Slovenská technická univerzita
Fakulta elektrotechniky a informatiky
Katedra informatiky a výpočtovej techniky
Odbor: INFORMATIKA

Pavol Lupták

Vstupno-výstupné charakteristiky impulzných neurónov

Záverečný projekt

Vedúca projektu: RNDr, Ľubica Beňušková, CSc
Dátum odovzdania: 18. mája 2001

Chcel by som sa poďakovať pani RNDr. Ľubici Beňuškovej za postrehy, rady a pripomienky k tomuto projektu.

ANOTÁCIA

Slovenská technická univerzita v Bratislava
Fakulta Elektrotechniky a Informatiky
Študijný odbor: INFORMATIKA

Autor: **Pavol Lupták**
Záverečný projekt: **Vstupno-výstupné charakteristiky impulzných neurónov**
Vedúci záverečného projektu: **RNDr. Ľubica Beňušková, CSc**
máj 2000

Cieľom mojho projektu bolo navrhnúť a implementovať simuláciu a vizualizáciu vstupno-výstupných charakteristík impulzných neurónov. Impulzné neuróny predstavujú tretiu generáciu modelu neurónov snažiacich sa čo najpresnejšie zachytiť správania biologických neurónov. Simulácia umožňuje odhadnuť správanie sa neurónu na základe znalostí jeho presynaptických vstupov. Vizualizácia dokáže graficky znázorniť spôsob správania sa neurónu na zvolenom časovom intervale.

Projekt bol plne objektovo špecifikovaný a navrhnutý v jazyku C++ použitím univerzálnej, robustnej, platformovo nezávislej grafickej knižnici QT.

Testovaný bol na OS Linux v systéme okien XFree86.

ANNOTATION

Slovak University of Technology, Bratislava
Faculty of Electrical Engineering
and Information Technology

Degree Course: INFORMATICS

Author: **Pavol Lupták**
Thesis: **Input-output characteristics of impulse neurons**
Supervisor: **RNDr. Ľubica Beňušková, CSc**
máj 2000

The purpose of my project was to design and implement the complete simulation and visualisation of input-output characteristics of impulse neurons. Impulse neurons represent the third generation of the most precise mathematic model of neurons describing behaviour of biological neurons. The simulation allows to estimate the behaviour of neurons using the knowledge related to its presynaptic inputs. The visualisation allows to display a behaviour of neuron on the specified time interval.

This project is fully object-oriented and designed in C++ using universal, robust, platform independent GUI toolkit QT.

The application was tested on OS Linux using XFree86.

Obsah

1 Úvod

Organizmus interagujúci s okolím musí byť schopný prijímania a spracovávania vstupných vnemov (rozpoznávanie potenciálnych zdrojov jedla, predátorov). V prípade, že by sa nám podarilo skonštruovať robota reagujúceho na vyššie popísane vnemy, nereagoval by s takou ľahkosťou a flexibilitou ako poniektoré zvieratá. Za túto úžasnú vlastnosť vďačíme neurónovému systému - mozgu, ktorý sa za dlhé evolučné obdobie niekoľkonásobne 'optimalizoval'. Umelé neurónové systémy komunikujúce pomocou frekvenčného kódu veľakrát vystihujú biologické systémy horšie, ako neurónové systémy založené na impulznom kóde.

Impulzný neurón predstavuje jednoduchý a celkom presný model biologického neurónu. Simuláciou a vizualizáciou neurónov sa zaoberá viacero existujúcich neurónových simulátorov[1].

Simuláciou impulzných neurónov sa zaoberá množstvo nezávislých prác, ja som vychádzal z prác Christopher M.Bishopa a Wolfganga Maassa[2].

Použiteľný voľne prístupny simulačný systém špecifický na simuláciu a vizualizáciu impulzných neurónov som zatiaľ nenašiel, systém GENESIS pokrýva impulzné neuróny len zčasti.

Mojim cieľom bolo navrhnúť a implementovať základnú jednoduchú simuláciu a vizualizáciu charakteristík impulzného neurónu pod OS Linuxom. Výsledkom projektu je **platformovo nezávisla** aplikácia [3] s plne objektovým návrhom, silnou simuláciou a jednoduchou vizualizáciou. Dôraz bol kladený na návrh simulačných tried reprezentujúcich impulzný neurón.

[1] Veľmi populárny je GENESIS, http://www.bbb.caltech.edu/GENESIS/whatisit.html
[2] Cambridge and Graz University
[3] Použitá multiplatformová grafická knižnica QT, http://www.trolltech.com

2 Analýza problému

2.1 Neurónový kód

Napriek detailným informáciám o neurónoch a ich spojeniach, neuroveda si stále kladie zatiaľ nezodpovedanú otázku: Komunikujú neuróny pomocou frekvenčného kódu (*rate code*) alebo pomocou impulzného kódu (*impulse code*[4])? Za posledného storočie, biologický výskum získal enormné množstvo detailných informácií o štruktúre a funkciách mozgu (Kandel a Schwartz, 1991). Na neurónoch rozlišujeme trojuholníkový alebo kruhovitým tvar tela bunky a dlhé drôtovité zakončenia. Typický neurón ma tri časti nazývané dendrický strom, soma a axón. Signály prichádzajúce od ostatných neurónov na dendrický strom sa prenášajú na somu a axón. Prenosová oblasť prenosu medzi somou a axónom je veľmi zaujímavá (prejavuje sa tu nelineárny proces). Ak je dosiahnutý prah excitácie (zapríčinený dostatočným vstupom), vyšle sa výstupný signál, ktorý sa šíri po axóne a jeho zakončeniach do ostatných neurónov. Spojenie medzi axonálnou vetvou a dendritom (alebo somou) neurónu, ktorý prijíma sa nazýva synapsa. Neurón, ktorý posiela si označíme ako `presynaptický neurón`, ktorý prijíma bude `postsynaptický neurón`. Neurón v kortexe je prepojený z viac ako 10^4 postsynaptickými neurónmi. Množstvo jeho axonálnych vetiev je zakončených v priamom susedstve neurónu, ale axón môže byť natiahnutý až na niekoľko milimetrov a spájať neuróny v iných častiach mozgu. Neurónové signály sa dajú zachytiť umiestnením citlivej elektródy blízko somy alebo axónu neurónu. Na napäťovovej stope v typickom zázname vidíme sekvenciu krátkych impulzov nazývaných tiež akčné potenciály alebo spajky (*spike*). Reťazec impulzov emitovaných jedným neurónom sa zvykne nazývať `séria impulzov` (*spike train*) (sekvencia stereotypných udalostí opakujúca sa v pravidelných alebo nepravidelných intervaloch). Čas trvania akčného potenciálu je typicky 1-2 ms. Vzhľadom na to, že všetky impulzy daných neurónov vyzerajú rovnako, v tvare akčného potenciálu nie je zakódovaná žiadna informácia. Tá je zakódovaná v množstve a časovaní impulzov. Moment, kedy daný neurón vyšle impulz (akčný potenciál) sa nazýva tiež čas odpálenia (*firing time*) neurónu. Čas odpálenia neurónu i označíme ako $t_i^{(f)}$. Reťazec impulzov i-teho neurónu je teda charakterizovaná množinou časov odpálení:

$$F_i = \{t_i^{(1)}, ..., t_i^{(n)}\} \tag{1}$$

kde $t_i^{(n)}$ je posledný impulz neurónu i. Experimentálne sa odpaľovacie časy merajú s krokom Δt. Séria impulzov (spike train) môže byť popísaná ako sekvencia jednotiek

[4]Ref(1,2) v použitej literatúre

a núl pre 'impulz' a 'žiadny impulz' v časoch $\Delta t, 2\Delta t, ...$, atď. Voľba jednotiek a núl je samozrejme náhodná. Rovnako dobre môžeme udalost impulzu označiť číslom $1/\Delta t$. S takouto definícou, séria impulzov neurónu i korešponduje zo sekvenciou čísel $S_i(\Delta t), S_i(2\Delta t), ...$ s

$$S_i(n\Delta t) = \begin{cases} 1/\Delta t & ak \quad n\Delta t \leq t_i^{(f)} < (n+1)\Delta t \\ 0 & inak \end{cases} \tag{2}$$

Formálne, môžeme zobrať limitu $\Delta t \to 0$ a zapísať reťazec impulzov ako sekvenciu δ funkcií:

$$S_i(t) = \sum_{t_i^{(f)} \in F_i}^{n} \delta(t - t_i^{(f)}) \tag{3}$$

kde $\delta(.)$ označuje Diracovu δ funkciu s $\delta(s) = 0$ pre $s \neq 0$ a $\int_{-\infty}^{\infty} \delta(s)ds = 1$. Nahliadli sme na impulznú reťaz jednoduchého neurónu. Vzhľadom k tomu, že sa v mozgu nachádza veľmi veľa neurónov, tisícky reťazí sú konštantne vysielané odlišnými neurónmi. Aká informácia je obsiahnutá v časovo-priestorovom zázname impulzov? Ako môžu iné neuróny dekódovať signál? Môžeme ako externí pozorovatelia čítať tento kód a pochopiť odkaz v zázname neurónovej aktivity? V súčasnosti na to nevieme jednoznačnú odpoveď. Tradičný názor je taký, že relevantná informácia je uložená v priemernej odpaľovacej frekvencii (*mean firing rate*) neurónu. Odpaľovacia frekvencia je obvykle definovaná časovým priemerom. Experimentátori nastavia časové okno T=100 ms alebo T=500 ms a spočítajú množstvo impulzov $n_{sp}(T)$, ktoré sa objavia v tomto čase. Po predelení dĺžkou časového okna dostávame priemernú odpaľovaciu frekvenciu:

$$v = \frac{n_{sp}(T)}{T} \tag{4}$$

Zvyčajne udávanú v jednotkách s^{-1} alebo Hz. Koncept priemerných odpaľovacích frekvencií bol úspešne aplikovaný počas posledných 80-tich rokov (začiatok sa datuje prácam Adriana v rokoch 1926-1928, ktorý ukázal, že odpaľovacia frekvencia receptoru naťahovania vo svaloch je úmerná sile aplikovanej do svalu). V nasledujúcich desaťročiach sa merania odpaľovacích frekvencií stali štandardným prostriedkom na popis všetkých typov senzorov alebo kortinálnych neurónov. Je však jasné, že priblíženia založené na časovom priemere zanedbávajú akúkoľvek možnú informáciu obsiahnutú v presnom načasovaní impulzov. Niet preto divu, že koncept odpaľovacích frekvencií bol opakovane kritizovaný a prediskutovávaný. Počas posledných rokov sa nahromadilo stále viac a viac experimentov, ktoré hovoria o tom, že priamočiary koncept odpaľovacích frekvencií založený na časovom priemere je príliš jednoduchý na popis mozgovej aktivity. Jeden z hlavných argumentov je, že reakčné časy v ex-

perimentoch so správaním trvajú príliš krátko nato, aby povoľovali pomalé časové priemery.

2.2 Kódovanie časom do prvého impulzu

Pozrime sa na neurón, ktorý náhle dostane nový vstup v čase t_0. Keď sa pozrieme na nejaký obraz, skáčeme pohľadom z jedného bodu na druhý. Po každom kmitnutí oka, získame na fotoreceptore v sietnici nový vizuálny vstup. Informácia o čase kmitnutí oka t_0 je v mozgu ľahko prístupná. Môžeme si teda predstaviť kód, kde pre každý neurón, čas prvého impulzu od času t_0 obsahuje všetku informáciu o novom podnete. Neurón, ktorý vyšle impulz krátko po čase t_0 znamená silnú stimuláciu, vyslanie impulzu o čosi neskôr, znamená slabšiu stimuláciu. V konečnej verzii tejto kódovacej schémy, počítame teda len prvý impulz každého neurónu. Všetky ostatné impulzy su teda pre nás irelevantné. Môžeme sa teda domnievať, že každý neurón vysiela presne jeden impulz počas kmitnutia oka a neskôr je ukončený inhibičným vstupom. Je teda jasné, že v tomto prípade na kódovaní informácie sa podieľajú len časy dopravy impulzov a nie ich množstvo. Je pravda, že uvedená schéma 'Čas do prvého impulzu' je zidealizovaná. S. Thorpe (Thorpe et al., 1996) argumentuje, že mozog nemá dostatočne veľa času na vyhodnotenie viac ako jedného impulzu od každého neurónu na jeden krok. Preto by mal prvý impulz obsahovať najviac dôležitej informácie. Použitím informačno-teoretických meraní na experimentálnych dátach, viaceré vedecké skupinky ukázali, že najviac informácie o novom podnete sa dopraví počas prvých 20 alebo 50 milisekúnd po začiatku neurónovej odpovede. Rýchly výpočet prechodov po prijatí nového podnetu sa tiež diskutuje v štúdiach modelu (Hopfield a Herz, 1995; Tsodyks a Sejnowsky, 1995; van Vreeswijk a Sompolinsky, 1997).

2.3 Kódovanie fázou

Schému 'Kódovanie časom do prvého impulzu' môžeme aplikovať tiež v prípade, kde referenčný signál nie je jednorázová udalosť, ale opakovaný signál. V hipokampe ako aj v ostatných častiach mozgu, oscilácie nejakej globálnej premennej (napríklad populačnej aktivity) sú celkom bežné. Tieto oscilácie môžu poskytovať vnútornú referenciu signálu. Neurónové impulzné reťazce môžu kódovať informáciu vo fáze impulzu s ohľadom na inú osciláciu v pozadí. Ak sa vstup nemení medzi jedným cyklom a nasledujúcim, tak sa zrejme opakuje rovnaký druh fáz. Koncept kódovanie fázami študovalo niekoľko skupín (Hopfield, 1995; Jensen a Lisman, 1996; Maas, 1996) a experimentálne (O'Keefe a Recce, 1993). Je evidentný dôkaz, že fáza neurónov počas oscilácie v

hipokampe potkanov prenáša priestorovú informáciu (polohu) zvieraťa, ktorá nie je zachytená v samotnej odpaľovacej frekvencii neurónu (O'Keefe a Recce, 1993).

2.4 Korelácie a synchronizácie

Impulzy z ostatných neurónov môžeme tiež použiť ako referenčný signál pre impulzný kód. Napríklad, synchronizácia medzi dvojicou alebo väčšou skupinou neurónov môže znamenať špeciálne udalosti a prenášať informáciu, ktorá nie je obsiahnutá v odpaľovacej frekvencii neurónov. Uvažujme napríklad komplexnú scénu pozostávajúcu z viacerých objektov. V mozgu spôsobí aktivitu väčšieho množstva neurónov. Neurónmi reprezentujúcimi jeden rovnaký objekt označíme také, ktoré vysielajú impulz synchrónne (Malsburg, 1981; Malsburg a Buhmann, 1992, Eckhorn et al., 1988; Gray et al., 1989). Keď sa na to pozrieme všeobecne, nielen synchronizácia, ale tiež presný časovo-priestorový vzorec impulzov môže byť doležitý pri prenose informácie. Napríklad impulzný vzorec 3 neurónov, kedy neurón č.1 vyšle impulz v ľubovoľnom čase t_1, po ňom vyšle impulz neurón č.2 v čase $t_1 + \delta_{12}$ a neurón č.3 v čase $t_1 + \delta_{13}$ môže reprezentovať istý podnet. Dôležitosť časovo-priestorových impulzných vzorcov intenzívne študovali Abeles (Abeles, 1991; Abeles et al., 1993; Abeles, 1994).

2.5 Neurónové modely

Neurónová aktivita môže byť popísaná na viacerých úrovňach abstrakcie. Na mikroskopickej úrovni je veľké množstvo iónových kanálov, pórov v membránach buniek ktoré sa otvárajú a zatvárajú v závislosti od napätia a prítomnosti (alebo absencie) rozličných chemických molekúl. Na vyššej úrovni abstrakcie, kedy nás netrápi priestorová štruktúra neurónu alebo presný iónový mechanizmus, neurón uvažujeme ako homogénnu jednotku, ktorá generuje impulzy, ak je dosiahnutý prah excitácie. Impulzné neurónové modely by mali byť v kontraste s frekvenčnými. Frekvenčné modely zanedbávajú impulznú štruktúru neurónového výstupu, a preto sú na na vyššej úrovni abstrakcie.

2.6 Jednoduchý impulzný neurónový model

Spike Response Model - definície Stav neurónu i je popísaný stavovou premennou u_i. Neurón vyšle impulz, ak u_i dosiahnutie prah excitácie ϑ (*threshold*). Množinu všetkých odpaľovacích časov neurónu i označujeme ako

$$F_i = \{t_i^{(f)}; 1 \leq f \leq n\} = \{t | u_i(t) = \vartheta\} \tag{5}$$

Posledný impulz $t_i^{(f)} < t$ neurónu i označíme tiež $t_i^{(n)}$. Hodnotu stavovej premennej u_i ovplyvňujú dva odlišné procesy.

Prvý nastáva hneď po vyslaní výstupného impulzu v čase $t_i^{(f)}$, hodnota premennej u_i poklesne (tzv. reset). Matematicky sa to realizuje pridaním negatívneho príspevku $\eta_i(t - t_i^{(f)})$ k stavovej premennej u_i.

Druhý nastáva, keď model neurónu začne prijímať vstup z presynaptických neurónov $j \in \Gamma_i$, kde $\Gamma_i = \{j | j$ presynaptický k i$\}$.

Presynaptický impulz v čase $t_j^{(f)}$ zvačšuje (alebo zmenšuje) hodnotu u_i neurónu i pre $t > t_j^{(f)}$ o veľkosť $w_{ij}\epsilon_{ij}(t - t_j^{(f)})$. Váha w_{ij} je faktor, ktorý závisí od 'sily' spojenia. Efekt presynaptického impulzu môže byť pozitívny (excitačný) alebo negatívny (inhibičný). Hodnota jadra (*kernel*) $\epsilon_{ij}(s)$ sa musí približovať k 0 pre $s \leq 0$. Pauza počas prenosu môže byť vnorená v definícii $\epsilon_{ij}(s)$.

Stav $u_i(t)$ modelu neurónu i v čase t je daný lineárnou superpozíciou všetkých príspevkov,

$$u_i(t) = \sum_{t_i^{(f)} \in F_i} \eta_i(t - t_i^{(f)}) + \sum_{j \in \Gamma_i} \sum_{t_j^{(f)} \in F_j} w_{ij}\epsilon_{ij}(t - t_j^{(f)}) \tag{6}$$

Interpretácia výrazov na pravej strane je zjavná. η_i príspevky popisujú odpoveď neurónu i na vlastné impulzy . ϵ_{ij} jadrá modelu neurónov zodpovedajú odpovedi na

presynaptické impulzy.

Stavovú premennú u_i môžeme tiež interpretovať ako potenciál elektrickej membrány, jadrá ϵ_{ij} ako postsynaptické potenciály a η_i príspevky ako neuronálnu refraktérnosť.

Uvažujme príklad vhodných funkcií η_i a ϵ_{ij}. η_i obvykle nie je kladné pre $s > 0$.

Matematická formulácia môže vyzerať

$$\eta_i(s) = -\vartheta exp\left(-\frac{s}{\tau}\right) H(s) \tag{7}$$

kde τ je časová konštanta a H(s) Heavisideova funkcia, ktorá ide k 0 pre $s \leq 0$ a k 1 pri $s > 0$. Efekt tejto formulácie je taký, že po každom vyslaní impulzu stavová premenná u_i sa nastaví na ϑ.

Jadrá ϵ_{ij} popisujú odpoveď na presynaptické impulzy. Pre excitačné synapsy, jadro ϵ_{ij} je nezáporné a nazýva sa excitačný postsynaptický potenciál (EPSP). Pre inhibičné synapsy jadro nadobúda nekladné hodnoty a nazýva sa inhibičný postsynaptický potenciál (IPSP). Jedna z možných matematických formulácií jadra je

$$\epsilon_{ij}(s) = \left[exp\left(-\frac{s-\Delta^{ax}}{\tau_m}\right) - exp\left(-\frac{s-\Delta^{ax}}{\tau_s}\right)\right] H(s-\Delta^{ax}) \tag{8}$$

kde τ_s, τ_m sú časové konštanty a Δ^{ax} je axonálne prenosové oneskorenie. Amplitúda odpovede je škálovaná faktorom w_{ij}. Pre inhibičné synapsy, bude mať jadro ϵ_{ij} záporné znamienko (na začiatku výrazu) na pravej strany formuly.

Z matematického pohľadu, sieť impulzných neurónov je počítaná funkciou, ktorá mapuje vektor časových sérií $< F_i >_{i \in I_{input}}$ na vektor iných časových sérií $< F_i >_{i \in I_{output}}$. Častokrát sa vstupy a výstupy neurónovej siete pokladajú skôr za časové série (*time series*), ako nejaké vektory čísel (obvyklé v tradičných neurónových modeloch). Biologické organizmy musia byť schopné veľmi rýchlo odpovedať na stále sa meniace prostredie, v ktorom sa nachádzajú. Vzhľadom na to, že umelé verzie impulzných neurónových sietí sú dobre uspôsobené na výpočet časových sérií, v praxi sa jednoduchý neurónový impulzný model používa na návrh umelých impulzných neurónových sietí.

2.6.1 Dynamický model prahu excitácie (*Dynamic threshold model*)

Uvažujme podmienku $u_i(t) = \vartheta$, kedy je dosiahnutý prah excitácie. Zo vzťahu na popis jednoduchého impulzného neurónového modelu dostaneme

$$\sum_{j \in \Gamma_i} \sum_{t_j^{(f)} \in F_j} w_{ij}\epsilon_{ij}(t - t_j^{(f)}) = \vartheta - \sum_{t_i^{(f)} \in F_i} \eta_i(t - t_i^{(f)}) \tag{9}$$

kde sme prehodili sumu η_i na pravú stranu. O celej pravej strane môžeme uvažovať ako o dynamickom prahu excitácie, ktorý sa zväčšuje po každom vyslaní impulzu a pomaly zmenšuje naspäť k asymptotickej hodnote ϑ v prípade nevysielania impulzu neurónu i.

2.6.2 Externý vstup

Okrem impulzného vstupu od ostatných neurónov, môže neurón prijímať čisto analógový vstup $I^{ext}(t)$ napríklad od neimpulzného senzorového neurónu. V tomto prípade, na pravú stranu rovnice pridáme výraz:

$$h^{ext}(t) = \int_0^\infty \epsilon(s) I^{ext}(t-s) ds \tag{10}$$

kde $\epsilon(s)$ je ďalšie jadro, ktoré popisuje odpoveď potenciálu membrány na externý pulzný vstup. Zavedieme si novu premennú h, ktorá bude označovať všetky príspevky od ostatných neurónov a externých zdrojov:

$$h(t) = \sum_{j \in \Gamma_i} w_{ij} \sum_{t_j^{(f)} \in F_i} \epsilon_{ij}(t - t_j^{(f)}) + h^{ext}(t) \tag{11}$$

Membránový potenciál neurónu sa bude dať teda vyjadriť v skratke:

$$u_i(t) = \eta_i(t - t_i) + h_i(t) \tag{12}$$

2.7 Kódovanie použitím impulzov

Uvažujeme doposiaľ spomenuté impulzné kódy.

2.7.1 Čas do prvého impulzu

Uvažujme jednoduchý neurón i, ktorý prijíma impulzy z N presynaptických neurónov j cez synaptické spojenia, ktoré majú rovnakú váhu $w_{ij} = w_0$. Nemáme žiadny vstup. Uvažujeme, že posledný impulz neurónu i bol vyslaný veľmi dávno, takže impulzný postpotenciál $\eta(.)$ môžeme zanedbať. V čase $t = t^{pre}$, je simultánne generovaných a posielaných postsynaptickému neurónu celkové množstvo $n_1 < N$ presynaptických impulzov. Pre $t > t^{pre}$ je potenciál i-teho neurónu $u_i(t) = n_1 w_0 \epsilon(t - t^{pre})$. Neurón vyšle výsledný impulz, vždy, keď u_i dosiahne prah excitácie ϑ. Odpaľovací čas $t_i^{(f)}$ prvého neurónu uvažujeme ako

$$t_i^{(f)} = min\{t > t^{pre} | u_i(t) = \vartheta\} \tag{13}$$

Časový rozdiel $t_i^{(f)} - t^{pre}$ je meradlo množstva presynaptických impulzov. V načasovaní prvého impulzu je zakódovaná 'sila' vstupu.

2.7.2 Kódovanie fázou

Kódovanie fázou je možné len vtedy, ak existuje nejaký periodický signál v pozadí, ktorý slúži ako referencia. Zakomponujeme periodický signál v pozadí do externého vstupu:

$$h^{ext}(t) = h_0 + h_1 \cos\left(2\pi \frac{t}{T}\right) \tag{14}$$

kde h_0 je konštanta a h_1 je amplitúda T-periodického signálu.

2.8 McCullochove-Pittsove neuróny verzus impulzné neuróny

Najjednoduchšia výpočtová jednotka v tradičných neurónových sieťových modeloch je McCullochov-Pittsov neurón, tiež označovaný ako prahový model alebo perceptron. McCullochov-Pittsov neurón i s reálnymi hodnotami váh α_{ij} a prahom excitácie ϑ prijíma ako vstup n binárnych alebo reálnych čísel $x_1, ..., x_n$. Jeho výstup má hodnotu

- 1, ak $\sum_{j=1}^{n} \alpha_{ij}.x_j \geq \vartheta$
- 0, inak

Vo viacvrstvových sieťach uvažujeme rôzne brány prahu excitácie, ktoré označujeme tiež sigmoidálné brány (*sigmoidal gate*). Výstup sigmoidálnej brány je definovaný pomocou nezmenšujúcej spojitej aktivačnej funkcie g: R $\rightarrow$ R s ohraničeným intervalom ako

$$g(\sum_{j=1}^{n} \alpha_{ij}.x_j - \vartheta) \tag{15}$$

Použitím zigmoidálných brán namiesto brán prahu excitácií (*treshold gates*) môžeme okrem výpočtov z analógovým výstupom, použiť aj výpočtovú silu neurónových sietí pre výpočet funkcií s binárnym výstupom (Maas et al, 1991; DasGupta a Schnitger, 1996).

Vyzerá to tak, že impulzný neurón môže principiálne simulovať ľubovoľne danú bránu prahu excitácie s pozitívnym prahom excitácie ϑ pre binárny vstup. Predpokladáme, že jadrá (funkcie odpovedí) $\epsilon_{ij}(x)$ sú všetky rovnaké až na ich znamienko (volíme kladné, ak $\alpha_{ij} > 0$ alebo negatívne ak $\alpha_{ij} \leq 0$) a všetky presynaptické neuróny j vysielajú impulz v rovnakom čase $t_j = T_{input}$. V tomto prípade, impulzný neurón i vyšle impulz len vtedy keď :

$$\sum_{j-vysiela-impulz-v-T_{input}} w_{ij}\epsilon_{ij} \geq \vartheta \tag{16}$$

kde ϵ_{ij} nadobúda extrém $\epsilon_{ij}(s)$,($\epsilon_{ij} = max_s\epsilon_{ij}(s)$ ak ϵ_{ij} predstavuje EPSP, $\epsilon_{ij} = min_s\epsilon_{ij}(s)$ ak ϵ_{ij} predstavuje IPSP), $w_{ij} \geq 0$ je synaptická váha a $\vartheta > 0$ ak je dosiahnutý prah

excitácie neurónu i. Potom impulzný neurón môže simulovať danú bránu excitácie - ak vstupné bity $x_1, .., x_n$ sú zakódované pomocou vysielania/nevysielania presynaptických neurónov j=1, .., n v obvyklom čase T_{input} a výstupný bit brány prahu excitácie je zakódovaný vysielaním/nevysielaním impulzného neurónu i počas pozorovaného časového okna. Keď sa na to pozrieme bližšie, zistíme, že je oveľa ťažšie podobným spôsobom simulovať viac-vrstvovú sieť brán prahov excitácií sieťou impulzných neurónov. Presný odpaľovací čas impulzného neurónu i závisí od konkrétneho vstupu $x_1, ..., x_n$.

Ak $\sum_{j-vysiela-impulz-v-T_{input}} w_{ij}\epsilon_{ij} \geq \vartheta$ nadobudne hodnotu oveľa väčšiu ako 0, potom stavová premenná $u_i(t) = \sum_{j-vysiela-impulz-v-T_{input}} w_{ij}\epsilon_{ij}(t - T_{input})$ dosiahne prah excitácie ϑ skôr, ako v prípade, keď hodnota vstupu $\sum_{j-vysiela-impulz-v-T_{input}} w_{ij}\epsilon_{ij} \geq \vartheta$ je síce pozitívna, ale blízka 0. Časovanie v jednoznačnej simulácií viac-vrstvového obvodu (siete) je nestabilné - aj napriek tomu, že všetky vyslania impulzov v jednej rovine sa realizujú synchrónne, táto synchronizácia sa vo všeobecnosti stratí v ďalšej vrstve. V dôsledku toho potrebujme odlišný synchronizačný mechanizmus na simulovanie viac-vrstvového binárneho obvodu, alebo iný bežný model na digitálne výpočty - sieť impulzných neurónov z bitmi 1 a 0 zakódovanými vyslaním/nevyslaním impulzu.

2.9 Vstupno-výstupne charakteristiky

2.9.1 Simulácia jedného impulzného neurónu

Predmetom záujmu bude sledovanie správania jedného impulzného neurónu (dosiahnutie jeho prahu excitácie na základe odlišných vstupov z presynaptických neurónov), kde bude graficky znázornený priebeh stavovej premennej u_i pre daný neurón i pri menených hodnotách jadra ϵ_{ij} ako aj vlastnej refraktérnosti η_i. Pri dynamickom modeli môžeme sledovať zmenu hodnoty prahu excitácie ϑ. Veličiny týkajúce sa hodnôt presynaptických neurónov budú zadávané používateľom, resp. pri väčšom množstve budu generované náhodné, alebo podľa preddefinovanej funkcie. Množstvo presynaptických vstupov bude tiež voliteľné. Pre názorné príklady bude postačovať relatívne malé množstvo vstupov.

2.9.2 Simulácie impulznej neurónovej siete

Podobne ako pri simulácii jedného impulzného neurónu, predmetom záujmu budú všetky doposiaľ merané a graficky spracované veličiny. Sledovania správania však rozšírime na celú sieť impulzných neurónov, ktoré návzajom môžu (spoločné presynaptické vstupy, rovnaké parametre, váhy, ..) a nemusia súvisieť.

3 Špecifikácia programu

Pri špecifikovaní výslednej aplikácie (**Spikes response simulation**) bol kladený dôraz na

- Prenositeľnosť na ľubovoľný OS
- Plne objektový návrh[5]
- Flexibilnosť a intuitívnosť ovládania
- Robustnosť a ľahká modifikovateľnosť
- Nezávislosť simulačnej a vizuálnej časti

Snažil som sa skĺbiť hore uvedené vlastnosti pri návrhu výslednej simulačno-vizuálnej aplikácii.

3.1 Prenositeľnosť na ľubovoľný OS

Simulačná časť je navrhnutá plne objektovo, bez viazanosti na konkrétnu platformu alebo typ c++ kompilátora.

Vizuálna časť je viazaná na konkrétny grafický adaptér ovládaný prostriedkami príslušného operačného systému na danej platforme a preto zvyknú byť s touto časťou problémy. Vzhľadom k tomu som sa rozhodol postaviť vizuálnu stránku aplikácie na GUI grafickej knižnici QT[6], ktorá je plne objektová a multiplatformová (väčšina UNIX prostredí bežiacich na X11, Microsoft Windows NT, 95/98). Vzhľadom k tomu, že grafická knižnica je pre platformu Microsoft Windows distribuovaná ako komerčný balík, použitá bola voľne prístupna QT knižnica pod OS Linuxom. Po rekompilácii s príslušnym portom QT knižnice ja moja aplikácia spustiteľná prakticky na ľubovoľnom OS, ktorý podporuje QT.

3.2 Plne objektový návrh

Simulačná časť bude pozostávať z univerzálne navrhnutých tried pre ľubovoľný neurón, inheritáciou príslušnej triedy a preťažením príslušných metód získame špecifickú triedu pre presynaptický a postsynaptický neurón. Každý simulovaný neurón je reprezentovaný univerzálnym objektom s definovanými metódami.

Vizuálna časť používa inheritované objekty QT grafických widgetov, okien a všetkých ostatných grafických prostriedkov podporovaných touto knižnicou. Všetky

[5] Ref(5) v použitej literatúre
[6] Ref(4) v použitej literatúre

grafické operácie sa vykonávajú nad konkrétnym objektom, jednotlivé objekty komunikujú pomocou signálov a slotov.

3.3 Flexibilnosť a intuitívnosť ovládania

Snaha o ovládanie aplikácie intuitívnym spôsobom, v prípade, že používateľ rozumie skúmanej problematike. Nie je nutný "help", nakoľko navrhnuté okná sa volajú všetky z hlavného menu a všetky nastavované parametre sa týkaju pre používateľa známej oblasti. Aplikácia umožňuje simulované a pozorované údaje ukladať a spätne nahrávať. Simulované a pozorované údaje predstavujú všetky matematické konštanty použité pri simulácii a výsledné hodnoty charakteristiky u_i v pozorovanom časovom intervale, ako aj dĺžku tohto intervalu.

3.4 Robustnosť a ľahká modifikovateľnosť

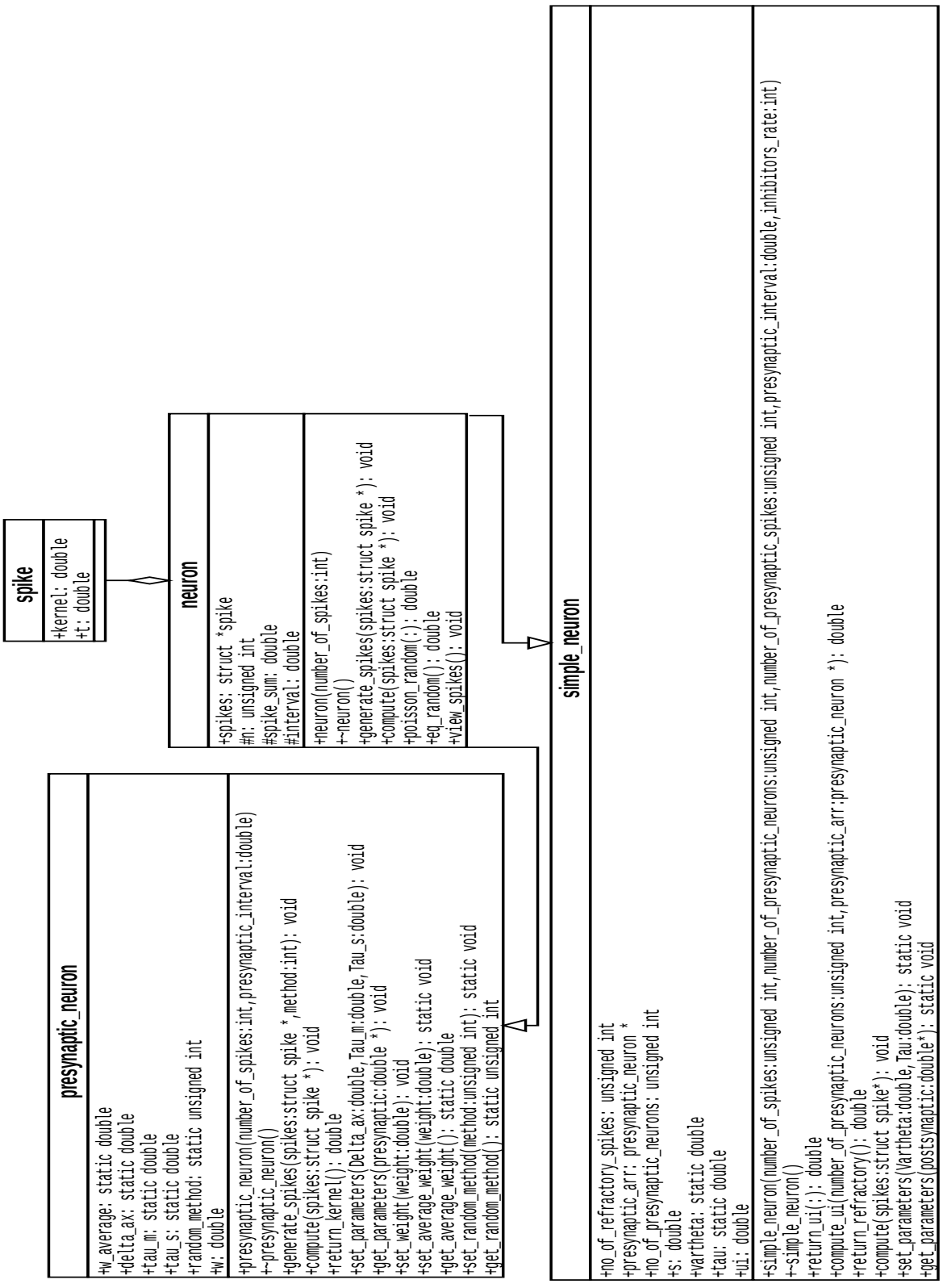
Celá aplikácia je distribuovaná do viacerých logicky oddelených celkov reprezentovaných konkrétnymi cpp súbormi, prípadne zmeny sa vždy týkaju len jedného celku, ktorý sa môže samostatne vyvíjať a neskôr dolinkovať k výslednému programu.

3.5 Nezávislosť simulačnej a vizuálnej časti

Simulačná a vizuálna časť programu sú od seba úplne nezávislé, čo umožňuje použiť simulačnú časť ľubovoľnej vizuálnej aplikácii v ľubovoľnom prostredí a vizualizovať len želané hodnoty. Vzhľadom k tomu, že na vizualizáciu charakteristík je potrebný výstup zo simulačnej časti, veľa času som venoval vyvíjaniu samotnej simulácie (čo značne prekračuje pôvodné zadanie ročníkového projektu), vizuálna aplikácia je len jednoduchý príklad grafického výstupu komplexného procesu simulácie.

3.6 UML návrh simulačných tried

Pomocou symbolov UML špecifikuje návrh tried (definované metódy, ich vstupné parametre, privátne a verejné atribúty, spôsob vnorenia (agregácia) a dedenia (inheritácia)).



Obrázok 1: Simulačné triedy

4 Návrh programu

4.1 Simulačná časť

Tvorí srdce celej aplikácie, zahrnutá je v súboroch compute.h, compute.cpp. Pozorovaný neurón musí byť reprezentovaný konkrétnou inštanciou triedy, objektom s patričnými počiatočnými vlastnosťami. Vlastnosti, alebo tiež parametre objektu sa inicializujú buď na začiatku, alebo už v existujúcom objekte a ovplyvňujú samotný beh simulácie.

4.1.1 Deklarácia štruktúry spajku

```
struct spike
{
  double kernel;
  double t;
};
```

Štruktúra spike predstavuje jeden konkrétny spajk v rámci jednoduchého neurónu, pozorovaný neurón je naviazaný na ľubovoľný počet presynaptických neurónov, každý z týchto neurónov môže mať voliteľný počet spajkov nastavených pri inicializácii objektu (volaní, konštruktora objektu neurón), rovnako pozorovanému neurónu pri inicializácii nastavujeme maximálne množstvo postsynaptických spajkov (vzniknutých v dôsledku odrazivosti).

Štruktúra obsahuje dva základné parametre.

Čas (t) vzhľadom na globálne meradlo času, kedy presynaptický neurón vyslal spajk, u postsynaptického neurónu je to čas vzniku refraktérneho spajku, ktorý vznikne v danom čase v dôsledkuprekročenia u_i nad hodnotu ϑ (prah excitácie).

Kernel (kernel) predstavuje celkový príspevok daného spajku k výslednej sume kernelu celého neurónu. V prípade postsynaptického neurónu je to príspevok k celkovej refraktárnej zložky pozorovaného neurónu.

4.1.2 Deklarácia tried

- **neuron**
  Reprezentuje základne vlastnosti jednoduchého neurónu, predstavuje rodičovskú triedu pre ostatné triedy a deklaruje všeobecne metódy používane pri simulácii neurónu. Metódy sú deklarované ako verejne prístupné, atribúty triedy sú deklarované ako "protected".

```
neuron(int number_of_spikes);
~neuron();
```

  Základný konštruktor triedy neuron má len jeden vstupný povinný parameter - množstvo spajkov, ktoré vyšle v priebehu simulácie. Deštruktor zabezpečí korektné odstránenie príslušného objektu.

```
void generate_spikes(struct spike *spikes);
```

  Metóda vygeneruje a nastaví náhodne časy spajkov pre daný neurón podľa zvolenej metódy generovania náhodných čísel (ako vstup sa udáva ukazovateľ na štruktúru spajkov). Množstvo spajkov bolo dané pri inicializácii neurónu, takže ako parameter sa neuvádza. Zatiaľ sú implementované dve metódy generovania náhodných čísel a to generovanie náhodných čísel s rovnomerným a a poissonovým rozložením.

```
double poisson_random();
double eq_random();
```

  Jednoduché matematické procedúry, ktoré vygenerujú náhodne číslo z intervalu $< 0, 1 >$ buď s rovnomerným (`double eq_random()` alebo poissonovým (`double poisson_random()` náhodným rozdelením.

```
void compute(struct spike *spikes);
```

  Metóda podľa definovaných matematických formúl (v prvej kapitole) vypočíta čiastkové hodnoty kernelov pre každý spajk v neuróne, všetky zosumuje a uloží do premennej `spike_sum`.

```
static void set_parameters(double, ...);
```

Nastavia sa voliteľné výpočtové parametre pre danú simuláciu závisle od typu neurónu na ktorý sa vzťahujú. Metóda je deklarovaná ako statická, čo znamená, že nastavené parametre majú globálnu platnosť pre všetky objekty danej triedy. Používa statické premenné ako výpočtové parametre, ktoré sú jedinečné pre všetky objekty danej triedy.

```
unsigned int n;
double spike_sum;
double interval;
```

Atribút `n` predstavuje množstvo vysielaných spajkov v rámci daného neurónu (reprezentuje vlastne "dĺžku spike-train").

`spike_sum` predstavuje výslednu sumu jednotlivých čiastkových kernelov (v prípade postsynaptického neurónu sú to refraktérne zložky) pre všetky spajky a jednu konkrétnu inštanciu neurónu. Počíta sa až pri zavolaní metódy `compute()`.

`interval` predstavuje časový rozsah, na ktorom sú generované a neskôr vysielané spajky pre daný neurón. Mimo tohto intervalu nikdy nedochádza k vysielaniu spajkov pre daný neurón.

- **presynaptic_neuron**
  Reprezentuje presynaptický vstup, ktorý je zviazaný s pozorovaným neurónom, neurón je naviazaný s viacerými presynaptickými vstupmi, ktoré rôznym spôsobom ovplyvňujú jeho správanie a výsledné generované excitačné/inhibičné napätie. Trieda je inheritovaná so základnej triedy `neuron`, prefažené a novovytvorené metódy oproti rodičovskej triede sú špecializované na výpočet presynaptických parametrov.

```
presynaptic_neuron(int number_of_spikes, double presynaptic_interval);
~presynaptic_neuron();
```

Konštruktor triedy má dva povinné vstupné parametre - množstvo spajkov, ktoré vyšle v priebehu simulácie a čas, počas ktorého sú generované a neskôr vysielané jednotlivé spajky.

```
static void set_parameters (double Delta_ax, double Tau_m, double Tau_s);
static void get_parameters (double *presynaptic);
static double delta_ax, tau_m, tau_s;
```

Statická metóda set_parameters nastaví presynaptické parametre δ^{ax} , τ_m a τ_s potrebné pre výpočet celkového kernelu $\epsilon(s)$ pre daný presynaptický neurón. Parametre majú globálnu platnosť pre všetky objekty danej triedy.

Metóda get_parameters uloží do poľa double presynaptic[3] súčasné nastavené hodnoty δ^{ax}, τ_m a τ_s. Ako parameter sa teda udáva len ukazovateľ na pole double.

```
void set_weight (double weight);
double w;
```

Metóda uloží pre daný presynaptický neurón požadovanú váhu do privátnej premennej triedy w. Váha predstavuje "silu spoja" presynaptického neurónu s pozorovaným neurónom. Môže nadobúdať záporne hodnoty pre inhibičné synapsy, alebo kladné pre excitačné, každý presynaptický neurón môže mať ľubovoľnú váhu.

```
static void set_average_weight (double weight);
static double get_average_weight ();
static double w_average;
```

Statická metóda set_average_weight nastaví priemernú hodnotu váh pre všetky presynaptické neuróny (premenná double w_average. Priemerná hodnota váh je statická premenná triedy, teda jedinečná pre všetky objekty presynaptických neurónov. Samotný výpočet presynaptického kernelu používa ale individuálnu váhu w odlišnú pre každý objekt neurónu. Závisi od konkrétnej implementácie výpočtovej metódy, či nastaví váhu daného neurónu podľa priemernej hodnoty double w_average.

Priemerná hodnota váh bola zavedená len v dôsledku zjednodušenia zadávaných vstupov (snaha o minimalizáciu množstva vstupných parametrov), kedy predpokladáme, že všetky presynaptické neuróny budú mať rovnakú váhu líšiacu sa len znamienkom (kladným pre excitačné, záporným pre inhibičné neuróny).

Metóda get_average_weight vráti priemernú hodnotu váh jedinečnú pre všetky presynaptické neuróny.

```
static void set_random_method (unsigned int);
static unsigned int get_random_method();
static unsigned int random_method;
```

Staticka metóda set_random_method nastaví náhodnú metódu pomocou ktorej sa budú generovať spajky jednotlivých presynaptických neurónov (po zavolaní metódy generate_spikes). K dispozícii sú zatiaľ dve náhodné metódy, EQUALLY (rovnomerné rozloženie) a POISSON (poissonove rozloženie)

```
void generate_spikes(struct spike *spikes, int method);
```

Preťažená metóda, ktorá vygeneruje n spajkov v presynaptickom neurónu použitím vybranej náhodnej metódy.

```
void compute(struct spike *spikes);
```

Vypočíta výsledny kernel pre celý presynaptický neurón zosumovaním spajkov, ktoré v danom časovom intervale už boli vyslané. Na konci simulácie uvažujeme vo výslednom kerneli čiastkové zložky kernelov všetkých vygenerovaných spajkov pre daný neurón.

- **simple_neuron**
  Najkomplexnejšia trieda reprezentujúca postsynaptický neurón, ktorý je zviazaný s definovaným množstvom presynaptických vstupov a súčasne má refraktérne vlastnosti.

```
simple_neuron(unsigned int number_of_spikes, unsigned int
number_of_presynaptic_neurons, unsigned int number_of_presynaptic_spikes,
double presynaptic_interval, int inhibitors_rate);
~simple_neuron();
```

Konštruktor objektu simple_neuron musí byť volaný s nasledujúcimi povinnými parametrami :

  - number_of_spikes
    Predstavuje maximálne množstvo refraktárnych spajkov, ktoré môžu vzniknúť počas celkovej simulácie v postsynaptickom neuróne. Skutočný počet vzniknutých refraktérnych spajkov môže byť nižší, táto hodnota predstavuje len hraničnú hodnotu potrebnú pre alokáciu pamäti spajkov. Vo vizuálnej aplikácii sa táto hodnota nastaví na podiel celkového času simulácie a dĺžky časového kroku (timing_step) simulácie. To zabezpečí, že pri daných hodnotách času simulácie a dĺžky časového kroku nebude nikdy prekročené takto definované hraničné množstvo refraktérnych spajkov.

- number_of_presynaptic_neurons
  Množstvo presynaptických neurónov zviazaných s daným pozorovaným postsynaptickým neurónom. Objekty takéhoto množstva presynaptických neurónov sú vytvorené v okamihu inicializácie objektu simple_neuron hneď v konštruktore.
- number_of_presynaptic_spikes
  Priemerná hodnota množstva spajkov v presynaptických neurónoch zviazaných s pozorovaným postsynaptickým neurónom. Pri vytvárani objektov presynaptických neurónov konštruktor presynaptic_neuron bude volaný s touto hodnotou. V našej konkrétnej simulácii bude mať teda každý presynaptický neurón rovnaký počet spajkov rôzne rozložených v danom časovom intervale.
- presynaptic_interval
  Dĺžka intervalu v ktorom budú generované a neskôr vysielané spajky presynaptických neurónov. Môže a nemusí sa zhodovať s celkovým časom simulácie. V prípade, že je menšia ako celkový čas simulácie (duration_of_simulation), po uplynutí času presynaptic_interval sa na celkovom u_i postsynaptického neurónu nebudú zúčastňovať už žiadne nové prírastky spajkov, takže priebeh napätia bude mať klesajúci tvar. V prípade, že presynaptic_interval bude totožný s celkovovým časom simulácie, spajky presynaptických neurónov budú vysielané až do konca simulácie.
- inhibitors_rate
  Percentuálne vyjadrenie množstva inhibičných presynaptických neurónov (so zápornou váhou) s celkového množstva presynaptických neurónov zviazaných s daným postsynaptickým neurónom. V prípade, že inhibitors_rate bude mať hodnotu 50%, priebeh u_i bude na dlhom časovom intervale zhruba vyvážený (čo sa týka kladných a záporných hodnôt), v prípade, že bude mať hodnotu 0%, nebudeme uvažovať o inhibičných presynaptických vstupoch, takže všetky vstupy budú mať excitačný charakter, u_i bude permanentne väčšie ako nula.

Deštruktor objektu simple_neuron zabezpečí korektné zrušenie naviazaných objektov reprezentujúcich presynaptické neuróny.

```
static void set_parameters (double Vartheta, double Tau);
static void get_parameters(double *postsynaptic);
```

```
static double vartheta,tau;
```

Statická metóda set_parameters nastaví postsynaptické parametre ϑ a τ potrebné pre výpočet refraktérnej zložky. Parametre majú globálnu platnosť pre všetky objekty danej triedy.

Metóda get_parameters uloží do poľa double presynaptic [2] súčasné nastavené hodnoty ϑ a τ. Ako parameter sa teda udáva len ukazovateľ na pole double.

```
void compute (struct spike *spikes) ;
double return_refractory();
```

Metóda compute vypočíta výslednu refraktérnu zložku pre postsynaptický neurón zosumovaním doteraz vzniknutých príspevkov refraktérných spajkov.

return_refractory predstavuje len wrapper, ktorý zavolá metódu compute s príslušným množstvom refraktérných spajkov pre daný postsynaptický neurón.

```
double compute_ui(unsigned int number_of_presynaptic_neurons, presynaptic_neur
on *presynaptic_arr);
double return_ui();
```

Metóda compute_ui predstavuje poslednú metódu simulácie na získanie výstupného u_i. Ako vstupné parametre sa uvádza množstvo presynaptických neurónov a ukazovateľ na ne. Kernely jednotlivých presynaptických kernelov sa spočítajú s príspevkami refraktérných spajkov, otestuje sa presiahnutie prahu excitácie (podmienka $ui > \vartheta$), v prípade prekročenia prahu sa vygeneruje nový refraktérny spajk, ktorý sa uloží na koniec poľa refraktérných spajkov postsynaptického neurónu. V ďalšom časovom kroku simulácie sa tento spajk svojím príspevkom už podiela pri výpočte nového u_i.

Metóda return_ui predstavuje wrapper k metóde compute_ui, ktorá sa zavolá s príslušným počtom presynaptických neurónov, ktoré sú naviazané na daný objekt postsynaptického neurónu.

```
unsigned int no_of_refractory_spikes;
presynaptic_neuron *presynaptic_arr;
unsigned int no_of_presynaptic_neurons;
double s;
double ui;
```

Atribúty triedy simple_neuron obsahujú zaradom množstvo refraktérných spajkov pre daný neurón, ukazovateľ na pole objektov presynaptických neurónov zviazených s neurónom, ich počet, premenná s je pomocná premenná vystupujúca v matematických formulách na výpočet refraktérnej zložky (predstavuje rozdiel aktuálneho času a času, kedy vznikol v neuróne daný refraktérny spajk). Premenná ui predstavuje výsledne vypočítané napätie na neuróne u_i.

4.2 Vizuálna časť

Vizualizuje výsledky simulácie, je závislá na použitej grafickej knižnici. Vstupne parametre sú zadávané pomocou špeciálne navrhnutých okien (widgety), hodnoty napätia sú nanášane na graf v posuvnom okne.

4.2.1 Návrh widgetov a ich obsluha

Všetky typy okien (widgety) boli navrhnutá v špeciálnom programe QT Designer štandartne dodávanom ku knižnici QT[7], ktorý umožňuje rýchly a flexibilný návrh widgetov. Navrhované widgety sú formátu XML, čo umožňuje ich prenositeľnosť do ľubovoľného jazyka, utilitka uic dodávaná s knižnicou QT umožňuje preklad XML formulára do c++ zdrojového a hlavičkového súboru obsahujúci základné deklarácie a definície tried použitých grafických komponentov. Používateľské widgety sa vytvárajú inheritovaním týchto tried a preťažením používaných metód. Komunikácia medzi grafickými komponentami je zabezpečená pomocou mechanizmu signal-slot (natívny spôsob komunikácie QT komponentov). QT Designerom boli vygenerované súbory main_simulation.cpp/main_simulation.h, presynaptic.cpp/presynaptic.h, postsynaptic.cpp/postsynaptic.h, simulation.cpp/simulation.h obsahujúce kompletnú základnú deklaráciu a definíciu použitých widgetov.

4.2.2 Kreslenie grafu

Hlavná časť vizualizácie (kreslenie grafu) je uložená v súbore impulsemain.h, impulsemain.cpp, rovnako je tu vytvorené prepojenie (interface) so simulačnou časťou.

Po ukončení behu simulácie a vygenerovaní všetkých hodnôt napätia v danom časovom intervale, sa analyzujú hraničné body simulácie na osi X ako aj Y. Získa sa celkový rozmer výsledného grafu, ktorý sa preškáluju do posuvného okna a vykreslí. Škálovanie zabezpečí zobrazenie všetkých výstupných hodnôt počas celého simulačného času. Po presune hlavného okna aplikácie, resp. prekrytí s inou aplikáciou, sa okno aplikácie korektne prekreslí.

Vyznačené osi grafu sú škálované v rovnakej mierke. Výsledné body sú spájane bikvadratickou bezériovou krivkou a vykreslované hrubou červernou farbou, zelenou farbou je vykreslená hranica označujúca prah excitácie ϑ.

[7] http://www.trolltech.com/products/qt/qtinfo.html

5 Testovanie a overovanie programu

Testovanie programu a výsledkov priebehalo na troch nezávislých počítačoch s rôznou distribúciou OS Linux a verziou knižnice QT. Na každom počítači prebehlo základné testovanie bez problémov.

5.1 Kompilačné módy

Kompilátor g++ poskytuje viacero možností kompilácie výslednej aplikácie. Pre potreby ladenia pomocou GNU prostriedku gdb bolo použité *DEBUG* nastavenie -pipe -Wall -W -ggdb -DDEBUG . Na vytvorenie koncovej verzie som použil *RELEASE* nastavenie -pipe -Wall -W -O3 -DNODEBUG . *RELEASE* nastavenie v sebe zahrňuje aj nízkoúrovňovú inštrukčnú optimalizáciu.

5.2 Použité ladiace prostriedky

1. gdb

   GNU debugger 5.0.11 som použil na hľadanie možných chýb a debugovanie programu.

2. gprof

   GNU gprof 2.10.91 mi pomohol odhaliť časovo-náročné funkcie simulácie a zlepšiť optimalizáciu.

3. gnuplot

   Prostriedok GNU plot 3.7.1 som použil na vizualizáciu medzivýsledkov simulácie v čase, keď som ešte nemal implementovanú vizuálnu časť.

6 Používateľská príručka

6.1 Inštalácia aplikácie

Na korektnú inštaláciu aplikácie musia byť splnené nasledujúce systémové požiadavky:

- Operačný systém
  Program bol úspešné otestovaný na OS Linux, mal by byť prenositeľný na ľubovoľnú platformu podporujúcu kompilátor jazyka C++ a grafickú knižnicu QT
- Knižnica QT
  Program využíva voľne prístupnú verziu (pod X11) knižnice QT, testovaný bol na verzii 2.2 až poslednú 2.3. Na úspešnú kompiláciu je nutné mať nainštalované hlavičkové súbory QT, pomocné prostriedky QT (uic, moc) a správne nastavenú "environment" premennú QTDIR[8].
- X11 Window System
  Tento oknový systém je potrebný len na UNIX kompatibilných platformách.
- C++ kompilátor a linker
  Program bol úspešne kompilovaný pomocou g++ z balíka GNU GCC, testované verzie g++ 2.95 až 2.96

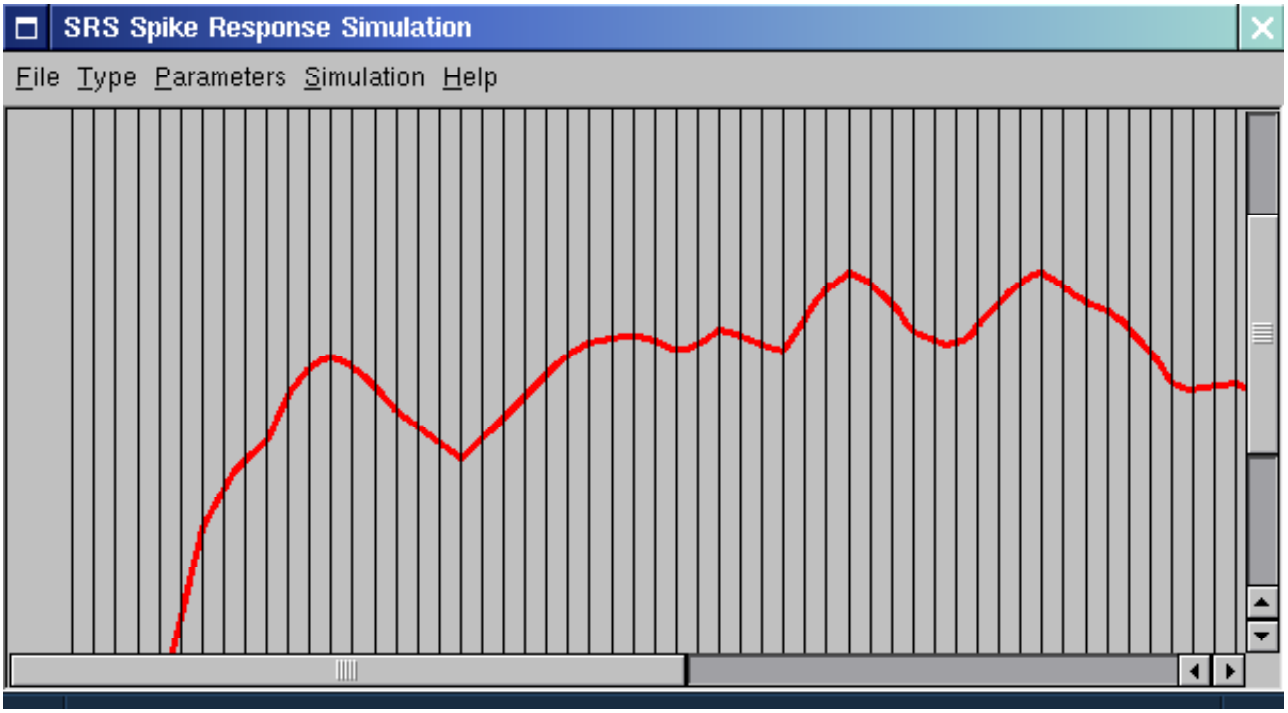
Aplikácia je distribuovaná vo forme zdrojových kódov, na úspešnú kompiláciu stačí napísať príkaz Make. Výsledný názov spustiteľného súboru je impulse. Štandartne sa generuje optimalizovaný binárny súbor bez debug informácie.

6.2 Používateľské rozhranie

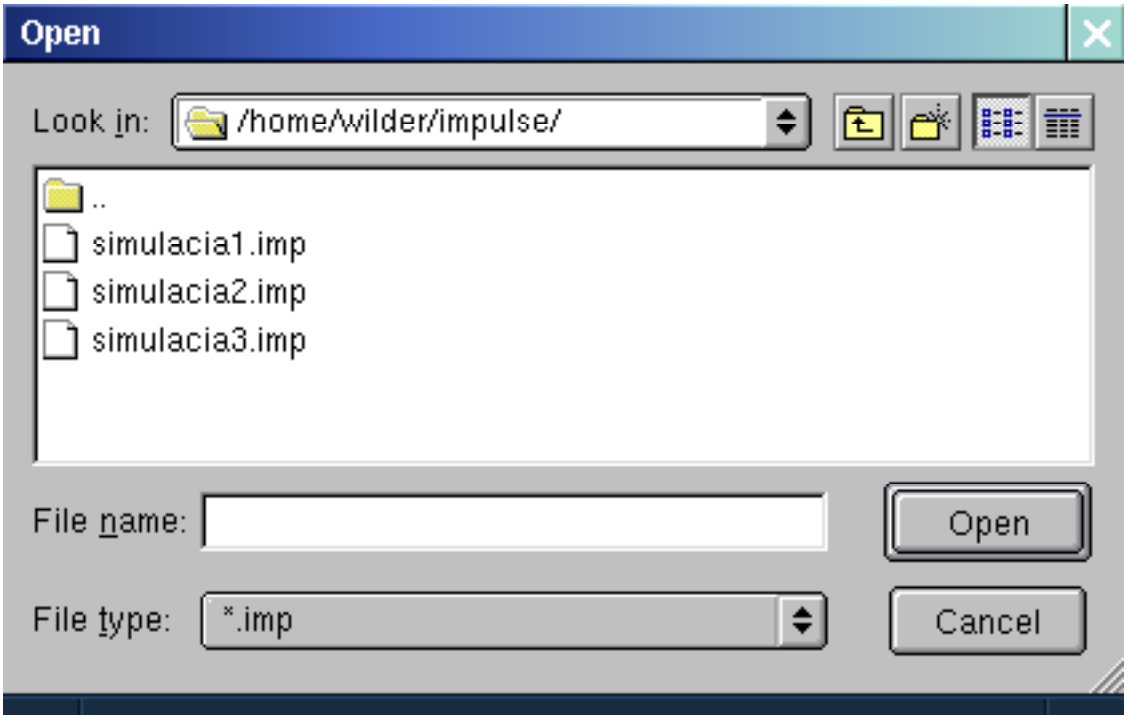
Po spustení binárného súboru ./impulse máme k dispozícii hlavné menu. Z uvedeného menu máme priamy prístup do nasledujúcich položiek:

- File
  - Open state
    Otvorí sa nám menu na otváranie/ukladanie súboru. Môžeme načítať zo súboru stav predtým uloženej simulácie. V súbore sú uložené všetky presynaptické, postsynaptické a simulačné parametre, vrátane hodnôt u_i na simulovanom časovom intervale, ako aj body prahu excitácie (okamihy tresholdov).

[8] bližšie info na http://www.trolltech.com/developer/faq/install.html



Obrázok 2: Hlavné menu



Obrázok 3: Menu na otváranie/ukladanie súboru

  - Save current state / Save as

    Uložíme súčasnú konfiguráciu všetkých parametrov, v prípade, že máme už vizualizovanú simuláciu, uložíme aj hodnoty napätia na simulovanom časovom intervale.

  - Exit

    Ukončíme prácu s programom.

- Type

  - Simple neuron

    Štandartné nastavenie typu simulácie.

  - Network of neurons

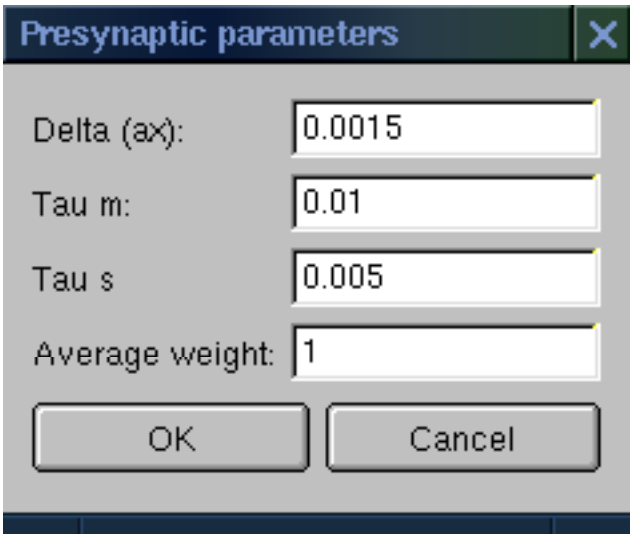
    Zatiaľ neimplemntovaný typ simulácie pre budúce využitie.

- Parameters

  Parametre po spustení aplikácie obsahujú štandartné hodnoty (experimentálne odporúčené). Zmeniť ich môžeme prepísaním v príslušnom menu alebo načítaním nového stavu simulácie zo súboru.

  - Presynaptic parameters

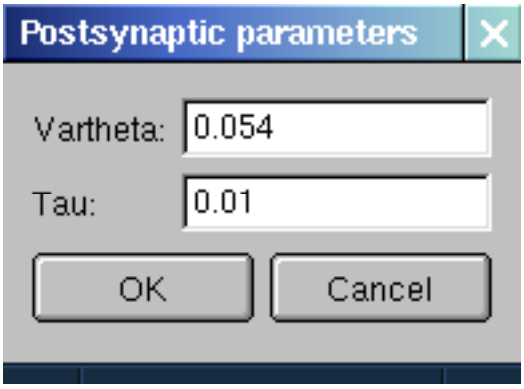
    Presynaptické parametre sa týkajú simulačných výpočtov presynaptických neurónov, konštánt δ^{ax}, τ_m, τ_s a priemernej váhy presynaptického vstupu zviazeného s pozorovaným neurónom.



Obrázok 4: Presynaptické parametre

- Postsynaptic parameters

  Postsynaptické parametre predstavujú konštanty ϑ a τ, potrebné pre výpočet refraktérnych spajkov. ϑ predstavuje prah excitácie po ktorom prekročení sa generuje v postsynaptickom neuróne refraktérny spajk.



Obrázok 5: Postsynaptické parametre

- Simulation parameters

  Pre simulovaný impulzný neurón nastavujeme hodnoty množstva presynaptických vstupov, priemerné množstvo spajkov vyslaných v presynaptickom časovom intervale, dĺžku presynaptického časového intervalu, percentuálne zastúpenie inhibičných synáps, celkový čas simulácie a čas krokovania celej simulácie. Rovnako môžeme nastaviť náhodnú metódu, ktorá sa použije pri generovaní časov vyslaní presynaptických spajkov v jednotlivých impulzných neurónoch.

- Simulation

  - Start

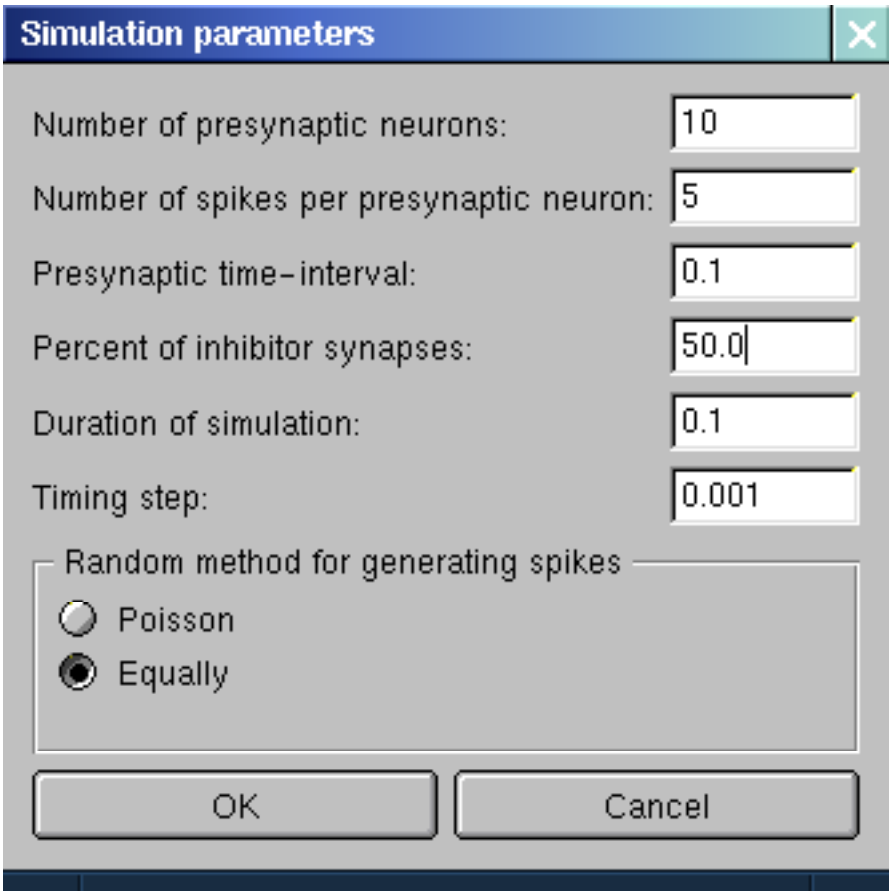
    Spustenie simulácie podľa prednastavených presynaptických, postsynaptických a simulačných parametrov. Použitie tejto voľby spustí kompletnú simuláciu a vykreslí hodnoty u_i na danom časovom intervale.

  - Stop

    Zastavenie simulácie, v súčasnej implementácii sa simulácia zastaví po dosiahnutí celkového času simulácie.

- Help

  - Help



Obrázok 6: Simulačné parametre

Spustenie helpu.

- About
  Informácie o autorovi.

6.3 Obsah priloženého média

Na priloženom dátovom médiu k dokumentu sa nachádza :

- archív impulse-1.0.tar.gz obsahujúci všetky súbory potrebné ku skompilovaniu a inštalácii aplikácie **Spike Response Simulation**.
- súbory impulse.ps a impulse.pdf predstavujúce elektronickú formu tohto dokumentu

7 Zhodnotenie

Impulzné neuróny predstavujú jeden z najmodernejších a najpresnejších popisov biologického neurónu. Spoznanie charakteristík správania sa impulzného neurónu predstavuje krok k poznaniu, determinovaniu možných reakcií biologických neurónu a návrhu moderných neuročipov.

Aplikácia vhodne poslúži pri vyúčbe správania sa impulzných neurónov ako aj simulovaní neurónov konkretných konfigurácií.

Dôraz bol kladený na simulačnú stránku aplikácie a premyslený návrh simulačných tried, ktoré predstavujú použitelný základ na vybudovanie simulačných tried kompletnej siete impulzných neurónov.

Vizualizácia je veľmi jednoduchá a znázorňuje len jednu konkrétnu simulovanú charakteristiku. Vzhľadom na programovú nezávislosť aplikácie, nie je problém napísať funkčné vizuálne "front-end" rozhranie na ľubovoľnej platforme použitím ľubovoľného c++ kompilátora a vizualizovať len zvolené charakteristiky.

Návrh a implementácia projektu mi značne pomohli pri pochopení princípu prenosu informácie pomocou impulzného kódovania, naučil som sa používať veľmi robustné grafické rozhranie QT a abstrahovať vlastnosti a správanie matematického modelu impulzného neurónu do objektov.

8 Zoznam použitej literatúry

(1) Bishop M.Christopher, Maas Wolfgang : Pulsed Neural Networks, Massachusetts Institute of Technology, 1999

(2) Maass : Lower Bounds for the computational power of spiking neurons, Neural Computation, 1996

(3) Beňušková Ľ.: Kognitívne vedy - Neurovedy I - Neurón a mozog

(4) http://www.trolltech.com/qt/, online dokumentácia ku knižnici QT, Trolltech corp., Nórsko

(5) Šešera, Ľ., Mičovský A.: Objektovo-orientovaná tvorba systémov a jazyk C++, Perfect, Bratislava 1994