*lambda_oc(A,P,V0,K,voltage_steady(ind_voltage_oc,1),moment);
         ind_voltage_oc=ind_voltage_oc+8;
       elseif i-j==1
         Rate(i,j)=j*lambda_co(A,P,V0,K,voltage_steady(ind_voltage_co,1),moment);
         ind_voltage_co=ind_voltage_co+8;
       end
     end
    end
  end

matrix_of_a_left_hemichannel=formationOfMatrix(A1_variable,P1_variable,V01_variabl
e,K1_variable,V_left_steady',time1_of_a_connexin);

matrix_of_a_right_hemichannel=formationOfMatrix(A2_variable,P2_variable,V02_variab
le,K2_variable,V_right_steady',time2_of_a_connexin);
  matrix_of_both_hemichannels=zeros(49);
  size_of_left_matrix=size(matrix_of_a_left_hemichannel,1);
  size_of_left_right_matrix=size(matrix_of_both_hemichannels,1);
  for i=1:size_of_left_matrix:size_of_left_right_matrix
    for j=1:size_of_left_matrix:size_of_left_right_matrix
     matrix_of_both_hemichannels(i:i+(size_of_left_matrix-
1),j:j+(size_of_left_matrix-1))=...
     matrix_of_a_left_hemichannel((i+(size_of_left_matrix-
1))/size_of_left_matrix,...
      (j+(size_of_left_matrix-
1))/size_of_left_matrix)+matrix_of_a_right_hemichannel;
    end
  end
  function prob=stationary(matrix)
    A=[matrix'; ones(1,size(matrix,1))];
    B=[zeros(size(matrix,1),1);1];
    prob=A\B;
  end

 function matrix=fully_formed_matrix(rate)
  sum_of_rate_rows=sum(rate,2);
    Diagonal=zeros(size(rate,1));
    for i=1:1:size(rate,1)
        for j=1:1:size(rate,1)
         if i==j
          Diagonal(i,j)=sum_of_rate_rows(j);
         end
        end
    end
```

```
    matrix=rate-Diagonal;
 end
  matrix_of_both=fully_formed_matrix(matrix_of_both_hemichannels);
  X=stationary(matrix_of_both);
 conductance=zeros(49,1);
   right=1;
   number=7;
   counter=1;
    while right<=number
         for left=1:1:number
         conductance(counter)=number_of_channels*X(counter)*...
         (g_left_hemichannel(P1_variable,V_left_steady(1,counter),left)*...
         g_right_hemichannel(P2_variable,V_right_steady(1,counter),right))/...
        (g_left_hemichannel(P1_variable,V_left_steady(1,counter),left)+...
         g_right_hemichannel(P2_variable,V_right_steady(1,counter),right));
         counter=counter+1;
         end
    right=right+1;
    end
 n=0:1:48; n=n';
 table=[n [V_left_steady' V_right_steady'] conductance];
% % % % % For drawing the graphic conductance(V), when V_variable change
% % values from the specified interval
function Cartesian=setCartesian(states_1,states_2)
       one=sortrows(repmat(states_1,[size(states_2,1) 1]));
       second=repmat(states_2,[size(states_1,1) 1]);
       Cartesian=[one second];
end
minimum=V_min_variable; max=V_max_variable; step=V_step_variable;
    count_voltages=numel(minimum:step:max);
    conductance_interval=zeros(count_voltages,1);
    index=1;
for V_variable_variates=minimum:step:max
   right=1;
   number=7;
   counter=1;
    while right<=number
         for left=1:1:number
         V_left_hemichannel_2(1,counter)=(V_variable_variates*...
         g_right_hemichannel(P2_variable,0,right))/...

(g_left_hemichannel(P1_variable,0,left)+g_right_hemichannel(P2_variable,0,right));
         V_right_hemichannel_2(1,counter)=V_variable_variates-
V_left_hemichannel(1,counter);
         counter=counter+1;
         end
    right=right+1;
    end
    for t=2:1:5
      right=1;
      number=7;
      counter=1;
       while right<=number
         for left=1:1:number
       V_left_hemichannel_2(t,counter)=(V_variable_variates*...
       g_right_hemichannel(P2_variable,V_right_hemichannel(t-1,right),right))/...
      (g_left_hemichannel(P1_variable,V_left_hemichannel(t-1,left),left)+...
       g_right_hemichannel(P2_variable,V_right_hemichannel(t-1,right),right));
       V_right_hemichannel_2(t,counter)=V_variable_variates-
V_left_hemichannel(t,counter);
       counter=counter+1;
         end
       right=right+1;
       end
```

```
    end
    V_left_steady_2=V_left_hemichannel_2(5,1:1:49);
    V_right_steady_2=V_right_hemichannel_2(5,1:1:49);

matrix_of_a_left_hemichannel_2=formationOfMatrix(A1_variable,P1_variable,V01_varia
ble,K1_variable,V_left_steady_2',time1_of_a_connexin);

matrix_of_a_right_hemichannel_2=formationOfMatrix(A2_variable,P2_variable,V02_vari
able,K2_variable,V_right_steady_2',time2_of_a_connexin);
  matrix_of_both_hemichannels_2=zeros(49);
  size_of_left_matrix_2=size(matrix_of_a_left_hemichannel_2,1);
  size_of_left_right_matrix_2=size(matrix_of_both_hemichannels_2,1);
  for i=1:size_of_left_matrix_2:size_of_left_right_matrix_2
    for j=1:size_of_left_matrix_2:size_of_left_right_matrix_2
     matrix_of_both_hemichannels_2(i:i+(size_of_left_matrix_2-
1),j:j+(size_of_left_matrix_2-1))=...
     matrix_of_a_left_hemichannel_2((i+(size_of_left_matrix_2-
1))/size_of_left_matrix_2,...
      (j+(size_of_left_matrix_2-
1))/size_of_left_matrix_2)+matrix_of_a_right_hemichannel_2;
    end
  end
  matrix_of_both_2=fully_formed_matrix(matrix_of_both_hemichannels_2);
  X_2=stationary(matrix_of_both_2);

  for s=1:1:7
    state(s)=s-1;
  end
   Cartesian_product_1=setCartesian(state',state');
   sum_1=sum(Cartesian_product_1(:,1:2),2);
   statmatrix_0=[Cartesian_product_1 sum_1 X_2];
   number=size(statmatrix_0,1);
   X_20=zeros(13,1);
       for number1=1:1:number
          for l=1:1:13
            if statmatrix_0(number1,3)==(l-1)
              X_20(l)=X_20(l)+statmatrix_0(number1,4);
            end
          end
       end
       for j=1:1:13
          statmatrix(j,index)=abs(X_20(j));
       end

  conductance_2=zeros(49,1);
   right=1;
   number=7;
   counter=1;
    while right<=number
         for left=1:1:number
         conductance_2(counter)=number_of_channels*X_2(counter)*...
        (g_left_hemichannel(P1_variable,V_left_steady_2(1,counter),left)*...
         g_right_hemichannel(P2_variable,V_right_steady_2(1,counter),right))/...
        (g_left_hemichannel(P1_variable,V_left_steady_2(1,counter),left)+...
         g_right_hemichannel(P2_variable,V_right_steady_2(1,counter),right));
         counter=counter+1;
         end
    right=right+1;
    end
 conductance_interval(index)=sum(conductance_2);
 index=index+1;
end
```

```
 figure_handle_0=figure();
  table_handle = uitable('Position', [100,340,450,300],'Data',table,...
                         'ColumnName', {'State','Voltage of the Left
Hemichannel','Voltage of the Right Hemichannel' , 'Conductance'},...
                         'ColumnFormat', {'numeric','numeric','numeric'},...
                         'ColumnWidth', {80 150 150 80});
  bar_handle =
bar(axes('Units','Pixels','Position',[600,350,600,300]),table(:,1),table(:,4));
  title 'Bar of a conductance of a channel (depending on a state)';
  xlabel('a state of left and right hemichannels, const.')
  ylabel('Conductance of a channel, pS')
  axis([0 48 0 2]);
  set(bar_handle);
  set(table_handle,'FontName','Courier','FontSize',12);
  voltage_interval=minimum:step:max; voltage_interval=voltage_interval';
  table_handle_2 = uitable('Position', [100,50,210,260],'Data',[voltage_interval
conductance_interval],...
                        'ColumnName', {'Voltage','Conductance'},...
                        'ColumnFormat', {'numeric','numeric'},...
                        'ColumnWidth', {80 80});
  plot_handle =
plot(axes('Units','Pixels','Position',[600,60,300,220]),voltage_interval,conductan
ce_interval,'LineWidth',2);
  set(table_handle_2,'FontName','Courier','FontSize',12);
  title 'Conductance dependence on voltage';
  xlabel('V, mV')
  ylabel('G(V), pS')
  axis([-100 100 0 7]);
  set(plot_handle);
  grid on;
  set(figure_handle_0,'Units','normalized','NumberTitle',
'off','Menubar','none','Name','Continuous Time Model of 12
Connexins','Color',defaultBackground);
  figure_handle_1=figure();
  subplot(1,2,1),
plot(voltage_interval,statmatrix(1,:)','b',voltage_interval,statmatrix(2,:)','r',.
..

voltage_interval,statmatrix(3,:)','g',voltage_interval,statmatrix(4,:)','y',...

voltage_interval,statmatrix(5,:)','k',voltage_interval,statmatrix(6,:)','c',...
                       voltage_interval,statmatrix(7,:)','m','LineWidth',2);
  xlabel('Voltage (V), mV')
  ylabel('Probability')
  legend('0 closed connexins','1 closed connexin','2 closed connexins',...
         '3 closed connexins','4 closed connexins','5 closed connexins','6 closed
connexins'); grid on;
  subplot(1,2,2),
plot(voltage_interval,statmatrix(8,:)','b',voltage_interval,statmatrix(9,:)','r',.
..

voltage_interval,statmatrix(10,:)','g',voltage_interval,statmatrix(11,:)','y',...

voltage_interval,statmatrix(12,:)','k',voltage_interval,statmatrix(13,:)','c','Lin
eWidth',2);
  xlabel('Voltage (V), mV')
  ylabel('Probability')
  legend('7 closed connexins','8 closed connexin','9 closed connexins',...
         '10 closed connexins','11 closed connexins','12 closed connexins'); grid
on;
  set(figure_handle_1,'Units','normalized','NumberTitle',
'off','Menubar','none','Name','Dependency of
Probabilities','Color',defaultBackground);
end
```

```
function six_connexins_button2_Callback(source,data)
% variables
A1_variable = str2num(get(A1_handle,'String'));
P1_variable = str2num(get(P1_handle,'String'));
V01_variable = str2num(get(V01_handle,'String'));
K1_variable = str2num(get(K1_handle,'String'));
pcdc1_variable = str2num(get(pcdc1_handle,'String'));
pdcc1_variable = str2num(get(pdcc1_handle,'String'));
pdcdc1_variable=1-pdcc1_variable;
A2_variable = str2num(get(A2_handle,'String'));
P2_variable = str2num(get(P2_handle,'String'));
V02_variable = str2num(get(V02_handle,'String'));
K2_variable = str2num(get(K2_handle,'String'));
pcdc2_variable = str2num(get(pcdc2_handle,'String'));
pdcc2_variable = str2num(get(pdcc2_handle,'String'));
pdcdc2_variable=1-pdcc2_variable;

V_variable = str2num(get(V_handle,'String'));
V_min_variable = str2num(get(V_min_handle,'String'));
V_step_variable = str2num(get(V_step_handle,'String'));
V_max_variable = str2num(get(V_max_handle,'String'));
number_of_channels=str2num(get(Channels_handle,'String'));
% end of variables
if P1_variable < -1 || (P1_variable >-1 && P1_variable < 1) || P1_variable >1
||...
   P2_variable < -1 || (P2_variable >-1 && P2_variable < 1) || P2_variable >1
 message_handle=msgbox('P1/P2 has to be +1 or -1!','Error');
 set(message_handle,'Units','normalized','Color',defaultBackground);
end
if (V_max_variable<V_min_variable) || (V_step_variable > V_max_variable) ||...
   (V_max_variable<V_min_variable) && (V_step_variable > V_max_variable)
 message_handle=msgbox('V max is smaller than V min or V step is greater than V
max.','Error');
 set(message_handle,'Units','normalized','Color',defaultBackground);
end
if round(number_of_channels) ~= number_of_channels || number_of_channels<=0
 message_handle=msgbox('Number of channels must be positive integer!','Error');
 set(message_handle,'Units','normalized','Color',defaultBackground);
end
if pcdc1_variable < 0 || pcdc1_variable > 1 ||...
   pcdc2_variable < 0 || pcdc2_variable > 1 ||...
  ((pcdc1_variable < 0 || pcdc1_variable > 1) &&...
  (pcdc2_variable < 0 || pcdc2_variable > 1)) ||...
   pdcc1_variable < 0 || pdcc1_variable > 1 ||...
   pdcc2_variable < 0 || pdcc2_variable > 1 ||...
   ((pdcc1_variable < 0 || pdcc1_variable > 1) &&...
   (pdcc2_variable < 0 || pdcc2_variable > 1))
 message_handle=msgbox('pcdc and pdcc must be in the range of [0;1]!','Error');
 set(message_handle,'Units','normalized','Color',defaultBackground);
end

  states=zeros(28,3);
      n=1; nnn=0;
      while n<8
         for nn=1:1:7-(n-1)
          states(nnn+1,1)=n-1; states(nnn+1,2)=nn-1; states(nnn+1,3)=6-(n-1)-(nn-
1);
          nnn=nnn+1;
         end
      n=n+1;
      end
    for n=1:1:28
```

```
V_left_hemichannel(1,n)=(V_variable*g_right_hemichannel_3(P2_variable,0,0,0,6))/..
.
     (g_left_hemichannel_3(P1_variable,0,states(n,1),states(n,2),states(n,3))+...
      g_right_hemichannel_3(P2_variable,0,0,0,6));
    end
    for t=2:1:5
       for n=1:1:28
       V_left_hemichannel(t,n)=(V_variable*...
       g_right_hemichannel_3(P2_variable,V_variable-V_left_hemichannel(t-
1,n),0,0,6))/...
      (g_left_hemichannel_3(P1_variable,V_left_hemichannel(t-
1,n),states(n,1),states(n,2),states(n,3))+...
       g_right_hemichannel_3(P2_variable,V_variable-V_left_hemichannel(t-
1,n),0,0,6));
       end
    end
    V=V_left_hemichannel(5,:)';

%      formation of a matrix, when connexin can be in 3 states: open,
%      closed or deep-closed
     function matrix=formationOfMatrix_3(A,P,V0,K,voltage_steady)
      states=zeros(28,3);
      n=1; nnn=0;
      while n<8
         for nn=1:1:7-(n-1)
          states(nnn+1,1)=n-1; states(nnn+1,2)=nn-1; states(nnn+1,3)=6-(n-1)-(nn-
1);
          nnn=nnn+1;
         end
       n=n+1;
       end
        function Cartesian=setCartesian(states_1,states_2)
        one=sortrows(repmat(states_1,[size(states_2,1) 1]));
        second=repmat(states_2,[size(states_1,1) 1]);
        Cartesian=[one second];
        end
        matrix=zeros(28);
        o=1;
        while o<=28
             states1=zeros(states(o,1)+1,3);
             for l=1:1:states(o,1)+1
                states1(l,1)=states(o,1)-(l-1);
                states1(l,2)=l-1;
                states1(l,3)=0;
             end
             n=1; nnn=0;
             while n<=states(o,2)+1
               for nn=1:1:states(o,2)+1-(n-1)
                states2(nnn+1,1)=n-1;
                states2(nnn+1,2)=nn-1;
                states2(nnn+1,3)=states(o,2)-(n-1)-(nn-1);
                nnn=nnn+1;
               end
             n=n+1;
             end
             states3=zeros(states(o,3)+1,3);
             for l=1:1:states(o,3)+1
                states3(l,1)=0;
                states3(l,2)=l-1;
                states3(l,3)=states(o,3)-(l-1);
             end
             Cartesian_product_half=setCartesian(states1,states2);
             Cartesian_product_full=setCartesian(Cartesian_product_half,states3);
```

```
            sum_1=sum(Cartesian_product_full(:,1:3:7),2);
            sum_2=sum(Cartesian_product_full(:,2:3:8),2);
            sum_3=sum(Cartesian_product_full(:,3:3:9),2);
            indices=[Cartesian_product_full sum_1 sum_2 sum_3];
            number=size(indices,1);
            probabilities=zeros(number,1);
            for number1=1:1:number
               probabilities(number1)=...

nchoosek(states(o,1),indices(number1,1))*pdcc1_variable^(indices(number1,2))*...
                pdcdc1_variable^(indices(number1,1))*...

(factorial(states(o,2))/(factorial(indices(number1,4))*factorial(indices(number1,5
))*factorial(indices(number1,6))))*...

pco_new(A,P,V0,K,pcdc1_variable,voltage_steady(o))^(indices(number1,4))*...

pcc_new(A,P,V0,K,pcdc1_variable,voltage_steady(o))^(indices(number1,5))*...
                pcdc1_variable^(indices(number1,6))*...
                nchoosek(states(o,3),indices(number1,9))*...
                poc(A,P,V0,K,voltage_steady(o))^(indices(number1,8))*...
                poo(A,P,V0,K,voltage_steady(o))^(indices(number1,9));

            end
            probabilities2=zeros(28,1);
            for number1=1:1:number
                for l=1:1:28
                   if indices(number1,10)==states(l,1) &&
indices(number1,11)==states(l,2)
                     probabilities2(l)=probabilities2(l)+probabilities(number1);
                   end
                end
            end
            for j=1:1:28
              matrix(o,j)=probabilities2(j);
            end
            o=o+1;
        end
     end

matrix_a=formationOfMatrix_3(A1_variable,P1_variable,V01_variable,K1_variable,V);

    function normalised=normalised_matrix(matrix)
        normalised=zeros(size(matrix,1));
        sum_of_p_rows=sum(matrix,2);
        for i=1:1:length(sum_of_p_rows)
         normalised(i,:)=matrix(i,:)/sum_of_p_rows(i);
        end
    end
  matrix=normalised_matrix(matrix_a);
  A=[matrix' - eye(size(matrix,1)); ones(1,size(matrix,1))];
  B=[zeros(size(matrix,1),1);1];
  X=A\B;
 initial_distribution=zeros(28,1);
 initial_distribution(1)=1;
 p(1,:)=initial_distribution';
 for i=1:1:100
    p(i+1,:)=p(1,:)*(matrix^i);
 end

 conductance=zeros(28,1);
 for n=1:1:28
 conductance(n)=number_of_channels*X(n)*...
```

```
 (g_left_hemichannel_3(P1_variable,V(n),states(n,1),states(n,2),states(n,3))*...
 g_right_hemichannel_3(P2_variable,V_variable-V(n),0,0,6))/...
 (g_left_hemichannel_3(P1_variable,V(n),states(n,1),states(n,2),states(n,3))+...
 g_right_hemichannel_3(P2_variable,V_variable-V(n),0,0,6));
 end
 n=0:1:27; n=n';
 table = [n V conductance];
% % % % For drawing the graphic conductance(V), when V_variable change
% values from the specified interval
    min=V_min_variable; max=V_max_variable; step=V_step_variable;
    count_voltages=numel(min:step:max);
      V_left_hemichannel2=zeros(5,28);
      conductance_interval=zeros(count_voltages,1);
      index=1;
      for V_variable_variates=min:step:max
        for n=1:1:28
        V_left_hemichannel2(1,n)=(V_variable_variates*...
        g_right_hemichannel_3(P2_variable,0,0,0,6))/...

(g_left_hemichannel_3(P1_variable,0,states(n,1),states(n,2),states(n,3))+...
        g_right_hemichannel_3(P2_variable,0,0,0,6));
        end

        for t=2:1:5
           for n=1:1:28
            V_left_hemichannel2(t,n)=(V_variable_variates*...
          g_right_hemichannel_3(P2_variable,V_variable_variates-
V_left_hemichannel2(t-1,n),0,0,6))/...
          (g_left_hemichannel_3(P1_variable,V_left_hemichannel2(t-1,n),...
           states(n,1),states(n,2),states(n,3))+...
           g_right_hemichannel_3(P2_variable,V_variable_variates-
V_left_hemichannel2(t-1,n),0,0,6));
           end
        end
       V2=V_left_hemichannel2(5,:)';

matrix_a2=formationOfMatrix_3(A1_variable,P1_variable,V01_variable,K1_variable,V2)
;
       matrix2=normalised_matrix(matrix_a2);
      X2=[matrix2' - eye(size(matrix2)); ones(1,length(matrix2))] \...
          [zeros(length(matrix2),1);1];

    states=zeros(28,3);
      n=1; nnn=0;
      while n<8
         for nn=1:1:7-(n-1)
          states(nnn+1,1)=n-1; states(nnn+1,2)=nn-1; states(nnn+1,3)=6-(n-1)-(nn-
1);
          nnn=nnn+1;
         end
      n=n+1;
      end
   sum_1=sum(states(:,1:2),2);
   statmatrix_0=[states sum_1 X2];
   number=size(statmatrix_0,1);
   X_20=zeros(7,1);
       for number1=1:1:number
          for l=1:1:7
            if statmatrix_0(number1,4)==(l-1)
              X_20(l)=X_20(l)+statmatrix_0(number1,5);
            end
          end
       end
```

```
       for j=1:1:7
          statmatrix(j,index)=abs(X_20(j));
       end

     conductance2=zeros(28,1);
     for n=1:1:28
     conductance2(n)=number_of_channels*X2(n)*...

(g_left_hemichannel_3(P1_variable,V2(n),states(n,1),states(n,2),states(n,3))*...
      g_right_hemichannel_3(P2_variable,V_variable_variates-V2(n),0,0,6))/...

(g_left_hemichannel_3(P1_variable,V2(n),states(n,1),states(n,2),states(n,3))+...
      g_right_hemichannel_3(P2_variable,V_variable_variates-V2(n),0,0,6));
     end
     conductance_interval(index)=sum(conductance2);
     index=index+1;
      end

  figure_handle_0=figure();
  table_handle = uitable('Position', [100,340,450,300],'Data',table,...
                         'ColumnName', {'State','Voltage of the Left
Hemichannel','Conductance'},...
                         'ColumnFormat', {'numeric','numeric','numeric'},...
                         'ColumnWidth', {80 150 80});
  bar_handle =
bar(axes('Units','Pixels','Position',[600,350,600,300]),table(:,1),table(:,3));
  title 'Bar of a conductance of a channel (depending on a state)';
  xlabel('a state of left and right hemichannels, const.')
  ylabel('Conductance of a channel, pS')
  axis([0 27 0 2]);
  set(bar_handle);
  set(table_handle,'FontName','Courier','FontSize',12);
  voltage_interval=min:step:max; voltage_interval=voltage_interval';
  table_handle_2 = uitable('Position', [100,50,210,260],'Data',[voltage_interval
conductance_interval],...
                       'ColumnName', {'Voltage','Conductance'},...
                       'ColumnFormat', {'numeric','numeric'},...
                       'ColumnWidth', {80 80});
  plot_handle =
plot(axes('Units','Pixels','Position',[600,60,300,220]),voltage_interval,conductan
ce_interval,'LineWidth',2);
  set(table_handle_2,'FontName','Courier','FontSize',12);
  title 'Conductance dependence on voltage';
  xlabel('V, mV')
  ylabel('G(V), pS')
  axis([-100 100 0 7]);
  set(plot_handle);
  grid on;
  set(figure_handle_0,'Units','normalized','NumberTitle',
'off','Menubar','none','Name','Model of 6 Connexins (3
States)','Color',defaultBackground);
  figure_handle_1=figure();
  i=1:1:101;
  subplot(2,2,1),
  plot(i',p(:,1),'b',i',p(:,2),'r',i',p(:,3),'g',i',p(:,4),'y',...
     i',p(:,5),'k',i',p(:,6),'c',i',p(:,7),'m','LineWidth',2);
  legend('p_0','p_1','p_2','p_3','p_4','p_5','p_6');
  grid on;
  subplot(2,2,2),
  plot(i',p(:,8),'b',i',p(:,9),'r',i',p(:,10),'g',i',p(:,11),'y',...
     i',p(:,12),'k',i',p(:,13),'c',i',p(:,14),'m','LineWidth',2);
  legend('p_7','p_8','p_9','p_1_0','p_1_1','p_1_2','p_1_3');
  grid on;
```

```
  subplot(2,2,3),
  plot(i',p(:,15),'b',i',p(:,16),'r',i',p(:,17),'g',i',p(:,18),'y',...
     i',p(:,19),'k',i',p(:,20),'c',i',p(:,21),'m','LineWidth',2);
  legend('p_1_4','p_1_5','p_1_6','p_1_7','p_1_8','p_1_9','p_2_0');
  grid on;
  subplot(2,2,4),
  plot(i',p(:,22),'b',i',p(:,23),'r',i',p(:,24),'g',i',p(:,25),'y',...
     i',p(:,26),'k',i',p(:,27),'c',i',p(:,28),'m','LineWidth',2);
  legend('p_2_1','p_2_2','p_2_3','p_2_4','p_2_5','p_2_6','p_2_7');
  grid on;
  set(figure_handle_1,'Units','normalized','NumberTitle',
'off','Menubar','none','Name','Simulation of
Probabilities','Color',defaultBackground);
  figure_handle_2=figure();

plot(voltage_interval,statmatrix(1,:)','b',voltage_interval,statmatrix(2,:)','r',.
..

voltage_interval,statmatrix(3,:)','g',voltage_interval,statmatrix(4,:)','y',...

voltage_interval,statmatrix(5,:)','k',voltage_interval,statmatrix(6,:)','c',...
        voltage_interval,statmatrix(7,:)','m','LineWidth',2);
   xlabel('Voltage (V), mV')
   ylabel('Probability')
   legend('0 closed connexins','1 closed connexin','2 closed connexins',...
          '3 closed connexins','4 closed connexins','5 closed connexins',...
          '6 closed connexins');
   grid on;
   set(figure_handle_2,'Units','normalized','NumberTitle',
'off','Menubar','none','Name','Dependency of
Probabilities','Color',defaultBackground);
end

function twelve_connexins_button2_Callback(source,data)
% 12 connexins model
% variables
A1_variable = str2num(get(A1_handle,'String'));
P1_variable = str2num(get(P1_handle,'String'));
V01_variable = str2num(get(V01_handle,'String'));
K1_variable = str2num(get(K1_handle,'String'));
pcdc1_variable = str2num(get(pcdc1_handle,'String'));
pdcc1_variable = str2num(get(pdcc1_handle,'String'));
pdcdc1_variable=1-pdcc1_variable;
A2_variable = str2num(get(A2_handle,'String'));
P2_variable = str2num(get(P2_handle,'String'));
V02_variable = str2num(get(V02_handle,'String'));
K2_variable = str2num(get(K2_handle,'String'));
pcdc2_variable = str2num(get(pcdc2_handle,'String'));
pdcc2_variable = str2num(get(pdcc2_handle,'String'));
pdcdc2_variable=1-pdcc2_variable;

V_variable = str2num(get(V_handle,'String'));
V_min_variable = str2num(get(V_min_handle,'String'));
V_step_variable = str2num(get(V_step_handle,'String'));
V_max_variable = str2num(get(V_max_handle,'String'));
number_of_channels=str2num(get(Channels_handle,'String'));
% end of variables
if P1_variable < -1 || (P1_variable >-1 && P1_variable < 1) || P1_variable >1
||...
   P2_variable < -1 || (P2_variable >-1 && P2_variable < 1) || P2_variable >1
 message_handle=msgbox('P1/P2 has to be +1 or -1!','Error');
 set(message_handle,'Units','normalized','Color',defaultBackground);
end
```

```
if (V_max_variable<V_min_variable) || (V_step_variable > V_max_variable) ||...
   (V_max_variable<V_min_variable) && (V_step_variable > V_max_variable)
 message_handle=msgbox('V max is smaller than V min or V step is greater than V
max.','Error');
 set(message_handle,'Units','normalized','Color',defaultBackground);
end
if round(number_of_channels) ~= number_of_channels || number_of_channels<=0
 message_handle=msgbox('Number of channels must be positive integer!','Error');
 set(message_handle,'Units','normalized','Color',defaultBackground);
end
if pcdc1_variable < 0 || pcdc1_variable > 1 ||...
   pcdc2_variable < 0 || pcdc2_variable > 1 ||...
  ((pcdc1_variable < 0 || pcdc1_variable > 1) &&...
  (pcdc2_variable < 0 || pcdc2_variable > 1)) ||...
   pdcc1_variable < 0 || pdcc1_variable > 1 ||...
   pdcc2_variable < 0 || pdcc2_variable > 1 ||...
   ((pdcc1_variable < 0 || pdcc1_variable > 1) &&...
   (pdcc2_variable < 0 || pdcc2_variable > 1))
 message_handle=msgbox('pcdc and pdcc must be in the range of [0;1]!','Error');
 set(message_handle,'Units','normalized','Color',defaultBackground);
end

 states=zeros(28,3);
 n=1; nnn=0;
 while n<8
   for nn=1:1:7-(n-1)
   states(nnn+1,1)=n-1; states(nnn+1,2)=nn-1; states(nnn+1,3)=6-(n-1)-(nn-1);
   nnn=nnn+1;
   end
   n=n+1;
 end
   right=1;
   number=28;
   counter=1;
  while right<=number
     for left=1:1:number
       V_left_hemichannel(1,counter)=(V_variable*...

g_right_hemichannel_3(P2_variable,0,states(right,1),states(right,2),states(right,3
))))/...

(g_left_hemichannel_3(P1_variable,0,states(left,1),states(left,2),states(left,3))+
...

g_right_hemichannel_3(P2_variable,0,states(right,1),states(right,2),states(right,3
)));
       V_right_hemichannel(1,counter)=V_variable-V_left_hemichannel(1,counter);
       counter=counter+1;
     end
    right=right+1;
  end
 for t=2:1:5
       right=1;
      number=28;
      counter=1;
   while right<=number
     for left=1:1:number
       V_left_hemichannel(t,counter)=(V_variable*...
       g_right_hemichannel_3(P2_variable,V_right_hemichannel(t-
1,right),states(right,1),states(right,2),states(right,3)))/...
      (g_left_hemichannel_3(P1_variable,V_left_hemichannel(t-
1,left),states(left,1),states(left,2),states(left,3))+...
       g_right_hemichannel_3(P2_variable,V_right_hemichannel(t-
1,right),states(right,1),states(right,2),states(right,3)));
```

```
       V_right_hemichannel(t,counter)=V_variable-V_left_hemichannel(t,counter);
       counter=counter+1;
     end
    right=right+1;
   end
 end
    V_left_steady=V_left_hemichannel(5,1:1:28*28);
    V_right_steady=V_right_hemichannel(5,1:1:28*28);
%      formation of a matrix
     function
matrix=formationOfMatrix12_3(A,P,V0,K,pcdc,pdcc,pdcdc,voltage_steady)
      states=zeros(28,3);
      n=1; nnn=0;
      while n<8
      for nn=1:1:7-(n-1)
       states(nnn+1,1)=n-1; states(nnn+1,2)=nn-1; states(nnn+1,3)=6-(n-1)-(nn-1);
       nnn=nnn+1;
      end
      n=n+1;
      end
       function Cartesian=setCartesian(states_1,states_2)
       one=sortrows(repmat(states_1,[size(states_2,1) 1]));
       second=repmat(states_2,[size(states_1,1) 1]);
       Cartesian=[one second];
       end
       matrix=zeros(28);
       o=1; number_1=1;
       while o<=28
            states1=zeros(states(o,1)+1,3);
            for l=1:1:states(o,1)+1
               states1(l,1)=states(o,1)-(l-1);
               states1(l,2)=l-1;
               states1(l,3)=0;
            end
            n=1; nnn=0;
            while n<=states(o,2)+1
              for nn=1:1:states(o,2)+1-(n-1)
               states2(nnn+1,1)=n-1;
               states2(nnn+1,2)=nn-1;
               states2(nnn+1,3)=states(o,2)-(n-1)-(nn-1);
               nnn=nnn+1;
              end
            n=n+1;
            end
            states3=zeros(states(o,3)+1,3);
            for l=1:1:states(o,3)+1
               states3(l,1)=0;
               states3(l,2)=l-1;
               states3(l,3)=states(o,3)-(l-1);
            end
            Cartesian_product_half=setCartesian(states1,states2);
            Cartesian_product_full=setCartesian(Cartesian_product_half,states3);
            sum_1=sum(Cartesian_product_full(:,1:3:7),2);
            sum_2=sum(Cartesian_product_full(:,2:3:8),2);
            sum_3=sum(Cartesian_product_full(:,3:3:9),2);
            indices=[Cartesian_product_full sum_1 sum_2 sum_3];
            number=size(indices,1);
            probabilities=zeros(number,1);
            for number1=1:1:number
               probabilities(number1)=...

nchoosek(states(o,1),indices(number1,1))*pdcc^(indices(number1,2))*...
              pdcdc^(indices(number1,1))*...
```

```
(factorial(states(o,2))/(factorial(indices(number1,4))*factorial(indices(number1,5
))*factorial(indices(number1,6))))*...

pco_new(A,P,V0,K,pcdc,voltage_steady(indices(number1,10)+number_1))^(indices(numbe
r1,4))*...

pcc_new(A,P,V0,K,pcdc,voltage_steady(indices(number1,10)+number_1))^(indices(numbe
r1,5))*...
                pcdc1_variable^(indices(number1,6))*...
                nchoosek(states(o,3),indices(number1,9))*...

poc(A,P,V0,K,voltage_steady(indices(number1,10)+number_1))^(indices(number1,8))*..
.

poo(A,P,V0,K,voltage_steady(indices(number1,10)+number_1))^(indices(number1,9));

            end
            probabilities2=zeros(28,1);
            for number1=1:1:number
                for l=1:1:28
                   if indices(number1,10)==states(l,1) &&
indices(number1,11)==states(l,2)
                     probabilities2(l)=probabilities2(l)+probabilities(number1);
                   end
                end
            end
            for j=1:1:28
              matrix(o,j)=probabilities2(j);
            end
            number_1=number_1+28;
            o=o+1;
       end
     end
  matrix_of_a_left_hemichannel=formationOfMatrix12_3(A1_variable,...

P1_variable,V01_variable,K1_variable,pcdc1_variable,pdcc1_variable,pdcdc1_variable
,V_left_steady');
  matrix_of_a_right_hemichannel=formationOfMatrix12_3(A2_variable,...

P2_variable,V02_variable,K2_variable,pcdc2_variable,pdcc2_variable,pdcdc2_variable
,V_right_steady');
  matrix_of_both_hemichannels=zeros(28*28);
  size_of_left_matrix=size(matrix_of_a_left_hemichannel,1);
  size_of_left_right_matrix=size(matrix_of_both_hemichannels,1);
  for i=1:size_of_left_matrix:size_of_left_right_matrix
    for j=1:size_of_left_matrix:size_of_left_right_matrix
     matrix_of_both_hemichannels(i:i+(size_of_left_matrix-
1),j:j+(size_of_left_matrix-1))=...
     matrix_of_a_left_hemichannel((i+(size_of_left_matrix-
1))/size_of_left_matrix,...
      (j+(size_of_left_matrix-
1))/size_of_left_matrix)*matrix_of_a_right_hemichannel;
    end
  end
  matrix_of_both_a=matrix_of_both_hemichannels;
     function normalised=normalised_matrix(matrix)
        normalised=zeros(size(matrix,1));
        sum_of_p_rows=sum(matrix,2);
        for i=1:1:length(sum_of_p_rows)
         normalised(i,:)=matrix(i,:)/sum_of_p_rows(i);
        end
     end
  matrix_of_both=normalised_matrix(matrix_of_both_a);
```

```
  X=stationary_probabilities(matrix_of_both);
  conductance=zeros(28*28,1);
   right=1;
   number=28;
   counter=1;
    while right<=number
         for left=1:1:number
         conductance(counter)=number_of_channels*X(counter)*...
        (g_left_hemichannel_3(P1_variable,V_left_steady(1,counter),...
         states(left,1),states(left,2),states(left,3))*...
         g_right_hemichannel_3(P2_variable,V_right_steady(1,counter),...
         states(right,1),states(right,2),states(right,3)))/...
        (g_left_hemichannel_3(P1_variable,V_left_steady(1,counter),...
         states(left,1),states(left,2),states(left,3))+...
         g_right_hemichannel_3(P2_variable,V_right_steady(1,counter),...
         states(right,1),states(right,2),states(right,3)));
         counter=counter+1;
         end
    right=right+1;
    end
 n=0:1:((28*28)-1); n=n';
 table=[n [V_left_steady' V_right_steady'] conductance];
% % % % % For drawing the graphic conductance(V), when V_variable change
% % values from the specified interval
function Cartesian=setCartesian(states_1,states_2)
       one=sortrows(repmat(states_1,[size(states_2,1) 1]));
       second=repmat(states_2,[size(states_1,1) 1]);
       Cartesian=[one second];
end
minimum=V_min_variable; max=V_max_variable; step=V_step_variable;
    count_voltages=numel(minimum:step:max);
    conductance_interval=zeros(count_voltages,1);
    index=1;
for V_variable_variates=minimum:step:max
 right=1;
   number=28;
   counter=1;
    while right<=number
         for left=1:1:number
         V_left_hemichannel_2(1,counter)=(V_variable_variates*...
         g_right_hemichannel_3(P2_variable,0,...
         states(right,1),states(right,2),states(right,3)))/...
        (g_left_hemichannel_3(P1_variable,0,...
         states(left,1),states(left,2),states(left,3))+...
         g_right_hemichannel_3(P2_variable,0,...
         states(right,1),states(right,2),states(right,3)));
         V_right_hemichannel_2(1,counter)=V_variable_variates-
V_left_hemichannel(1,counter);
         counter=counter+1;
         end
    right=right+1;
    end
    for t=2:1:5
       right=1;
      number=28;
      counter=1;
       while right<=number
         for left=1:1:number
       V_left_hemichannel_2(t,counter)=(V_variable_variates*...
       g_right_hemichannel_3(P2_variable,V_right_hemichannel(t-1,right),...
       states(right,1),states(right,2),states(right,3)))/...
      (g_left_hemichannel_3(P1_variable,V_left_hemichannel(t-1,left),...
       states(left,1),states(left,2),states(left,3))+...
       g_right_hemichannel_3(P2_variable,V_right_hemichannel(t-1,right),...
```

```
       states(right,1),states(right,2),states(right,3)));
       V_right_hemichannel_2(t,counter)=V_variable_variates-
V_left_hemichannel(t,counter);
       counter=counter+1;
         end
       right=right+1;
       end
    end
    V_left_steady_2=V_left_hemichannel_2(5,1:1:28*28);
    V_right_steady_2=V_right_hemichannel_2(5,1:1:28*28);

matrix_of_a_left_hemichannel_2=formationOfMatrix12_3(A1_variable,P1_variable,...

V01_variable,K1_variable,pcdc1_variable,pdcc1_variable,pdcdc1_variable,V_left_stea
dy_2');

matrix_of_a_right_hemichannel_2=formationOfMatrix12_3(A2_variable,P2_variable,...

V02_variable,K2_variable,pcdc2_variable,pdcc2_variable,pdcdc2_variable,V_right_ste
ady_2');
  matrix_of_both_hemichannels_2=zeros(28*28);
  size_of_left_matrix_2=size(matrix_of_a_left_hemichannel_2,1);
  size_of_left_right_matrix_2=size(matrix_of_both_hemichannels_2,1);
  for i=1:size_of_left_matrix_2:size_of_left_right_matrix_2
    for j=1:size_of_left_matrix_2:size_of_left_right_matrix_2
     matrix_of_both_hemichannels_2(i:i+(size_of_left_matrix_2-
1),j:j+(size_of_left_matrix_2-1))=...
     matrix_of_a_left_hemichannel_2((i+(size_of_left_matrix_2-
1))/size_of_left_matrix_2,...
      (j+(size_of_left_matrix_2-
1))/size_of_left_matrix_2)*matrix_of_a_right_hemichannel_2;
    end
  end
  X_2=stationary_probabilities(matrix_of_both_hemichannels_2);

   states=zeros(28,3);
   n=1; nnn=0;
   while n<8
     for nn=1:1:7-(n-1)
     states(nnn+1,1)=n-1; states(nnn+1,2)=nn-1; states(nnn+1,3)=6-(n-1)-(nn-1);
     nnn=nnn+1;
     end
    n=n+1;
   end
   Cartesian_product_1=setCartesian(states,states);
   sum_1=sum(Cartesian_product_1(:,1:2),2);
   sum_2=sum(Cartesian_product_1(:,4:5),2);
   sum_3=sum_1+sum_2;
   statmatrix_0=[Cartesian_product_1 sum_1 sum_2 sum_3 X_2];
   number=size(statmatrix_0,1);
   X_20=zeros(13,1);
       for number1=1:1:number
          for l=1:1:13
            if statmatrix_0(number1,9)==(l-1)
              X_20(l)=X_20(l)+statmatrix_0(number1,10);
            end
          end
       end
       for j=1:1:13
          statmatrix(j,index)=abs(X_20(j));
       end

  conductance_2=zeros(28*28,1);
```

```
   right=1;
   number=28;
   counter=1;
    while right<=number
         for left=1:1:number
         conductance_2(counter)=number_of_channels*X_2(counter)*...
        (g_left_hemichannel_3(P1_variable,V_left_steady_2(1,counter),...
         states(left,1),states(left,2),states(left,3))*...
         g_right_hemichannel_3(P2_variable,V_right_steady_2(1,counter),...
         states(right,1),states(right,2),states(right,3)))/...
        (g_left_hemichannel_3(P1_variable,V_left_steady_2(1,counter),...
         states(left,1),states(left,2),states(left,3))+...
         g_right_hemichannel_3(P2_variable,V_right_steady_2(1,counter),...
         states(right,1),states(right,2),states(right,3)));
         counter=counter+1;
         end
    right=right+1;
    end
 conductance_interval(index)=sum(conductance_2);
 index=index+1;
end

 figure_handle_0=figure();
  table_handle = uitable('Position', [100,340,450,300],'Data',table,...
                         'ColumnName', {'State','Voltage of the Left
Hemichannel','Voltage of the Right Hemichannel' , 'Conductance'},...
                         'ColumnFormat', {'numeric','numeric','numeric'},...
                         'ColumnWidth', {80 150 150 80});
  bar_handle =
bar(axes('Units','Pixels','Position',[600,350,600,300]),table(:,1),table(:,4));
  title 'Bar of a conductance of a channel (depending on a state)';
  xlabel('a state of left and right hemichannels, const.')
  ylabel('Conductance of a channel, pS')
  set(bar_handle);
  set(table_handle,'FontName','Courier','FontSize',12);
  voltage_interval=minimum:step:max; voltage_interval=voltage_interval';
  table_handle_2 = uitable('Position', [100,50,210,260],'Data',[voltage_interval
conductance_interval],...
                        'ColumnName', {'Voltage','Conductance'},...
                        'ColumnFormat', {'numeric','numeric'},...
                        'ColumnWidth', {80 80});
  plot_handle =
plot(axes('Units','Pixels','Position',[600,60,300,220]),voltage_interval,conductan
ce_interval,'LineWidth',2);
  set(table_handle_2,'FontName','Courier','FontSize',12);
  title 'Conductance dependence on voltage';
  xlabel('V, mV')
  ylabel('G(V), pS')
  axis([-100 100 0 7]);
  set(plot_handle);
  grid on;
  set(figure_handle_0,'Units','normalized','NumberTitle',
'off','Menubar','none','Name','Model of 12 Connexins','Color',defaultBackground);
  figure_handle_1=figure();
  subplot(1,2,1),
plot(voltage_interval,statmatrix(1,:)','b',voltage_interval,statmatrix(2,:)','r',.
..

voltage_interval,statmatrix(3,:)','g',voltage_interval,statmatrix(4,:)','y',...

voltage_interval,statmatrix(5,:)','k',voltage_interval,statmatrix(6,:)','c',...
                       voltage_interval,statmatrix(7,:)','m','LineWidth',2);
  xlabel('Voltage (V), mV')
  ylabel('Probability')
```

```
  legend('0 closed connexins','1 closed connexin','2 closed connexins',...
         '3 closed connexins','4 closed connexins','5 closed connexins','6 closed
connexins'); grid on;
  subplot(1,2,2),
plot(voltage_interval,statmatrix(8,:)','b',voltage_interval,statmatrix(9,:)','r',.
..

voltage_interval,statmatrix(10,:)','g',voltage_interval,statmatrix(11,:)','y',...

voltage_interval,statmatrix(12,:)','k',voltage_interval,statmatrix(13,:)','c','Lin
eWidth',2);
  xlabel('Voltage (V), mV')
  ylabel('Probability')
  legend('7 closed connexins','8 closed connexin','9 closed connexins',...
         '10 closed connexins','11 closed connexins','12 closed connexins'); grid
on;
  set(figure_handle_1,'Units','normalized','NumberTitle',
'off','Menubar','none','Name','Dependency of
Probabilities','Color',defaultBackground);
end

function aboutbutton_Callback(source,data)
 message_handle=msgbox({'This is the program of gap junction models (2 states and
3 states).';...
     '@Aurelija Sakalauskaitė, FMMM-9 grupės studentė'},'About GJ Model');
 set(message_handle,'Units','normalized','Color',defaultBackground);
end

function exitbutton_Callback(source,data)
    close all;
end
end
```