Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky

Paralelné výpočty pre spracovanie a vizualizáciu stavového priestoru nelineárnych obvodov

Diplomová práca

2017 **Zsolt Rácz**

Technická univerzita v Košiciach

Fakulta elektrotechniky a informatiky

Paralelné výpočty pre spracovanie a vizualizáciu stavového priestoru nelineárnych obvodov

Diplomová práca

Študijný program:	Informatika
Študijný odbor:	Informatika
Školiace pracovisko:	Katedra počítačov a informatiky (KPI)
Školiteľ:	Doc. Ing. Branislav Sobota, PhD
Konzultant:	Doc. Ing. Milan Guzan, PhD

Košice 2017 **Zsolt Rácz**

Abstrakt v SJ

Táto práca sa venuje problematike simulácie nelineárnych obvodov, vykazujúce chaotické správanie. Na začiatku práce sú predstavené chaos, ako jav, Chuaov obvod, hraničná plocha, a matematické podklady potrebné pre jej simuláciu. V rámci práce bol vyvinutý softvér pre simuláciu, 2D a 3D vizualizáciu stavu Chuaovho obvodu a výpočtu jeho hraničnej plochy. V práci je predstavených niekoľko spôsobov urýchlenia výpočtu, ako paralelizácia, zmeny parametrov simulácie alebo optimalizácie prekladača, ktoré boli aj implemetované do vyvinutého riešenia. Dosiahnuté zrýchlenie je dokladované experimentmi, ktorých výsledky sú zahrnuté v práci. Zároveň práca prezentuje grafické výstupy generované vytvorenou aplikáciou.

Kľúčové slová

Atraktor; Chaos; Chuaov obvod; Hraničná plocha; Intel TBB; Optimalizácia; Paralelizácia.

Abstrakt v AJ

The scope of this thesis are of non-linear circuits producing chaotic behaviour. The first part of the thesis presents chaotic phenomena in general, Chua's circuit, boundary surface and mathematical foundations required for its simulation. A software for simulation, 2D and 3D visualization of Chua's circuit's state and calculation of its boundary space has beed developed within the thesis. Methods of incrementing calculation speed, e.g. parallelization, calculation parameter variations and compiler optimizations are presented aswell, witch have been also implemented in the developed software. The performance improvements have been confirmed by experiments, whose results are also presented in this work. Finally it includes graphical outputs of calculations accomplished using the developed application.

Kľúčové slová v AJ

Attractor; Chaos; Chua's circuit; Boundary surface; Intel TBB; Optimization; Parallelization.

TECHNICKÁ UNIVERZITA V KOŠICIACH

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Katedra počítačov a informatiky

ZADANIE DIPLOMOVEJ PRÁCE

Študijný odbor: **Informatika**
Študijný program: **Informatika**

Názov práce:

Paralelné výpočty pre spracovanie a vizualizáciu stavového priestoru nelineárnych obvodov

Parallel computing for processing and visualization of state space of nonlinear circuits

Študent:	**Bc. Zsolt Rácz**
Školiteľ:	**doc. Ing. Branislav Sobota, PhD.**
Školiace pracovisko:	**Katedra počítačov a informatiky**
Konzultant práce:	**Ing. Milan Guzan, PhD.**
Pracovisko konzultanta:	**Katedra teoretickej a priemyselnej elektrotechniky**

Pokyny na vypracovanie diplomovej práce:

1. Oboznámiť sa s Chuaovým obvodom, jeho opisom systémom diferenciálnych rovníc (SDR) a ich riešením
2. Vytvoriť program na výpočet trajektórií a rezov hraničnými plochami (HP)
3. Zistiť možnosti paralelizácie výpočtov na viacjadrových CPU a následne ich aj realizovať
4. Zistiť možnosti zrýchlenia výpočtu HP a zrýchlenia dokladovať tabelárne
5. Modifikovať SDR za účelom výpočtu iných chaotických atraktorov a HP
6. Vypracovať dokumentáciu na základe pokynov vedúceho práce

Jazyk, v ktorom sa práca vypracuje: slovenský
Termín pre odovzdanie práce: 28.04.2017
Dátum zadania diplomovej práce: 31.10.2016

.....
doc. Ing. Jaroslav Porubän, PhD.
vedúci garantujúceho pracoviska

TECHNICKÁ UNIVERZITA V KOŠICIACH FAKULTA ELEKTROTECHNIKY A INFORMATIKY -13-

.....
prof. Ing. Liberios Vokorokos, PhD.
dekan fakulty

Čestné vyhlásenie

Vyhlasujem, že som diplomovú prácu vypracoval samostatne s použitím uvedenej odbornej literatúry.

Košice 28. 4. 2017 .....

Vlastnoručný podpis

Poďakovanie

Ďakujem Doc. Ing. Milanovi Guzanovi, PhD. za cenné rady, návrhy a množstvo času, ktoré mi venoval počas vypracovania diplomovej práce a Doc. Ing. Branislavovi Sobotovi, PhD. za jeho odborné vedenie a pripomienky.

Obsah

Zoznam obrázkov

Zoznam tabuliek

Zoznam symbolov a skratiek

2D Dvojrozmerný

3D Trojrozmerný

AES Štandard pokročilého šifrovania (Advanced Encryption Standard)

API Rozhranie pre programovanie aplikácií (Application programming interface)

AVX Advanced Vector Extensions

CHA Chaotický Attraktor

CPU Centrálna procesorová jednotka (Central processing unit)

CSV Čiarkou oddelené hodnoty (Comma separated values)

CUDA Compute Unified Device Architecture

DE Bezrozmerné rovnice (Dimensionless Equations)

GCC GNU Compiler Collection

GPL GNU General Public License

GPU Grafický procesor (graphics processing unit)

GUI Grafické používateľské rozhranie (Graphical User Interface)

HP Hraničná Plocha

ICC Intel C++ Compiler

LGPL GNU Lesser General Public License

OpenCL Open Computing Language

PLY Polygon File Format

RP Región príťažlivosti

SIMD Single Instruction Multiple Data

SLC Stabilný limitný cyklus

SQL Štruktúrovaný dopytovací jazyk (Structured Query Language)

SSE Streaming SIMD Extensions

SVG Scalable Vector Graphics

VA Volt-Ampérová

XML Rozšíriteľný značkovací jazyk (Extensible Markup Language)

ZB Zastupujúci bod

ZDR Záporný Diferenciálny Odpor

ZP Začiatočná Podmienka

Atraktor je stav, do ktorého dynamický systém v čase smeruje.

Bifurkácia je jav v dynamických systémoch, kedy minimálna zmena parametrov zapríčiní veľké kvalitatívne alebo topologické zmeny v jeho správaní.

Framework alebo *aplikačný rámec* je univerzálny softvér, ktorý poskytuje pomocné nástroje a knižnice pre vývoj softvéru.

GNU je projekt zameraný na slobodný software, inšpirovaný operačnými systémami unixového typu.

Granularita paralelného výpočtu predstavuje množstvo vykonaného výpočtu v jednej úlohe.

Limitný cyklus je uzavretá trajektória v stavovom priestore, ktorá je izolovaná od iných uzavretých trajektórií.

Linux je voľne širiteľný operačný systém, ktorý je vyvíjaný komunitou.

Mračno bodov je veľký súbor bodov, pričom každý bod má zadefinovanú svoju polohu v priestore.

Qt je multiplatformový aplikačný rámec pre vytváranie aplikácií s grafickým používatelským rozhraním.

Task stealing alebo *work stealing* je spôsob zefektívnenia paralelizácie, pri ktorom dochádza k dynamickom prerozdelení úloh medzi vláknami.

Toolkit je sada softvérových nástrojov.

Widget pochádza z anglického slova, ktoré znamená „vecička“. V kontexte grafických používateľských rozhraní je chápaný ako komponent používateľského rozhrania.

x86-64 je 64 bitová verza inštrukčnej sady x86.

Úvod

Výskum chaotického správania deterministických dynamických systémoch sa v posledných desaťročiach stalo jednou z dôležitých vedeckých oblastí. Chuaov obvod vďaka jej jednoduchosti je výborným nástrojom pre výskum chaotických javov. Jeho správanie je možné skúmať aj pomocou počítačovej simulácie. Ide však o pomerne náročný výpočet, ktorý pred niekoľko desiatkami rokov bol ešte problematický.

Hlavným motivátorom práce bola kniha *„A Gallery of Chua Attractors“* [5], ktorá uvádza 195 chaotických atraktorov fyzikálneho, a ďalších 251 atraktorov zovšeobecneného modelu Chuaovho obvodu. Všetky predstavené obvody však majú viac atraktorov. V ktorých bodoch stavového priestoru sa ale dochádza k bifurkacii trajektórií? Ak sa budeme zaujímať o utajenie požadovaného signálu, tak amplitúda utajovaného signálu nesmie presiahnuť HP oddeľujúcu nepredvídateľný chaotický signál od periodického – ľahko predvídateľného. Aký veľký môže byť vstupný signál? Pri viachodnotovej pamäti ako zabezpečiť jeho ovládanie? Na všetky otázky nám dá exaktnú odpoveď hraničná plocha.

Výpočet hraničnej plochy je založený na opakovanom výpočte trajektórií obvodu. Ide o výpočtovo náročnú operáciu. Pre takýto výpočet v bežnom rozlíšení $440 \times 440 \times 440$ je nutné vypočítať viac ako 85 miliónov trajektórií. Nie je teda prekvapujúce, že takýto výpočet trvá týždne aj na novších počítačoch.

V práci skúmame možnosti efektívnej realizácie výpočtov hraničných plôch, okrem iných aj s využitím paralelizácie. Pri výskume chaotických systémov je dôležitá aj názorná vizualizácia stavového priestoru. Preto dôraz kladieme aj na kvalitu grafických zobrazení.

Formulácia úlohy

Hlavnou úlohou tejto diplomovej práce je návrh a implementácia softvérového riešenia, ktoré je použiteľné pre výskum Chuaovho obvodu. Pred vývojom bolo potrebné sa dôkladne oboznámiť s Chuaovým obvodom, jeho matematickým modelom opísaným systémom diferenciálnych rovníc a ich riešením. Na základe týchto poznatkov bolo potrebné vyvinúť riešenie, ktoré umožní výpočet trajektórií a rezov hraničnými plochami (HP).

Keďže výpočty rezov HP sú spravidla pomerne zdĺhavé, bolo zadané použiť paralelizáciu výpočtov, ako spôsob urýchlenia výpočtov, ale aj výskum ďalších možností optimalizácie času na výpočet rezov HP. Úlohou bolo tiež poskytnúť prehľad miery zrýchlenia pri rôznych spôsoboch optimalizácie výpočtu na základe experimentov.

Ďalšou úlohou bolo navrhnutie úprav vytvoreného riešenia tak, aby počítal s rôznymi modifikáciami volt-ampérovej charakteristiky Chuaovej diódy, ale aj s bezrozmerným systémom rovníc Chuaovho obvodu.

Zároveň k vytvorenému softvérovému riešeniu bolo potrebné vypracovať dokumentáciu vo forme používateľskej a systémovej príručky.

1 Teória chaosu

V bežnom jazyku pojem chaos znamená zmätok, neporiadok alebo nejasnosť. V kontexte modernej vedy avšak sa pod chaosom rozumie nepravidelný a neperiodický vývoj dynamických systémov. Správanie chaotických systémov sa môže zdať ako úplne náhodné, avšak je v skutočnosti deterministické. Chaos môžeme definovať aj ako zložité dynamické správanie jednoduchých systémov [4].

Jedným z prvých priekopníkov teorie chaosu bol Edward Lorenz, meteorológ na MIT. V 60-ich rokoch, počas počítačovej simulácii modelov počasia zistil, že aj malá zmena v začiatočných podmienkach môže časom pôsobiť veľké rozdiely vo výsledkoch. Tento jav nazval efektom motýlich krídiel na jeho prednáške v roku 1972, kedy položil zaujímavú otázku: *„Môže byť tornádo v Texase pôsobené trepotom motýlích krídiel v Brazílii?“*. Otázka slúžila skôr pre vzbudenie pozornosti, jednoznačnú odpoveď na ňu nepoznáme, avšak podľa jeho pozorovania, aj malé vplyvy, ako tlkot krídla motýľa, môžu dlhodobo vzrásť a pôsobiť výraznú zmenu. Jeho výskum poukázal na to, že ani veľmi detailné modely atmosféry nemôžu dlhodobo poskytovať správne predpovede. Z Lorenzovho objavu sa postupne vyvinula nová vedecká disciplína – teória chaosu [23], [38].

Matematický zápis dynamických systémov sa skladá zo sústavy diferenciálnych rovníc. Po čase sa zvyčajne dostanú do ustáleného periodického pohybu, ale za určitých okolností môžu vykazovať chaotické správanie. Základnou podmienkou vzniku chaotických javov v dynamickom systéme je, aby rovnica systému obsahovala nelinearitu.

Riešenie sústav nelineárnych diferenciálnych rovníc je zložité a zvyčajne neexistuje analytický postup pre zistenie výsledného vzorca. Ostáva možnosť simulácie numerickými metódami, tie sa však skladajú z veľkého objemu výpočtov a ich výpočet je spravidla veľmi zdĺhavý. Preto sa chaos dostal do popredia až koncom 20-ho

storočia, kedy výkon počítačov začal prudko rásť a bolo ich možné využiť aj na dovtedy nevypočítateľné problémy [38].

Mnoho nelineárnych systémov za žiadnych okolností nevykazuje chaotický pohyb, ku chaosu sa dochádza iba pri určitých parametroch systému a aj to len pri určitých začiatočných podmienkach [4]. Určenie týchto parametrov a začiatočných podmienok nie je vôbec jednoznačné, pri zistení týchto údajov môžu pomôcť Ljapunove exponenty, alebo sa použije metóda pokus-omyl. V tabuľke 1 – 1 sú porovnané vlastnosti pravidelných a chaotických pohybov.

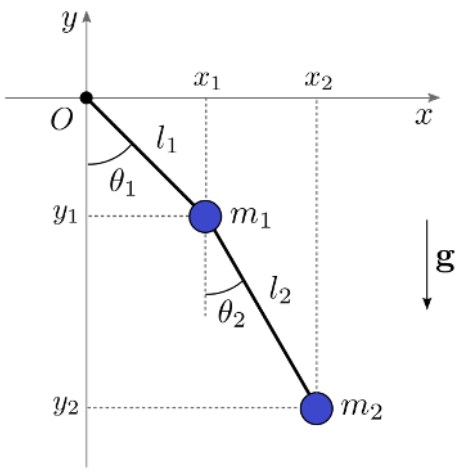
Tabuľka 1 – 1: Charakteristika pravidelného a chaotického pohybu [4]

Pravidelný pohyb	**Chaotický pohyb**
periodický	neperiodický
predvídateľný	nepredvídateľný
jednoduchý geometrický tvar	zložitý geometrický tvar

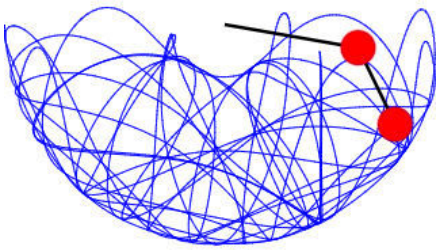
1.1 Dvojité kyvadlo ako príklad chaotického systému

Ľahko predstaviteľným príkladom chaosu je pohyb dvojitého kyvadla, ktoré je znázornené na Obr. 1 – 1a. Skladá sa z dvoch matematických kyvadiel, pričom druhé kyvadlo je uchytené na závaží prvého kyvadla. Hodnoty l_1 a l_2 predstavujú dĺžky vlákien a m_1 a m_2 hmotnosť závaží (hmotných bodov). Konfiguráciu systému udávajú uhly θ_1 a θ_2 spolu s príslušnými uhlovými rýchlosťami ω_1 a ω_2. Aj tento jednoduchý systém môže generovať chaos pri istých okolnostiach, napr. pri začiatočných hodnotách $\theta_1, \theta_2 = 90°$ a $\omega_1, \omega_2 = 0$. Chaotickú trajektóriu spodného závažia pri uvedených začiatočných hodnotách je možné vidieť na Obr. 1 – 1b.

Ďalšími príkladmi na chaotické systémy sú nelineárne elektrické obvody, dynamika srdca, tok vody z vodovodného kohútika, ekonomika štátu, výkyvy finančných



(a) Model dvojitého kyvadla [2]



(b) Trajektória spodného závažia [34]

Obr. 1 – 1: Dvojité matematické kyvadlo

trhov, vývoj počasia, klímy alebo ľudských spoločenstiev [15], [21], [18]. Ďalšie kapitoly tejto práce sa sústreďujú v prvom rade na nelineárne elektrické obvody – a to konkrétne na Chuaov obvod.

1.2 Chaos v elektrických obvodoch

Chaotické javy sa môžu vyskytnúť aj v nelineárnych elektrických obvodoch. Podmienkou je, aby obvod obsahoval aspoň jeden lokálne aktívny nelineárny obvodový prvok. Pod prvkom rozumieme taký prvok elektrického obvodu (rezistor, cievka, kondenzátor), u ktorého jeho určujúca veličina (odpor R, indukčnosť L, kapacita C a pod.) závisí od hodnoty obvodových veličín – napätí, resp. prúdov. Lokálna aktivita je dosiahnutá pripojením elektrického zdroja k prvku, graf VA charakteristiky prvku teda má byť na aspoň jednom intervale klesajúca. Takýmto prvkom je napr. Chuaova dióda, ktorá sa vyskytuje v Chuaovom obvode.

1.3 Praktické využitie teórie chaosu

Skúmanie chaotických systémov je dôležité pre lepšie pochopenie chaosu ako javu. Využitie teórie chaosu však sa už našlo v rôznych odvetviach. V tejto podkapitole uvedieme niekoľko príkladov.

Rehabilitačná topánka – pri chôdzi človeka môžeme spozorovať chaotické javy. Niektoré pohyby môžu viesť k strateniu rovnováhy a páde človeka. Rehabilitačná topánka slúži na zlepšenie motorických schopností seniorov a prípravu na nepredvídateľné udalosti pri chôdzi, znížením tak rizika spadnutia a úrazu. Taktiež je použiteľná aj pre rehabilitáciu po úrazoch. V rehabilitačnej topánke sú motory riadené chaotickým signálom, vďaka čomu sú zmeny polohy topánky nepredvídateľné [32].

Generovanie náhodných čísel – generátor náhodných čísel s nedostatočnou náhodnosťou, resp. s predvídateľnými výsledkami môže v kryptografii znamenať vážne bezpečnostné riziko. Kvalitný generátor náhodných čísel je preto veľmi dôležitý, obzvlášť pri generovaní šifrovacích kľúčov. Ako faktor náhodnosti môže byť použitý elektrický signál pochádzajúci z elektrického obvodu generujúceho chaos (napr. Chuaov obvod). Aj keď tento signál nie je doslovne náhodný, je však nepravidelný, neperiodický a vďaka týmto vlastnostiam aj nepredvídateľný. Obvody vykazujúce chaotické správanie môžu byť teda ideálnym zdrojom pseudo-náhodných čísel pre využitie v kryptografii, čo dokazujú aj experimenty v [25].

Šifrovanie signálu – chaos v kryptografii je možné využiť aj priamo pre zabezpečenie komunikácie. Pomocou dvoch synchronizovaných Chuaových obvodov generujúcich chaos je možné s vysokou presnosťou šifrovať a dešifrovať elektronický signál [39].

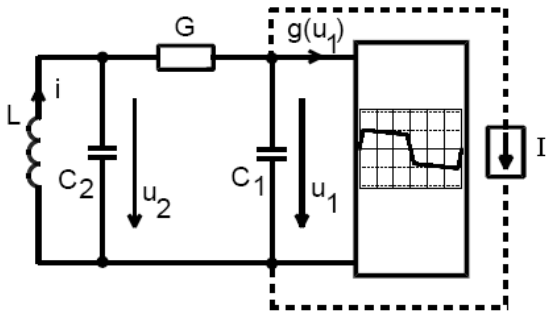
2 Chuaov obvod

Skúmanie správania konkrétneho chaotického systému za rôznych okolností je jedným spôsobom pre lepšie pochopenie chaosu, ako javu. Elektrické obvody sú relatívne ľahko skúmateľné z matematického, numerického, ale aj z experimentálneho hľadiska [24].

Chuaov obvod je jednoduchý obvod tretieho rádu a je významný tým, že bol prvým fyzikálnym obvodom, ktorý dokázateľne vykazoval neperiodické (chaotické) správanie [15]. Obvod bol predstavený v roku 1983. Predstavil ho jeho objaviteľ, prof. Leon O. Chua z univerzity Berkeley. Odvtedy sa obvod stal jedným z najpoprednejších a najviac skúmaných nelineárnych elektrických obovodov [27]. Obvod sa prvom rade využíva na skúmanie chaotických javov, ale našlo sa jeho využitie aj v oblasti šifrovania signálov [39].

2.1 Schéma obvodu

Schéma Chuaovho obvodu (Obr. 2 – 1) je pomerne jednoduchá, skladá sa iba zo štyroch lineárnych prvkov (2 kondenzátory, 1 rezistor, 1 cievka) a z jedného nelineárneho prvku tzv. Chuaovej diódy [15], [27].



Obr. 2 – 1: Schéma Chuaovho obvodu

Ako je možné vidieť aj na Obr. 2 – 1, Chuaov obvod sa skladá z jednoduchých, bežne dostupných elektrotechnických súčiastok. Chuaovu diódu je možné zostrojiť použitím operačných zosilňovačov, diód alebo tranzistorov. Realizácia obvodu je preto pomerne jednoduchá a lacná [27].

2.2 Matematický zápis

Našim cieľom je získať presné hodnoty pri rôznych a presne kontrolovateľných konfiguráciách, preto namiesto experimentálneho prístupu zvažujeme iba matematické a numerické prístupy. Vychádzame z matematickej definície obvodu, ktorú predstavujú nasledujúce diferenciálne rovnice:

$$\begin{aligned} C_1\frac{du_1}{dt} &= \frac{1}{R}(u_2 - u_1) - g(u_1) \\ C_2\frac{du_2}{dt} &= \frac{1}{R}(u_1 - u_2) + i \\ L\frac{di}{dt} &= -u_2 - \rho i \end{aligned} \tag{2.1}$$

kde: $g(u_1)$ je funkcia reprezentujúca VA charakteristiku Chuaovej diódy,
u_1 je napätie na kondenzátore C_1,
u_2 u2 je napätie na kondenzátore C_2,
i je prúd na cievke L.

Pri výskume chaotických javov v Chuaovom obvode sa často používa zovšeobecnený tvar diferenciálnych rovníc (2.1). Parametre obvodu reprezentujúce fyzikálne veličiny sú pri tomto zápise nahradené bezrozmernými hodnotami a namiesto u_1, u_2 a i sú použité x, y a z. Zovšeobecnený tvar rovníc Chuaovho obvodu je vyjadrený nasledovne:

$$\begin{aligned} \frac{dx}{d\tau} &= \kappa\alpha(y - x - f(x)) \\ \frac{dy}{d\tau} &= \kappa(x - y + z) \\ \frac{dz}{d\tau} &= -\kappa(\beta y + \gamma z) \end{aligned} \tag{2.2}$$

kde: $\alpha, \beta, \gamma, \kappa$ sú kontrolné parametre sústavy, pričom $\kappa = \pm 1$,
$f(x)$ je nelineárna funkcia Chuaovej diódy,
x, y, z sú bezrozmerné súradnice reprezentujúce stav obvodu.

2.3 Chuaova dióda

Dôležitou časťou obvodu je Chuaova dióda. Je to aktívny dvojpól s oblasťou záporného diferenciálneho odporu (ZDR), čo znamená, že jeho volt-ampérová (VA) charakteristika má na istom intervale klesajúci charakter. Pôvodná Chuaova dióda bola charakterizovaná 3-segmentovou, po častiach lineárnou, nepárnou VA funkciou, avšak odvtedy bolo predstavených niekoľko ďalších variánt. V tejto práci sa zaoberáme s troma variantami Chuaovej diódy. Rozlišujeme ich podľa typu funkcie:

1. 3-segmentová;
2. 5-segmentová;
3. kubická.

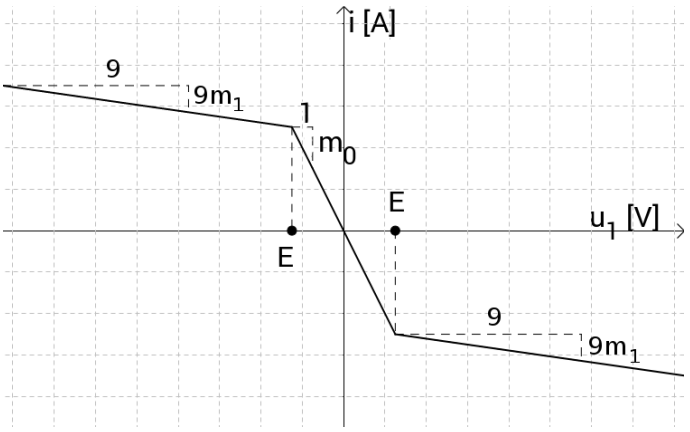
V nasledujúcich podkapitolách uvádzame ich matematické zápisy vo fyzikálnych jednotkách a v bezrozmernom tvare. Pre lepšiu predstavu sú uvedené aj grafy ich funkcií.

2.3.1 3-segmentová VA charakteristika

Matematický zápis 3-segmentovej VA charakteristiky je uvedený v (2.3), grafické znázornenie je na obrázku 2 – 2).

$$g(u_1) = m_1 u_1 + \frac{1}{2}(m_0 - m_1)(|u_1 + E| - |u_1 - E|)) \tag{2.3}$$

Parameter E v (2.3) reprezentuje bod zlomu v grafe, a parametre m_0, m_1 udávajú smernice priamky na daných úsekoch.



Obr. 2 – 2: VA charakteristika Chuaovej diódy

Bezrozmerný tvar funkcie je možné vidieť v (2.4). Parametre a, b zodpovedajú parametrom m_0, m_1 z (2.3).

$$f(x) = bx + \frac{1}{2}(a-b)(|x+E| - |x-E|)) \tag{2.4}$$

2.3.2 5-segmentová VA charakteristika

Matematický zápis 5-segmentovej VA charakteristiky je uvedený v (2.5), grafické znázornenie je na obrázku 2 – 3).

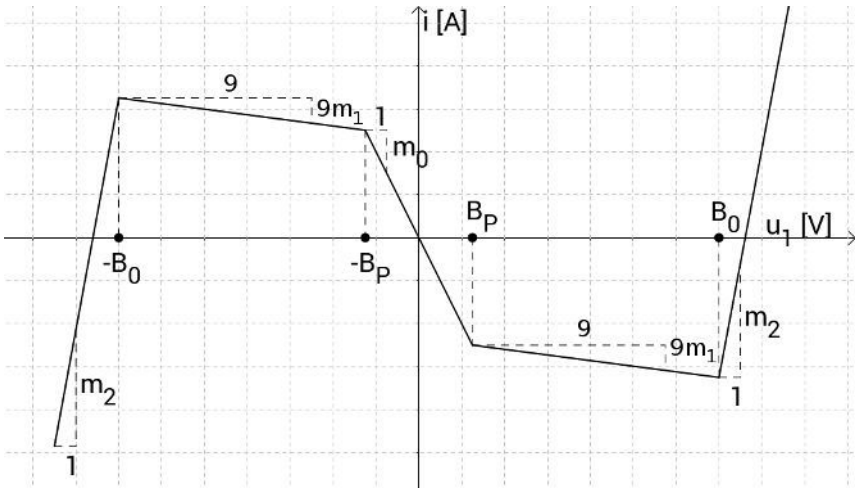
$$g(u_1) = m_2 u_1 + \frac{1}{2}(m_1 - m_0)(|u_1 - B_\mathrm{P}| - |u_1 + B_\mathrm{P}|) + \frac{1}{2}(m_2 - m_1)(|u_1 - B_0| - |u_1 + B_0|) \tag{2.5}$$

Parametre B_P a B_0 v (2.5) tvoria hranice segmentov, a parametre m_0, m_1 a m_2 reprezentujú smernice priamky na daných úsekoch.

5-segmentová funkcia Chuaovej diódy v bezrozmernom tvare je vyjadrená nasledovne:

$$f(x) = cx + \frac{1}{2}(b-a)(|x-E| - |x+E) + \frac{1}{2}(c-b)(|x-F| - |x+F|). \tag{2.6}$$

Parametre a, b, c v (2.6) zodpovedajú parametrom m_0, m_1, m_2 z (2.3) a parametre E, F parametrom B_P, B_0.

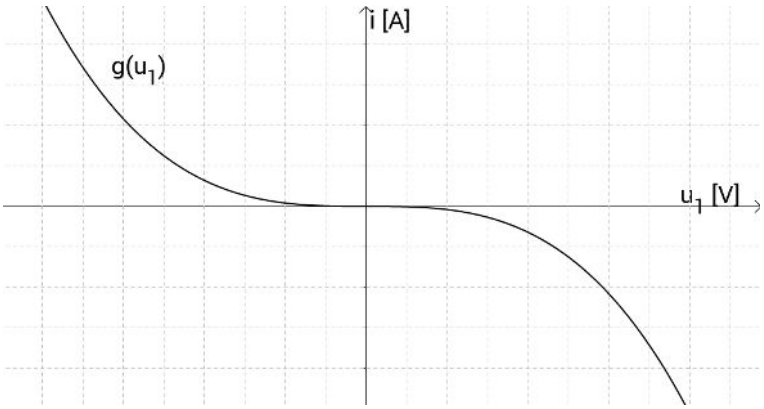


Obr. 2 – 3: VA charakteristika Chuaovej diódy

2.3.3 Kubická VA charakteristika

Matematický zápis kubickej VA charakteristiky je uvedený v (2.7), grafické znázornenie je na obrázku 2 – 4).

$$g(u_1) = h_0 + h_1 u_1 + h_2 {u_1}^2 + h_3 {u_1}^3 \tag{2.7}$$



Obr. 2 – 4: VA charakteristika Chuaovej diódy

Bezrozmerný zápis kubickej funkcie Chuaovej diódy môžeme vidieť v (2.8).

$$f(x) = h_0 + h_1 x + h_2 x^2 + h_3 x^3 \tag{2.8}$$

2.4 Fázový priestor

Fázový priestor slúži na názorné, geometrické priblíženie možného vývoja systému. Body vo fázovom priestore nazývame zastupujúcimi bodmi (ZB), ktoré reprezentujú stav systému v určitých časových okamihoch. Tieto body tvoria spojitú čiaru – trajektóriu ZB – pomocou čoho môžeme spozorovať dynamický vývoj systému [21]. Túto trajektóriu vo fázovom priestore nazývame aj fázovým portrétom systému. Fázový priestor Chuaovho obvodu je 3D priestor pozostávajúci z osí: i, u_1 a u_2.

2.4.1 Atraktory

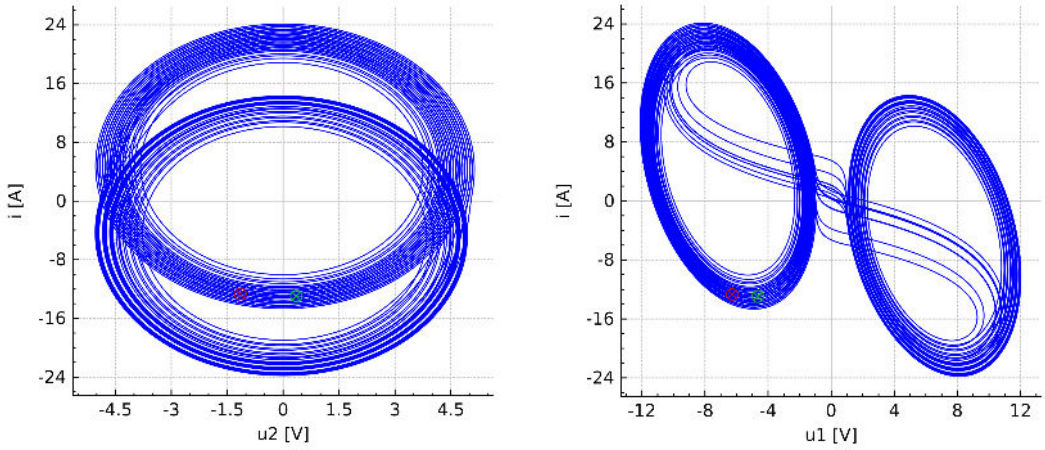
Trajektórie zastupujúceho bodu sú postupom času priťahované k istej podmnožine fázového priestoru, ktorú nazývame atraktorom. Typickým atraktorom lineárneho systému sú: bod, uzatvorená krivka, limitný cyklus alebo tórus. Pri nelineárnych systémoch sa však môžu pri istých začiatočných podmienkach (ZP) objaviť aj chaotické atraktory [21].

Pre parametre (2.9) sa Chuaov obvod v závislosti od ZP vyznačuje dvoma atraktormi: chaotickým atraktorom (CHA) a stabilným limitným cyklom (SLC).

$$\begin{array}{lllll} C_1 = 0{,}1\,\mathrm{F} & C_2 = 2\,\mathrm{F} & R = 1{,}428\,\Omega & L = 1/7\,\mathrm{H} & B_P = 1\,\mathrm{V} \\ B_0 = 14\,\mathrm{V} & \rho = 0\,\Omega & m_0 = -4\,\mathrm{S} & m_1 = -0{,}1\,\mathrm{S} & m_2 = 5\,\mathrm{S} \end{array} \tag{2.9}$$

Pre ilustráciu, pohľad na CHA vo fázovom priestore z dvoch rovín (u_2, i) a (u_1, i) je uvedený na obrázku 2 – 5. Trajektória pohybu ZB je vykreslená do integračného času $t = 250\,\mathrm{s}$ pri ZP $u_2 = 0{,}356\,\mathrm{V}$; $u_1 = -4{,}658\,\mathrm{V}$; $i = -12{,}954\,\mathrm{A}$.

Obvod pri rovnakých parametroch, ale iných začiatočných podmienkach vykazuje SLC. Na Obr. 2 – 6 je uvedený tento atraktor, ktorý bol vypočítaný pri ZP $u_2 = -15{,}069\,\mathrm{V}$; $u_1 = -15{,}072\,\mathrm{V}$; $i = 11{,}803\,\mathrm{A}$ pri dĺžke integračného času: $t = 250\,\mathrm{s}$.



(a) Projekcia do roviny u_2, i (b) Projekcia do roviny u_1, i

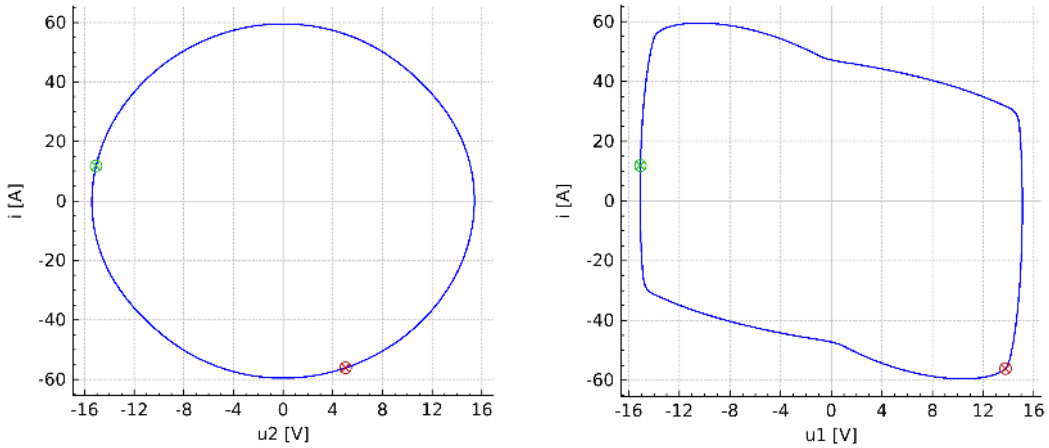
Obr. 2 – 5: Double-scroll chaotický atraktor Chuaovho obvodu

2.4.2 Výpočet trajektórie

Pri simulácii Chuaovho obvodu vychádzame zo sústavy (2.1). Keďže analytické riešenie tejto sústavy diferenciálnych rovníc nie je známe, je nutné sa spoliehať na numerické metódy. Na základe predchádzajúcich prác [15], [41], [3] pre riešenie sústavy bola vybraná metóda Runge-Kutta 4-ho rádu, ktorá je opísaná nasledujúcim vzťahom [15]:

$$\begin{aligned} k_1 &= f(x_n, y_n) \\ k_2 &= f(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_1) \\ k_3 &= f(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_2) \\ k_4 &= f(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_2) \\ y_{n+1} &= y_n + h\frac{k_1 + 2k_2 + 2k_3 + k_4}{6}. \end{aligned} \tag{2.10}$$

Vo vzťahu (2.10) parameter h reprezentuje integračný krok, pomocou ktorého je možné kontrolovať presnosť výpočtu. Na základe (2.10) boli odvodené rovnice pre výpočet systému troch diferenciálnych rovníc, ktoré je možné priamo aplikovať



(a) Projekcia do roviny u_2, i (b) Projekcia do roviny u_1, i

Obr. 2 – 6: Atraktor limitný cyklus

na výpočet Chuaovho obvodu. Rovnice vzhľadom na dĺžku ich zápisu sú uvedené v Prílohe A.

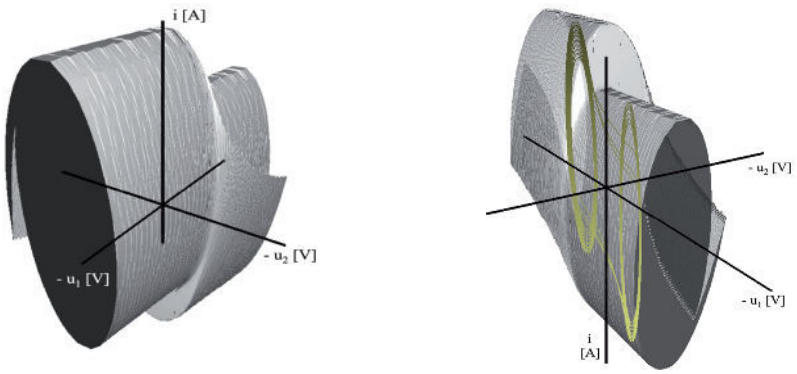
Ako to vyplýva aj z uvedených vzťahov, výpočet metódou Runge-Kutta spočíva v zistení výsledku na základe predchádzajúceho výsledku, teda pre zistenie výsledku v čase t_i musíme poznať aj hodnoty v časoch $t_0, t_1, ..., t_{i-1}$. Táto simulácia je preto výpočtovo pomerne náročná, obzvlášť pri vysokých hodnotách t a vysokej požadovanej presnosti. Napríklad pre zistenie výsledku v čase $t = 500$ pri integračnom kroku $h = 0{,}001$ potrebujeme 500 000 iterácií, avšak pri simulácií trajektórií sú bežné aj dlhšie a presnejšie výpočty.

2.5 Hraničná plocha

Hraničná plocha (HP) je dôležitou súčasťou analýzy v teórii nelineárnych obvodov. Je to objekt, pri ktorom sa dochádza k bifurkácii trajektórií – oddeľuje teda od seba jednotlivé atraktory v priestore. V grafickom 2D zobrazení (pomocou rezov HP) sa to prejaví farebným rozlíšením jednotlivých regiónov príťažlivosti (RP) [15].

HP v prípade Chuaovho obvodu a výskytu double-scroll CHA, je možné si predstaviť ako objekt tvaru rúry s výskytom (ostrého) „zubu“ pozdĺž jej boku. CHA je takto zapúzdrený hraničnou plochou, a všetky začiatočné podmienky zvolené vo vnútri rúry budú pritiahnuté ku CHA. Pre ZP, ktoré sa nachádzajú mimo rúry bude atraktorom nekonečno, alebo v prípade fyzikálneho systému SLC.

Pre lepšiu predstavu, je na Obr. 2 – 7 znázornená 3D rekonštrukcia HP Chuaovho obvodu. V ľavej časti obrázku, je vykreslená aj chaotická trajektória ZB. Podľa tejto ilustrácie si môžeme lepšie predstaviť, ako je CHA zapúzdrený HP.

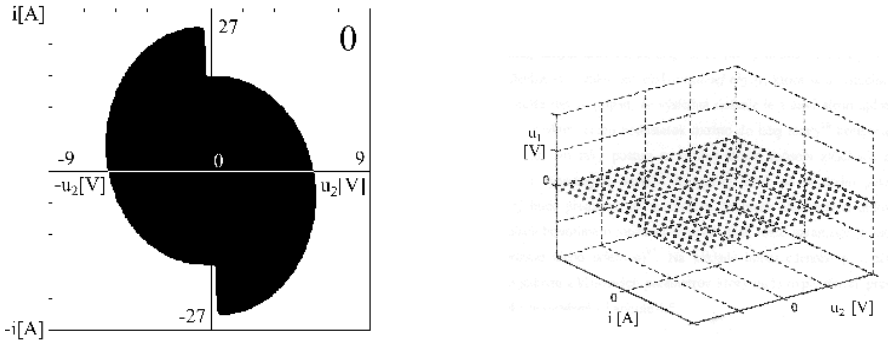


Obr. 2 – 7: HP Chuaovho obvodu v 3D, rekonštruovaná z 300 rezov [14]

2.6 Rezy hraničnými plochami

Rez 3D priestoru v geometrii znamená priesek objektu s rovinou. Môžeme si to predstaviť ako „rozkrájanie“ objektu v priestore na plátky, ktoré vytvárajú mnoho paralelných rezov. Rez 3D priestoru, ktorý leží v rovine kolmej na jednu z priestorových osí nazývame izočiarou, pretože všetky body v danom reze majú rovnakú tretiu súradnicu. Pri výpočte HP, sa používajú rezy. Sú názorné aj v 2D podobe, ako obrázok, ale po spojení série paralelných rezov je možné HP zobraziť aj v 3D projekcii.

Príklad rezu HP Chuaovho obvodu je uvedený na Obr. 2 – 8a. Rez bol vypočítaný zistením atraktora (CHA alebo SLC) pri 440×440 rôznych začiatočných podmienkach v rovine i, u_2 pre $u_1 = 0$ a parametre (2.9), pričom pri detekcii CHA bol zapísaný bod čiernej, a pri SLC bielej farby. Mriežka bodov reprezentujúcich začiatočné podmienky vo fázovom priestore je ilustrovaná na Obr. 2 – 8b. Dôležité je poznamenať, že HP nepredstavuje množina bodov vedúcich ku niektorým z atraktorov, ale kontúra medzi rozhraním oblastí rozličných farieb, napr. na Obr. 2 – 8a medzi čiernou a bielou farbou [15].



(a) (b)

Obr. 2 – 8: Rez HP (a), a mriežka ZP (b), v rovine i,u_2 pre $u_1 = 0\,\mathrm{V}$[15].

V tejto práci sa budeme ďalej zaoberať iba s izočiarami, teda s rezmi HP, ktoré ležia v rovinách kolmých na osi stavového priestoru, a to konkrétne v rovinách: i,u_2, i,u_1 a u_2,u_1.

Výpočet rezu HP prebieha zistením atraktorov pre trajektórie so začiatočnými podmienkami z istej oblasti roviny stavového priestoru. Pre každú začiatočnú podmienku je zapísaný jeden bod do výsledného rastra na zhodnú pozíciu, rozlíšenie výsledku je teda vždy zhodné s počtom začiatočných podmienok. Farby jednotlivých bodov sú určené podľa atraktora na danej pozícii a môžu byť zvolené ľubovoľne. Napríklad na reze ilustrovaného na Obr. 2 – 8a, ktorý bol vypočítaný na základe začiatočných podmienok zobrazených na Obr. 2 – 8b, sú začiatočné podmienky priťažené ku CHA označené čiernou, kým k SLC bielou farbou.

3 Softvér pre výskum Chuaovho obvodu

Simulácia elektrických obvodov šetrí čas aj peniaze. Ak by sme chceli skúmať rôzne varianty obvodu experimentálnym spôsobom, bolo by potrebné zostaviť obvod pre každú variantu. Pri softvérovej simulácie namiesto realizácie fyzických obvodov stačí zmeniť parametre výpočtu, a skúmanie novovzniknutého obvodu sa môže začať ihneď. Hodnoty prúdov a napätí dostávame priamo pri výpočte, ktorého numerickú presnosť je možné zvýšiť, kým pri fyzickom obvode pracujeme s meradlami, ktoré nemusia byť dostatočne presné. Softvérová simulácia vychádza z teoretického modelu obvodu, ktorý ráta s ideálnymi obvodovými prvkami. Vypočítané údaje teda nie sú ovplyvnené vonkajšími vplyvmi, ako teplota, výkyvy napätia zdroja, alebo výrobná kvalita súčiastok. Podmienky simulácie sú presne kontrolovateľné, kým nad fyzickými meraniami nikdy nie je možné zabezpečiť úplnú kontrolu. Našim hlavným zámerom je skúmať chaotické javy, a tie sú veľmi citlivé aj na malé zmeny, pri fyzicky konštruovaných obvodoch môžu spomenuté vplyvy viesť k nekontrolovateľným, neopakovateľným výsledkom. Skúmanie chaotických javov v obvodoch pomocou simulácie môže byť teda výhodnejším prístupom z viacerých hľadísk. Základným predpokladom simulácie je však vysoký výpočtový výkon [15].

3.1 Špecifikácia požiadaviek

Na základe zadania diplomovej práce a pokynov vedúceho a konzultanta, na softvér, boli stanovené nasledujúce funkčné a nefunkčné požiadavky:

3.1.1 Funkčné požiadavky

1. Simulácia Chuaovho obvodu s možnosťou zmeny všetkých parametrov vrátane VA charakteristiky;

2. Vizualizácia pohybu zastupujúceho bodu pre vypočítanú trajektóriu pomocou Mongeovej a 3D projekcie;

3. Výpočet a vizualizácia rezov HP v rôznych rovinách s možnosťou automatizovaného výpočtu série rezov;

4. Možnosť zadávania ľubovoľného počtu testovacích podmienok pre počítanie rezov HP a ich vizualizácie vzhľadom k trajektórii ZB;

5. Možnosť exportovať všetky vypočítané údaje do formátu podporovaného tabuľkovým editorom MS Excel, do súboru *txt* a ďalších formátov.

3.1.2 Nefunkčné požiadavky

1. *Výkon* – cieľom je vykonávanie časovo náročných výpočtov. Dôležité je, aby aplikácia využívala zdroje počítača čo najefektívnejšie. Kód aplikácie má byť preto čo najviac optimalizovaný pre výkon, aplikácia má obsahovať možnosti zmeny presnosti výpočtu a pre náročnejšie výpočty má využívať všetky procesorové jadrá;

2. *GUI* – aplikácia má disponovať s intuitívnym grafickým používateľským rozhraním, pomocou ktorého je ovládateľná jej celá funkcionalita;

3. *Multiplatformovosť* – aplikácia má byť prekladateľná a spustiteľná na 32 a 64 bitových verziách systému Windows a Linux.

3.2 Existujúce riešenia

Keďže niekoľko publikácií už sa zaoberalo simuláciou alebo vizualizáciou Chuaovho obvodu, pred začatím návrhu programu, prvým krokom bolo vyhľadanie existujúcich riešení. Bolo nájdených päť softvérových riešení, ktoré aspoň čiastočne spĺňajú

funkčné požiadavky. V tejto časti práce predstavíme nájdené riešenia, a uvedieme ich výhody/nevýhody vzhľadom na vyššie uvedené požiadavky.

3.2.1 MATLAB

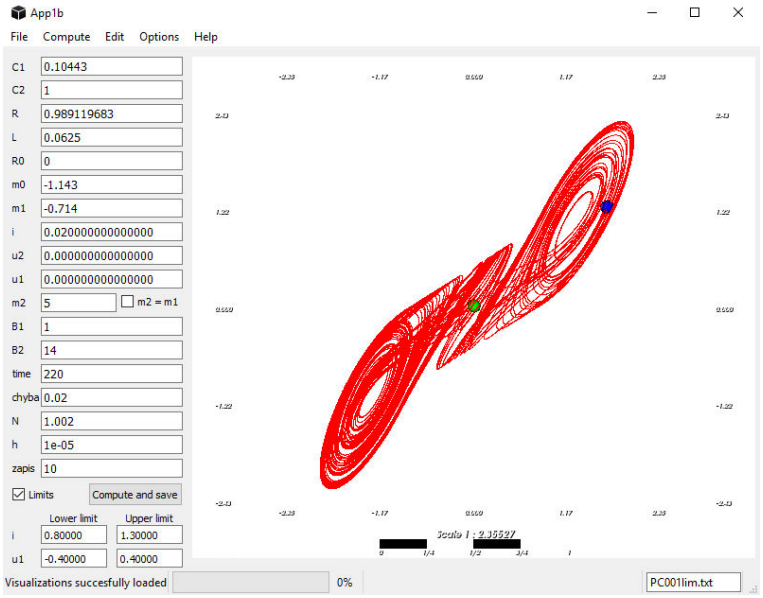
Softvér *MATLAB* je univerzálnym nástrojom a zároveň aj programovacím jazykom na vykonanie náročných matematických operácií, ako počítanie maticami, analýza funkcií, numerický výpočet integrálov, diferenciálnych rovníc, atď. Poskytuje vlastný programovací jazyk, pomocou ktorého je možné implementovať ľubovoľný algoritmus a výsledky umožní jednoducho vizualizovať v 2D alebo 3D. Preto mnoho štúdií vybralo pre simuláciu Chuaovho obvodu práve tento softvér. Následkom univerzálnosti *MATLABu* však je, ťažšia použiteľnosť programu, pre použitie je nutné vedieť programovať v jazyku *MATLAB*. Ďalšou nevýhodou *MATLABu* je jeho výrazne nižší výkon oproti ďalším programovacím jazykom, na ktorú poukazuje aj štúdia [1].

3.2.2 App1b

Aplikácia *App1b*, ktorá bola predstavená v práci [22], slúži pre výpočet a vizualizáciu trajektórie Chuaovho obvodu. Pracuje s parametrami, ktoré sa zadávajú cez používateľské rozhranie alebo sú načítané zo vstupného súboru. Vypočítané body trajektórie sa ukladajú do textového výstupného súboru a vizualizácia je realizovaná v perspektívnej 3D projekcii pomocou mračna bodov. Program je realizovaný v programovacom jazyku C++ a knižnice *Qt*. Snímka hlavného okna programu je ilustrovaná na Obr. 3 – 1.

Výhody riešenia:

- možnosť presného nastavenia parametrov obvodu a ďalších hodnôt, ktoré ovplyvňujú presnosť výpočtu;



Obr. 3 – 1: Hlavné okno programu *App1b*

- umožňuje prepnúť medzi 3-segmentovou a 5-segmentovou VA charakteristikou Chuaovej diódy;
- exportovanie vypočítaných bodov trajektórie;
- 3D vizualizácia trajektórie, s možnosťou priblíženia, posúvania, otáčania.

Nevýhody riešenia:

- chýba funkcionalita počítania a vizualizácie HP;
- nie je dostupná ortografická projekcia trajektórie a hodnoty z mriežky v 3D projekcii sú ťažko odpočítateľné;
- neumožňuje zadať kubickú VA charakteristiku Chuaovej diódy.

3.2.3 PC1IU1

Konzolová aplikácia *PC1IU1* bola predstavená v práci [3]. Vstupom je textový súbor s parametrami výpočtu a výstupom je vypočítaný rez HP Chuaovho obvodu v rovine i,u_1, taktiež v textovom formáte. Aplikácia používa technológiu OpenCL pre paralelizáciu výpočtu na CPU alebo GPU.

Výhody riešenia:

- využíva paralelizáciu pre urýchlenie výpočtov. Pomocou technológie OpenCL dokáže pre výpočty využiť aj GPU;
- je možné prispôsobiť parametre obvodu a sú dostupné ďalšie parametre pre prispôsobenie presnosti výpočtu.

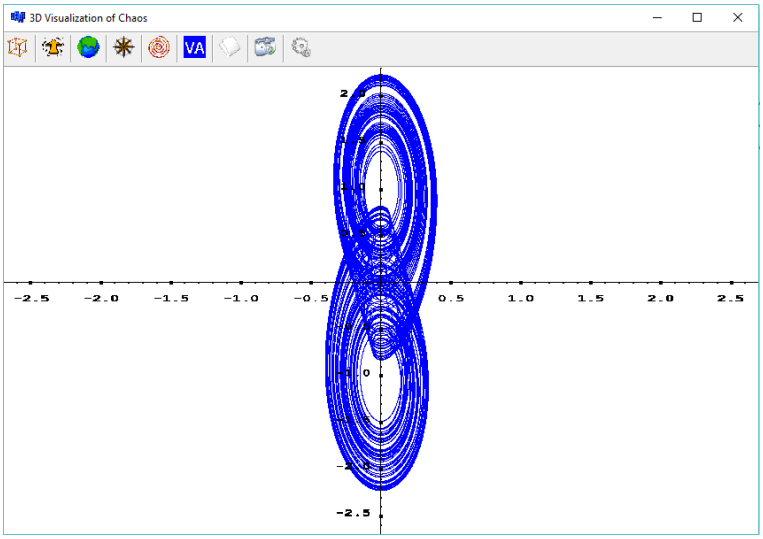
Nevýhody riešenia:

- nedisponuje grafickým používateľským rozhraním, nie sú teda dostupné žiadne možnosti vizualizácie;
- nie je možné uložiť samotné trajektórie;
- podporuje iba 5-segmentovú VA charakteristiku Chuaovej diódy;
- testovacie podmienky pre počítanie rezov HP sú pevne dané v zdrojovom kóde.

3.2.4 Chaosviz

Táto aplikácia sa zameriava na vizualizáciu chaotického atraktora, generovaného Chuaovým obvodom. Bola prezentovaná v diplomovej práci [28] a umožňuje zobrazenie trajektórie a VA charakteristiky v 2D a 3D projekciách. Pri vizualizácii poskytuje mnoho možností, ako: otáčanie a priblíženie trajektórie, zmena hrúbky a farby trajektórie a osí, využitie efektu „kométy“ pri vykreslovaní alebo nastavenie

rýchlosti vykreslenia. Údaje o zobrazenej trajektórii a VA charakteristike aplikácia načíta zo vstupných textových súborov. Snímku aplikácie je možné vidieť na Obr. 3 – 2.



Obr. 3 – 2: Snímka programu *Chaosviz*

Výhody riešenia:

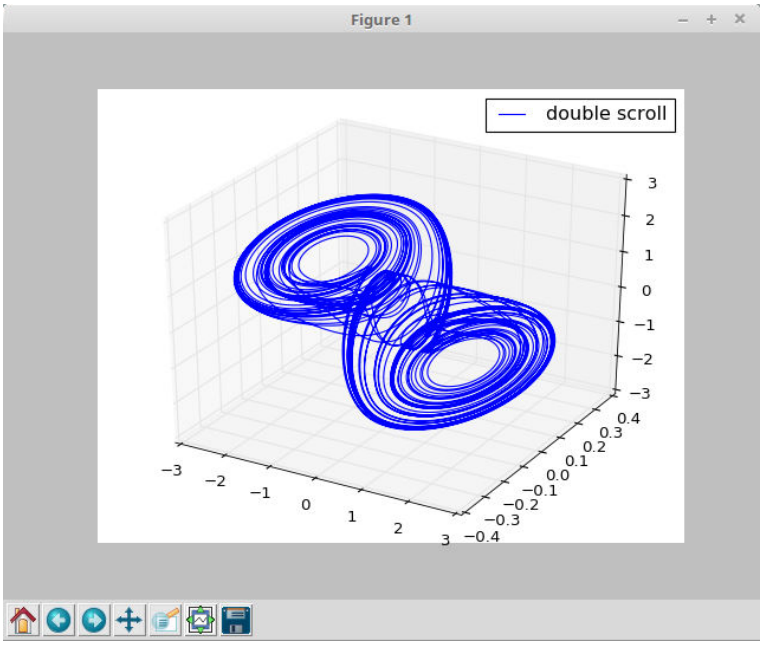
- vizualizácia trajektórie v 2D a 3D s množstvom nastaviteľných možností;
- možnosť zobrazenia VA charakteristiky

Nevýhody riešenia:

- aplikácia nevykonáva výpočet trajektórie, tá musí byť vypočítaná iným programom a dodaná vo forme textového súboru;
- nepočíta a nezobrazuje rezy HP;
- program bol vyvinutý pomocou *Borland C++ Builder*, ktorý nepodporuje Linux. Aplikácia teda nevyhovuje podmienke o multiplatformovosti.

3.2.5 Chua's Circuit

Projekt *Chua's Circuit* [29] je sadou skriptov v jazyku Python, ktoré umožňujú výskum atraktorov Chuaovho obvodu. Riešenie obsahuje tri typy skriptov: 1) vizualizácia trajektórie ZB v ortografickej 3D projekcii (Obr. 3 – 3), 2) zobrazenie ZB v 3D projekcii v reálnom čase, 3) výpočet Ljapunových exponentov a ich vizualizácia formou grafu.



Obr. 3 – 3: 3D Vizualizácia double-scroll atraktora pomocou *Chua's Circuit*

Výhody riešenia:

- vizualizácia trajektórie pomocou v 3D;
- možnosť výpočtu Ljapunových exponentov;
- riešenie je implementované v platformovo-nezávislom jazyku Python.

Nevýhody riešenia:

- parametre obvodu a začiatočné parametre sú zadefinované priamo v zdrojovom

kóde;

- chýba možnosť nastavenia 2D projekcie;
- vypočítané údaje nie je možné exportovať v textovom formáte;
- riešenie neposkytuje výpočet rezov HP.

3.3 Zhodnotenie existujúcich riešení

Všetky skúmané softvéry uvedené vyššie sú zaujímavými riešeniami pre výskum Chuaovho obvodu. Aplikácie *App1b*, *Chaosviz* a *Chua's Circuit* poskytujú široké možnosti vizualizácie jednej trajektórie, avšak chýba z nich možnosť výpočtu HP a jej zobrazenie pomocou rezov. Aplikácia *PC1IU1* – keďže využíva aj jadrá GPU – je z hľadiska výkonu vynikajúcim riešením pre výpočet HP, neumožňuje však výpočet jednej trajektórie a neposkytuje žiadne možnosti vizualizácie. Všetky funkčné požiadavky kladené na softvér z uvedených riešení je možné uspokojiť iba s *MATLABom*, avšak ten vyžaduje dôkladné znalosti softvéru pre správne nastavenie. Ďalšími vymedzujúcimi činiteľmi *MATLABu* sú nižší výkon a menej intuitívne GUI.

Môžeme si teda usúdiť, že ani jedno z vyššie uvedených riešení nespĺňa všetky stanovené požiadavky v dostatočnej miere. Preto v rámci riešenia tejto práce bolo rozhodnuté o vývoji novej aplikácie.

4 Návrh aplikácie Chuaviz

Z analýzy požiadaviek a existujúcich riešení v predchádzajúcej kapitole vyplynula nutnosť vývoja novej aplikácie. Projekt dostal názov *Chuaviz*.

4.1 Výber technológií

Prvým krokom pri vývoji bol výber vhodného programovacieho jazyka, frameworku, knižníc a ostatných vývojových nástrojov. Vychádzajúc z funkčných a nefunkčných požiadaviek aplikácie popísaných v kapitolách 3.1.1 a 3.1.2, boli zohľadnené pri výbere technológie: vysoký výpočtový výkon, kvalitné nástroje pre tvorbu používateľského rozhrania a multiplatformovosť. Pri výbere programovacieho jazyka, bolo dôležité, aby v danom jazyku bolo možné využívať objektovo-orientovaný prístup kvôli lepším možnostiam abstrakcie, organizácie kódu a znovapoužiteľnosti.

Podľa testov uvedených v publikáciách [30] a [11], C a C++ sú najrýchlejšie programovacie jazyky v porovnaní s ostatnými populárnymi jazykmi, ako: JAVA, C#, Perl, Python. C a C++ majú takmer rovnaký výkon, C++ je však modernejší jazyk s podporou objektovo-orientovaného prístupu. Kód napísaný v štandardnom C++ je prekladateľný pre viaceré platformy a existuje k nemu široká škála voľne dostupných knižníc. Prekladá sa priamo do strojového kódu, vďaka čomu poskytuje vyšší výkon oproti interpretovaným jazykom, a veľa možností optimalizácie. Na základe požiadaviek, pre realizáciu aplikácie bol teda vybraný programovací jazyk C++.

Od konečnej aplikácie očakávame vykresľovanie farebných grafov, bitmáp a pomerne zložitých formulárov. Samotné C++ neposkytuje žiadne prostriedky pre vývoj grafického používateľského rozhrania. Preto ďalším dôležitým krokom bol výber vhodnej grafickej knižnice. Po výskume a porovnaní dostupných riešení, pre realizáciu grafického rozhrania programu bol zvolený softvérový rámec *Qt*.

Qt je veľmi rozsiahle riešenie pre vývoj aplikácií. Okrem riešenia pre budovanie grafických rozhraní poskytuje veľké množstvo ďalších knižníc, ale aj sadu vývojových nástrojov. *Qt* je dostupné aj pre iné programovacie jazyky, avšak v tejto práci ho budeme používať výlučne s jazykom C++.

Pozrime sa, ako *Qt*/C++ spĺňa vyššie uvedené technologické požiadavky:

1. *Výkon* – kód písaný v jazyku C++ sa prekladá do strojového kódu pomocou prekladača, čo znamená že vygenerovaný kód je priamo vykonaný procesorom počítača. Okrem toho poskytuje mnoho ďalších optimalizácií výkonu, napr.: doplnenie častí kódu v jazyku asembler, makrá pre vektorové CPU inštrukcie (napr.: SSE, MMX, AVX), kľúčové slová, ako `const`, `inline` alebo `register`, a optimalizácie dané prekladačom. Ďalej je k dispozícii k nemu množstvo knižníc zameraných na výpočty a paralelizáciu, ako: OpenCL, Intel TBB, Nvidia CUDA, Pthreads, atď. Taktiež podporuje natívne systémové volania bez potreby ďalšej softvérovej vrstvy.

2. *Objektovo orientovaný prístup* – je štandardnou metódou programovania v jazyku C++. Jazyk však podporuje aj procedurálne programovanie so syntaxom rovnakým jazyku C. Rámec *Qt* je postavený na objektovo orientovanej architektúre.

3. *Grafické rozhranie* – *Qt* v začiatku vzniklo práve pre potrebu tvorby používateľských rozhraní, charakterizovaným rýchlosťou, nízkou odozvou a stabilitou [6].

4. *Multiplatformovosť* – pre štandardný jazyk C++ existuje prekladač pre všetky rozšírené operačné systémy a počítačové architektúry. *Qt* podporuje Windows, Linux, OS X a neoficiálne aj operačné systémy BSD. Taktiež umožňuje vývoj pre mobilné zariadenia na platformách Android, IOS a Windows Phone [37].

4.2 Softvérový rámec Qt/C++

Qt v dobe jeho vzniku slúžil výlučne pre tvorbu grafických používateľských rozhraní, v tej dobe sa nazýval *Qt toolkit*. Odvtedy však jeho funkcionalita bola vo veľkej miere rozšírená, aktuálne pod *Qt* rozumieme "komplexný aplikačný a GUI vývojový framework" [6]. Framework *Qt* aktuálne poskytuje dve možnosti licencovania: slobodnú licenciu a komerčnú licenciu. Slobodná licencia umožňuje vývoj aplikácií s otvoreným zdrojovým kódom (open-source), ale aj komerčné aplikácie. Podmienkou je splnenie pravidiel licencie GNU GPL alebo GNU LGPL. Aktuálna verzia frameworku *Qt* je 5. Qt sa skladá z niekoľkých modulov, ktoré sú rozdelené do dvoch skupín:

- Qt Essentials;
- Qt Add-Ons.

4.2.1 Moduly Qt Essentials

Tieto moduly tvoria solídny základ frameworku *Qt* a sú dostupné na všetkých platformách. Podľa dokumentácie, moduly Qt Essentials ostávajú konzistentné a spätne kompatibilné počas celého životného cyklu aktuálnej verzie frameworku [36]. Qt Essentials tvoria nasledujúce moduly:

- *Qt Core* – jadrová knižnica, na ktorej sú závislé aj ďalšie moduly. Obsahuje všeobecné rozšírenia jazyka C++ (ako údajové štruktúry, spôsob komunikácie medzi objektmi, atď.) na ktoré sú ostatné moduly stavané. Tento modul neobsahuje žiadnu grafickú časť.
- *Qt GUI* – základné triedy určené pre budovanie grafických používateľských rozhraní. Obsahuje v sebe aj integráciu pre OpenGL pre 2D grafiku.

- *Qt Widgets* – rozšírenie modulu Qt GUI o grafické komponenty, takzvané widgety. Widgety používateľského rozhrania komunikujú s hlavnou aplikáciou pomocou odovzdania správ, pričom každý widget má definované vlastné správanie a vzhľad. Modul Qt Widgets obsahuje bežne používané widgety v desktopových používateľských rozhraniach, ako sú textové polia, tlačidlá, tabuľky, menu, atď. Je možné však vytvoriť aj vlastné widgety s ľubovolnou funkcionalitou, alebo rozšíriť existujúce.

- *Qt Multimedia* – pomocné triedy pre prácu s audiom, videom a s audiovizuálnymi hardvérovými zariadeniami.

- *Qt Multimedia Widgets* – umožňuje vyniesť multimediálny obsah do používateľského rozhrania pomocou rozšírenia modulu Qt Widgets o multimediálne prvky, napr: video okno.

- *Qt Network* – triedy pre zjednodušenie sieťovej komunikácie.

- *Qt QML* – rozšírenie frameworku *Qt* o podporu jazyka QML a JavaScript.

- *Qt Quick* – tento modul integruje jazyk QML a JavaScript do frameworku a tak umožňuje vytvárať plnohodnotné aplikácie pomocou týchto jazykov. K využitiu tohto prístupu sú dostupné ďalšie moduly *Qt Quick Controls*, *Qt Quick Dialogs* a *Qt Quick Layouts*.

- *Qt SQL* – triedy pre prácu s databázami.

- *Qt Test* – triedy pre uľahčenie testovania aplikácií vytvorených v Qt [36].

4.2.2 Moduly Qt addons

Qt ponúka aj ďalšie moduly pod názvom Qt Add-Ons. Tieto moduly obsahujú rozšírenú funkcionalitu frameworku a nemusia byť dostupné na všetkých platformách.

Keďže zoznam týchto modulov je pomerne dlhý, spomenieme iba tie najpodstatnejšie, ktoré sú dostupné na viacerých platformách a môžu byť užitočné pri tvorbe našej aplikácie:

- *Qt Concurrent* – pomocné triedy umožňujúce spustenie vlákien programu. Poskytuje implementáciu paralelných algoritmov map, filter, map-reduce a filter-reduce.
- *Qt Image Formats* – knižnica pre prácu s bežnými formátmi obrázkov, ako napríklad: PNG, JPEG, BMP, GIF.
- *Qt Print Support* – poskytuje multiplatformový prístup k tlačiarni.
- *Qt Serial Port* – nástroje umožňujúce I/O operácie so sériovým portom počítača.
- *Qt SVG* – umožňuje zobrazenie vektorových obrázkov vo formáte SVG.
- *Qt WebEngine Widgets* – poskytuje nový widget, v ktorom je možné zobrazovať webové stránky a webové aplikácie.
- *Qt XML* – pomocné triedy pre manipuláciu s XML súbormi.

Úplný zoznam *Qt Add-Ons* je sa nachádza v oficiálnej dokumentácie frameworku *Qt* pod sekciou “All modules” [36].

4.2.3 Sada vývojových nástrojov

Framework *Qt* neponúka iba knižnice, ale i širokú sadu nástrojov pre vývoj softvéru:

- *qmake* – multiplatformový zostavovací nástroj.
- *Qt Creator* – integrované vývojové prostredie optimalizované pre knižnicu Qt.

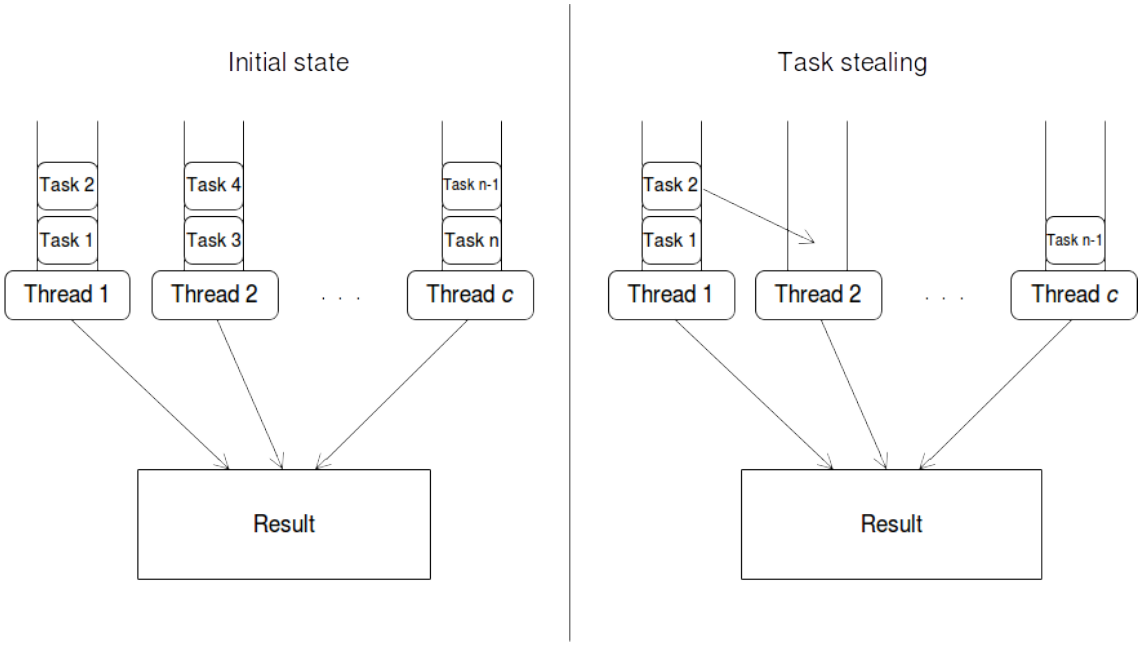
- *Qt Designer* – nástroj pre návrh GUI.
- *Qt Assistant* – nástroj pre prácu s dokumentáciou.
- *Qt Linguist* – nástroj pre internacionalizáciu aplikácií [6].

Jedine prekladač nie je distribuovaný spolu s vývojovými nástrojmi frameworku *Qt*. Ten je potrebný nainštalovať z tretích zdrojov. *Qt Creator* bez ďalších nastavení podporuje bežné prekladače: GCC, ICC, Clang, MinGW a MS Visual Studio [37].

4.3 Intel TBB

Intel TBB je softvérová knižnica, ktorá umožňuje v jazyku C++ vytvárať paralelné programy, ktoré sú prenosné medzi počítačmi. Programátorom poskytuje API na vyššej abstrakčnej úrovni, ako sú štandardné vlákna operačného systému. Knižnica umožňuje definovať úlohy (tasks), ktoré sú rozdelené dynamicky medzi dostupnými procesorovými jadrami. Rozdelenie úloh spravuje komponent *Task Scheduler*, ktorý využitím mechanizmu *task stealing* dokáže vyrovnať nerovnomernú záťaž medzi vláknami [7].

Obr 4 – 1 ilustruje ako funguje mechanizmus *task stealing* pri vláknach. Po spustení výpočtu sa vytvorí c vlákien, pričom c sa predvolene rovná počtu dostupných (fyzických alebo virtuálnych) CPU jadier. Na diagrame je ilustrovaný prípad, keď počet úloh $n = 2c$, čiže na začiatku každé vlákno má pridelené dve úlohy. Predpokladajme, že 1. úloha (Task 1) trvá podstatne dlhšie, ako ostatné. Druhé vlákno dokončí všetky svoje úlohy ale prvá má ich ešte dve – jednu aktuálne spracováva a druhú vo fronte. Táto situácia je ilustrovaná na pravej strane diagramu na Obr. 4 – 1. Pri tomto prípade zapôsobí *task stealing* mechanizmus, a druhá úloha z prvého vlákna sa presunie do vlákna 2. Tým pádom vlákno 2 neostane nečinným a celková účinnosť paralelizácie sa zvýši.



Obr. 4 – 1: Rozdelenie úloh medzi vláknami a *task stealing*

Intel TBB poskytuje aj niekoľko vopred definovaných všeobecných paralelných algoritmov vo forme C++ šablón, ako:

- *parallel_for* – Umožňuje jednoduchú paralelizáciu for cyklov s nezávislými výpočtami v jednotlivých iteráciách;
- *parallel_reduce* – Paralelný výpočet jednej hodnoty na základe viacerých prvkov (napr. suma čísel);
- *parallel_deterministic_reduce* – Podobný algoritmu *parallel_reduce* s tým rozdielom, že výpočet je zaručene vždy rozdelený na rovnaký počet podvýpočtov;
- *parallel_scan* – Implementácia paralelného algoritmu Prefix sum;
- *parallel_do* – Paralelizácia cyklov, pri ktorých počet iterácií nie je vopred známy (cyklus while);
- *parallel_pipeline* – Umožňuje aplikovať viac funkcií na prvky, pričom niektoré

funkcie môžu byť paralelne vykonateľné a niektoré nie;

- *parallel_foreach* – Aplikuje jednu funkciu na každý prvok poľa;
- *parallel_sort* – Umožňuje paralelné triedenie prvkov;
- *parallel_invoke* – Paralelné volanie ľubovoľných funkcií typu `void` [19].

4.4 QCustomPlot

Jednou z kľúčových požiadaviek kladených na aplikáciu je vizualizácia údajov. Pre splnenie tohto účelu bola zvolená otvorená knižnica *QCustomPlot* [8]. Táto knižnica rozširuje štandardné *Qt* o nový prvok, ktorý umožňuje 2D vizualizáciu údajov pomocou bežných grafických znázornujúcich tvarov, ako sú grafy, krivky, histogramy, atď. V rámci jedného widgetu je možné zobraziť aj viac tvarov, knižnica okrem iných ďalej ponúka pridávanie popisov, zmeny mierky osí, prispôsobenie farieb a písmen, interakcie (posúvanie, priblíženie) a vizualizáciu v reálnom čase.

Nasledujúce triedy pre vizualizáciu sú poskytnuté knižnicou:

- *QCPGraph* – zobrazenie množiny bodov v 2D rovine, pričom k jednému bodu na zvislej osi môže byť priradený najviac jeden bod na vodorovnej osi. Premietané body sú voliteľne pospájané do súvislej čiary.
- *QCPCurve* – zobrazenie ľubovoľnej krivky v 2D rovine.
- *QCPBars* – vykreslenie stĺpcových diagramov. Výška a šírka stĺpcov je automaticky vypočítaná podľa zadaných údajov, stĺpce pre lepšiu názornosť je možné aj zoskupovať.
- *QCPStatisticalBox* – vykreslenie krabicových grafov.
- *QCPColorMap* – slúži na znázornenie 3D údajov, pomocou 2D máp, pričom

tretí rozmer je zakódovaný, ako farba bodu.

- *QCPFinancial* – zobrazenie údajov v časovej závislosti, v prvom rade určené pre burzové grafy.

Pre aplikáciu *Chuaviz* sú zaujímavé možnosti *QCPCurve* pre zobrazenie trajektórie ZB a *QCPColorMap* pre zobrazenie rezov HP.

4.5 QPlot3D

QPlot3D je malá knižnica, ktorá rozširuje *Qt* o nový prvok, v ktorom je možné zobraziť krivky z množiny bodov v 3D. Dostupné sú rôzne nastavenia, ako prispôsobenie farby a hrúbky kriviek, zobrazenie osí a legendy, atď. Umožňuje aj používateľské interakcie: posúvanie, otáčanie a priblíženie zobrazeného 3D objektu. Projekcia je uskutočnená v perspektíve a využíva knižnicu OpenGL, ktorá využíva grafickú kartu pre grafické transformácie potrebné k zobrazeniu 3D scény [35].

4.6 Vývoj aplikácie

Vývoj aplikácie prebiehal na 64 bitovej verzii operačného systému Linux Mint 18. Vyvíjalo sa vo vývojovom prostredí *Qt Creator* s prekladačom GCC. Pre preklad do Windowsovej platformy bol použitý prekladač MS Visual Studio 2013.

4.6.1 Návrh výpočtov

Z funkčných požiadaviek kladených na aplikáciu vyplýva, že má poskytovať tri druhy výpočtov:

1. Jednej trajektórie pohybu ZB;

2. Jedného rezu HP;

3. Série rezov HP.

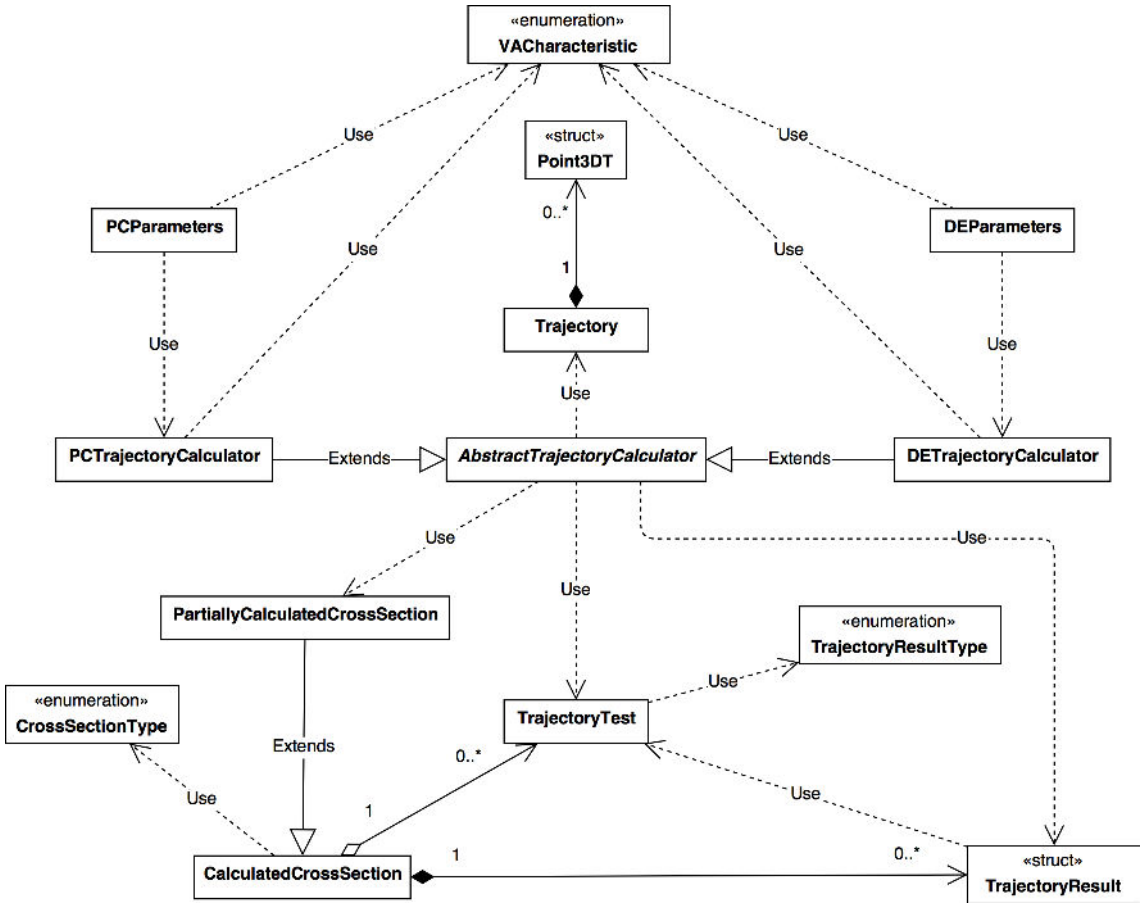
Princíp výpočtu jednej trajektórie ZB je uvedený v kapitole 2.4.2. Výpočet jednej trajektórie ZB bude vykonaný za účelom jej vizualizácie, preto bude potrebné uchovať každý vypočítaný bod. Pri výpočte rezov HP sa počíta množina trajektórií rovnakou metódou (Runge-Kutta), ako pri jednej trajektórie. Rozdielom však je, že tu sa na vypočítané body aplikujú testovacie podmienky pre zistenie atraktora, a k dosiahnutiu výsledného rezu nepotrebujeme uchovať vypočítané body trajektórie, iba zistený atraktor pre dané ZP. Pre sériu rezov môžeme výpočet jedného rezu niekoľkokrát opakovať so zmenou tretej súradnice.

Pre realizáciu výpočtov bol navrhnutý logický modul *Calculation*. Zámerom pri návrhu bola jeho ľahká rozšíriteľnosť pre rôzne iné systémy výpočtu trajektórií. Pre vykonanie výpočtov poskytuje triedy `PCTrajectoryCalculator`, ktorá implementuje výpočet fyzikálneho obvodu podľa (2.1), a `DETrajectoryCalculator` implementujúca bezrozmerný tvar podľa (2.2). Obidve triedy dedia od abstraktnej triedy `AbstractTrajectoryCalculator`, ktorá obsahuje dve abstraktné metódy pre výpočet bodov trajektórie a výpočet výsledku trajektórie. Potomkovia musia tieto metódy implementovať. V `AbstractTrajectoryCalculator` je implementovaný sekvenčný aj paralelný výpočet rezov HP, nie je teda nutné tieto výpočty implementovať v potomkoch zvlášť. S týmto návrhom je možné v budúcnosti modul rozšíriť o ďalšie systémy diferenciálnych rovníc jednoducho – s implementáciou iba dvoch metód.

Modul *Calculation* ďalej obsahuje nasledujúce pomocné údajové štruktúry a triedy: `Point3DT` – súradnice ZB v istom čase, `PCParameters` – parametre výpočtu pre fyzikálny obvod, `VACharacteristic` – VA charakteristika diódy (enumerácia), `DEParameters` – parametre výpočtu pre bezrozmerný obvod, `Trajectory` – vypočítaná trajektória s ukladaním bodov, `TrajectoryTest` – testovacia pod-

mienka atraktora, `TrajectoryResult` – výsledok výpočtu pre jeden bod HP, `CrossSectionType` – rovina rezu HP (enumerácia), `CalculatedCrossSection` – vypočítaný rez HP, `PartiallyCalculatedCrossSection` – čiastočne vypočítaný rez (používa sa pri prebiehajúcom výpočte) a `TrajectoryResultType` – typ atraktora trajektórie (enumerácia).

Zjednodušený diagram modulu, ktorý naznačuje vzťahy medzi vyššie spomenutými triedami, je možné vidieť na Obr 4 – 2. Plný diagram, spolu s detailným popisom tried a ich metód je uvedený na Obr. D – 5, v prílohe D (systémová príručka).



Obr. 4 – 2: Zjednodušený diagram tried modulu *Calculation*

Modul *Calculation* bol navrhnutý tak, aby nebol závislý od ostatných častí aplikácie a bol znovupoužiteľný. V zdrojovom kóde aplikácie sú spomínané konštruk-

čné prvky umiestnené v priečinku `calculation`, modul je možné distribuovať aj zvlášť, ako knižnicu a použiť ho aj v iných programoch.

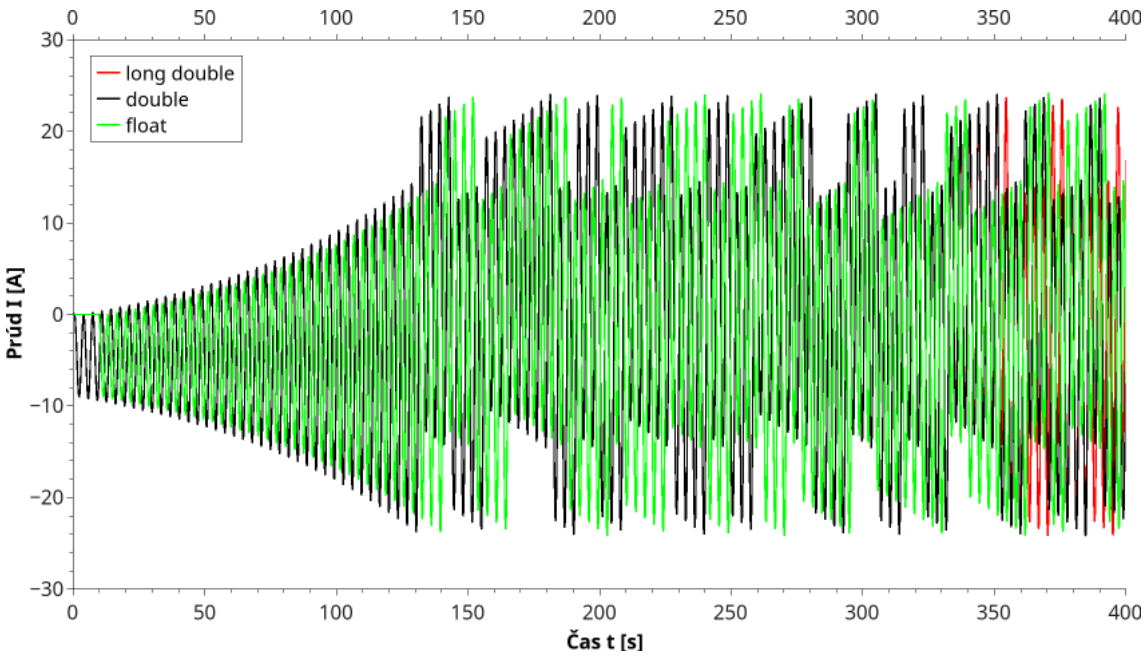
Parametre výpočtu trajektórie ZB boli predstavené v kapitole 2.2, ako parametre diferenciálnych rovníc. Pre väčšiu kontrolovateľnosť výpočtu však boli zavedené ďalšie parametre:

- $tmax$ – maximálna hodnota t, po prekročení výpočet trajektórie skončí.
- nth – definuje, že každý koľkáty bod má byť zapísaný do pamäti pri výpočte jednej trajektórie. Nemá vplyv na výpočet rezov HP.
- $pMax$ – maximálny počet bodov uložených do pamäti, po prekročení výpočet trajektórie skončí.
- $uhMax$, $ihMax$ – definujú maximálne odskoky ZB oproti predchádzajúcemu bodu trajektórie, $uhMax$ na osiach u_1, u_2, $ihMax$ na osi i. Pri prekročení je integračný krok znížený na polovicu, a výpočet sa zopakuje. Ich ekvivalentmi v bezrozmernom systéme rovníc sú $xyhMax$, $zhMax$, kde $xyhMax$ udáva maximálny odskok na osiach x a y, kým $zhMax$ na osi z.
- n – s týmto číslom je integračný krok vynásobený po každom úspešnom výpočte ZB.
- $h0$ – začiatočná veľkosť integračného kroku.
- $ttest$ – čas, po ktorom sú podmienky typu *Chaos* testované. Je použitý iba pri výpočte rezov HP.

Keďže všetky výpočty vychádzajú z výpočtu trajektórie ZB, jedným z najdôležitejších častí aplikácie je práve kód, ktorý vykonáva tento výpočet. Pri návrhu programového kódu pre výpočet trajektórií bol teda dôležitý v prvom rade jeho čitateľnosť a overiteľnosť, ale aj efektivita. Vychádzalo sa z matematických podkladov

uvedených v kapitole 2, konkrétne z rovníc (2.1) a (2.10). V prílohe B uvádzame okomentovaný úryvok kódu z triedy `PCTrajectoryCalculator` aplikácie *Chuaviz* pre výpočet jednej trajektórie.

Chaotické systémy sú veľmi citlivé aj na malé zmeny, preto dostatočná presnosť výpočtu je veľmi dôležitá. Pri implementácii výpočtu trajektórie sa uvažovalo medzi typmi *float*, *double* a *long double*, ktoré jazyk C++ ponúka. Kým *float* poskytuje presnosť na 6–9 miestnu decimálnu presnosť, *double* 15–17 a *long double* 18–21 [40]. Skúšobne boli vypočítané trajektórie s rovnakými parametrami s použitím rôznych typov. Os i z výsledkov bol vynesený na graf uvedený v Obr. 4 – 3.



Obr. 4 – 3: Porovnanie presnosti *float*, *double* a *long double* pri simulácii trajektórie pre hodnoty i.

Z grafu môžeme vidieť, že kým pri použití typu *float* sú výsledky rozličné už od 0 s, výsledky s *double* a *long double* začnú divergovať až okolo 350-ej sekundy simulácie. Pri testovaní bola meraná aj rýchlosť výpočtu. Zistilo sa, že výpočet s typom *float* a *double* trvá zhruba rovnakú dobu, čo potvrdzujú aj merania uskutočnené v práci [3]. Pri použití *long double* sme však spozorovali dvojnásobné spomalenie.

Vzhľadom na jeho rýchlosť, a relatívnu presnosť spozorovateľnú aj z Obr. 4 – 3, bolo rozhodnuté použiť typ *double* vo všetkých výpočtoch.

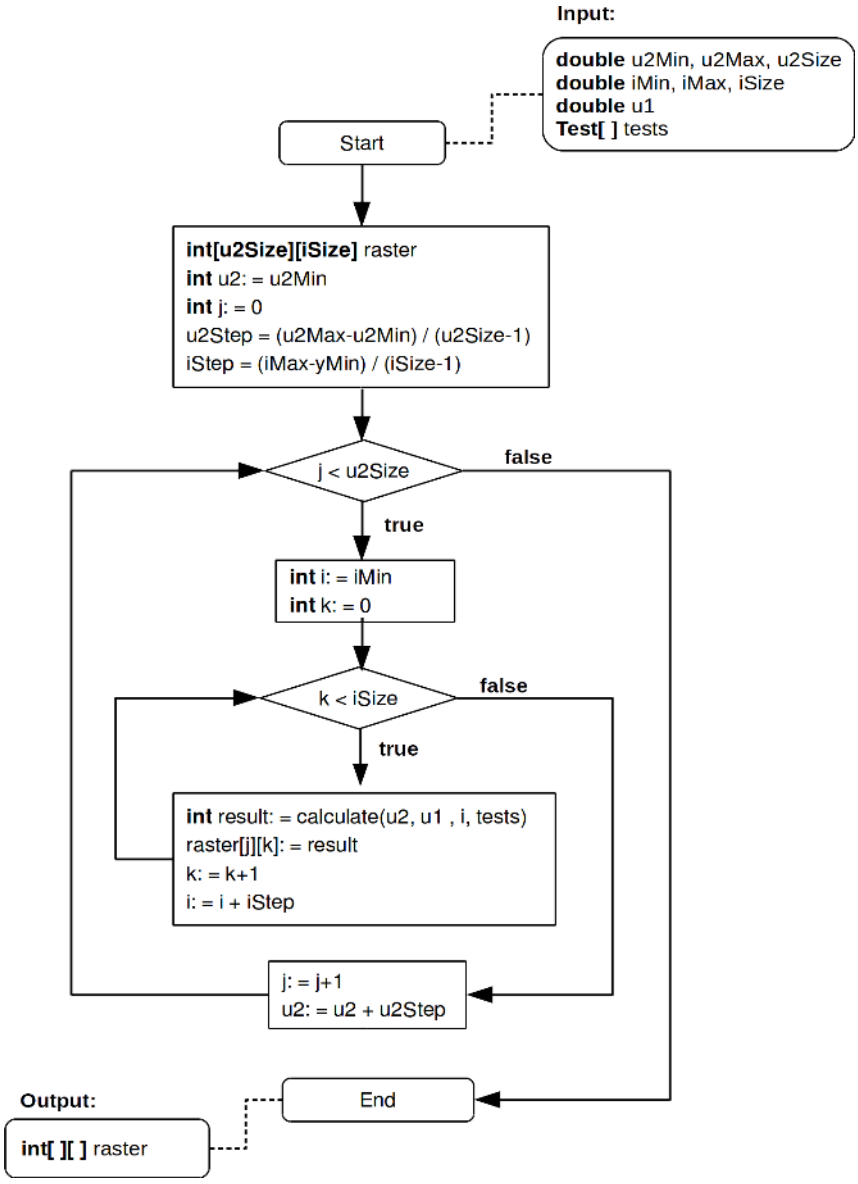
4.6.2 Výpočet rezov HP

Algoritmus výpočtu rezov HP v rovine i, u_2 je v znázornený na diagrame v Obr. 4 – 4. Vstupné parametre výpočtu sú hodnoty:

- `u2Min`, `u2Max` – začiatok a koniec intervalu začiatočných podmienok na osi u_1. Musí byť splnená podmienka: `u2Min` $<$ `u2Max`
- `u2Size` – počet začiatočných bodov (rozlíšenie) na osi u_2.
- `iMin`, `iMax` – začiatok a koniec intervalu začiatočných podmienok na osi i. Musí byť splnená podmienka: `iMin` $<$ `iMax`
- `iSize` – počet začiatočných bodov (rozlíšenie) na osi i.
- `u1` – začiatočná podmienka u_1, ktorá je konštantnou počas celého výpočtu jedného rezu.
- `tests` – množina testovacích podmienok, podľa ktorých sú určené typy trajektórie, reprezentované ako pole.

Ako to je zrejmé aj z Obr. 4 – 4, výpočet rezu sa skladá z dvoch vnorených cyklov, pričom telo vonkajšieho cyklu je vykonané `u2Size` krát, a telo vnútorného cyklu `iSize` krát pre každú iteráciu vonkajšieho cyklu. Pred začiatkom samotného výpočtu sú určené hodnoty `u2Step` a `iStep`, ktoré reprezentujú vzdialenosti medzi začiatočnými bodmi nachádzajúcimi sa po sebe na jednotlivých osiach. Pri každej iterácii vonkajšieho, resp. vnútorného cyklu zvyšujeme hodnoty premenných `u2` o `u2Step`, resp. `i` o `iStep`, teda prechádzame každým bodom mriežky začiatočných hodnôt. V tele vnútorného cyklu sa nachádza funkcia `calculate(u2,`

`u1, i, tests)`, ktorá podľa množiny testovacích podmienok `tests` zistí a vráti číselný identifikátor atraktoru trajektórie začínajúceho z bodu `[u2, u1, i]`.



Obr. 4 – 4: Algoritmus sekvenčného výpočtu rezu HP v rovine u_2, i [31].

Ako to môžeme vidieť aj z Obr. 4 – 4, funkcia `calculate` je vykonaná `u2Size` × `iSize` krát, čiže pre každý bod mriežky začiatočných hodnôt. Komplexita výpočtu teda pri rovnomernom zvyšovaní rozlíšenia na osiach u_2 a i rastie kvadraticky. Preto výpočty rezov HP pri vyšších rozlíšeniach môžu byť časovo veľmi náročné, napr. pre

rez veľkosti 440×440 bodov je potrebné vypočítať trajektóriu a určiť jeho atraktor pre 193 600 rôznych ZP. Ak za priemerný čas výpočtu jednej trajektórie predpokladáme iba 50 ms, výpočet jedného rezu by trval 2 hodiny a 40 minút. Ak by sme však chceli vypočítať HP v 3D priestore, v rozlíšení 440×440×440, potrebovali by sme 440 rezov, čoho vypočítanie by trvalo 48 dní, 21 hodín a 20 minút. Môžeme z toho usúdiť, že tento výpočet je časovo veľmi náročný, a jeho optimalizácia je nevyhnutná.

4.6.3 Paralelizácia výpočtov

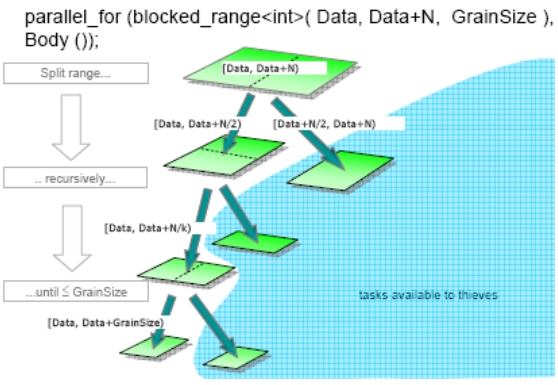
V súčasnosti procesory obsahujú viacero jadier a tak je zaujímavá paralelizácia časovo náročných výpočtov. Ako bolo uvedené aj na začiatku predchádzajúcej kapitoly, aplikácia má poskytovať tri druhy výpočtov.

Princíp výpočtu trajektórie bol uvedený v kapitole 2.4.2. Vidíme, že rovnica výpočtu je zapísaná v rekurzívnom tvare, čiže pre výpočet n-tého bodu trajektórie musíme poznať súradnice predchádzajúceho bodu. Jednotlivé iterácie výpočtu tvoria teda reťaz závislostí, ktorej začiatok je v nultom bode a koniec v poslednom. Paralelné počítanie jednej trajektórie preto nie je uskutočniteľné.

Ďalším výpočtom je výpočet rezov HP. V tomto prípade mriežku ZP presne poznáme v momente spustenia výpočtu. Pre každú jednu ZP potrebujeme zistiť atraktor, k čomu musíme simulovať trajektóriu od daných ZP. Na diagrame ilustrujúceho proces výpočtu HP v Obr. 4 – 4, je zistenie atraktora trajektórie označený funkciou `calculate`. Môžeme z neho vidieť, že každé volanie funkcie je závislé iba na parametroch obvodu, množiny testovacích podmienok a začiatočných hodnotách u_2, i a u_1. Keďže parametre obvodu, testovacie podmienky a hodnota u_1 sú konštantné počas celého výpočtu jedného rezu, a ZP sú vopred známe, všetky volania tejto funkcie je možné začať vykonávať už v momente spustenia výpočtu, bez nutnosti čakania na výsledky ostatných volaní funkcie. Vyplýva z toho teda, že paralelizácia

je využiteľná pre výpočet HP. Pre paralelizáciu bola zvolená knižnica *Intel TBB* predstavená v kapitole 4.3. Tá poskytuje niekoľko všeobecných paralelných algoritmov. Poskytuje aj algoritmus *parallel_for*, ktorý bol vybraný za účelom paralelného počítania rezov HP.

Konštrukcia *parallel_for* slúži na paralelné vykonanie série výpočtov, pričom jednotlivé výpočty sú závislé na jednej celočíselnej hodnote – iterátore – ktorá je zvyšovaná v rovnomerných krokoch. Je to teda obmedzená verzia štandardného `for` cyklu, v ktorom všetky iterácie môžu byť vykonané súbežne [19]. Pri implementácii tejto konštrukcie je potrebné všeobecne zapísať, ako počítať `for` cyklus pre ľubovoľný interval. Následne zadáme celý interval na ktorom chceme počítať. *Intel TBB* automaticky rozdelí zadaný interval a úlohy prerozdelí medzi vláknami, algoritmus rozdelenia intervalu je znázornený na obrázku 4 – 5.



Obr. 4 – 5: Algoritmus delenia intervalu v *parallel_for* [10].

Na základe poznatkov o počte paralelných úloh a efektivite ich rozdelenia [19], [7], [9] bolo rozhodnuté, že paralelizovať sa bude vonkajší `for` cyklus, teda jednou úlohou bude vypočítanie jedného stĺpca rezu. Znamená to, že pri počítaní rezu veľkosti M×N, kde M rovná sa počtu stĺpcov a N rovná sa počtu riadkov, bude vytvorených M úloh. V najhoršom prípade – ak pred ukončením výpočtu ostane práve jedna úloha, ktorú nie je možné ďalej deliť – dôjde k vypočítaniu tejto úlohy bez využitia paralelizácie. Znamená to, že v 1/n časti celého výpočtu bude využité

iba jedno procesorové jadro. V praxi však veľkosť rezov sa určuje v stovkách pixelov, napr. pri reze veľkosti 440×440 možné spomalenie v 1/440 časti výpočtu predstavuje akceptovateľné riziko.

Predvolená granularita výpočtu pri *parallel_for* je 1. Znamená to, že každá iterácia cyklu je považovaná za osobitnú úlohu. Podľa [19] a [7], vytvorenie čím menšieho počtu úloh znamená nižšie režijné náklady, avšak podľa experimentov v [9], príliš nízky počet zabráni efektívnemu rozdeleniu úloh a využitiu mechanizmu *task stealing*. Počet úloh by teda mal byť niekoľkonásobne vyšší, ako počet jadier pre efektívne uplatnenie *task stealingu*, avšak jednotlivé úlohy by mali byť dostatočne dlhé, aby pomer režijných nákladov k reálnym výpočtom bol minimálny.

Do aplikácie bola pridaná možnosť voliteľného použitia paralelizmu, aby bolo možné porovnať jeho rýchlosť so sekvenčným výpočtom.

4.6.4 3D vizualizácie

Pri návrhu 3D vizualizácie, ako základ bola použitá knižnica *QPlot3D* predstavená v kapitole 4.5.

Zistilo, že knižnica neposkytuje riešenie pre všetky požiadavky na 3D zobrazenie, ktoré vznikli pri vývoji. Knižnica *QPlot3D* je distribuovaná pod licenciou LGPL, ktorá povoľuje jej modifikáciu a redistribúciu. Preto jej funkcionalita bola rozšírená pre potreby aplikácie *Chuaviz*. Pridali sa nasledujúce vylepšenia:

- *3D hranol* – nová funkcia `Draw3DBox`, ktorá umožňuje vykresliť farebný 3D hranol. Slúži na zobrazenie testovacích hranolov.
- *3D guľa* – nová funkcia `Draw3DSphere`, pomocou ktorej je možné do priestoru pridať farebnú guľu, ktorá slúži pre naznačenie začiatku a konca trajektórie.
- *Pestrofarebná krivka* – nový atribút `colorful` krivky, ktorý ak je povolený,

krivka sa vykreslí postupným prechodom 300 farebných odtieňov od jeho začiatku až ku koncu. Bola pridaná ako možnosť vykreslenia trajektórie.

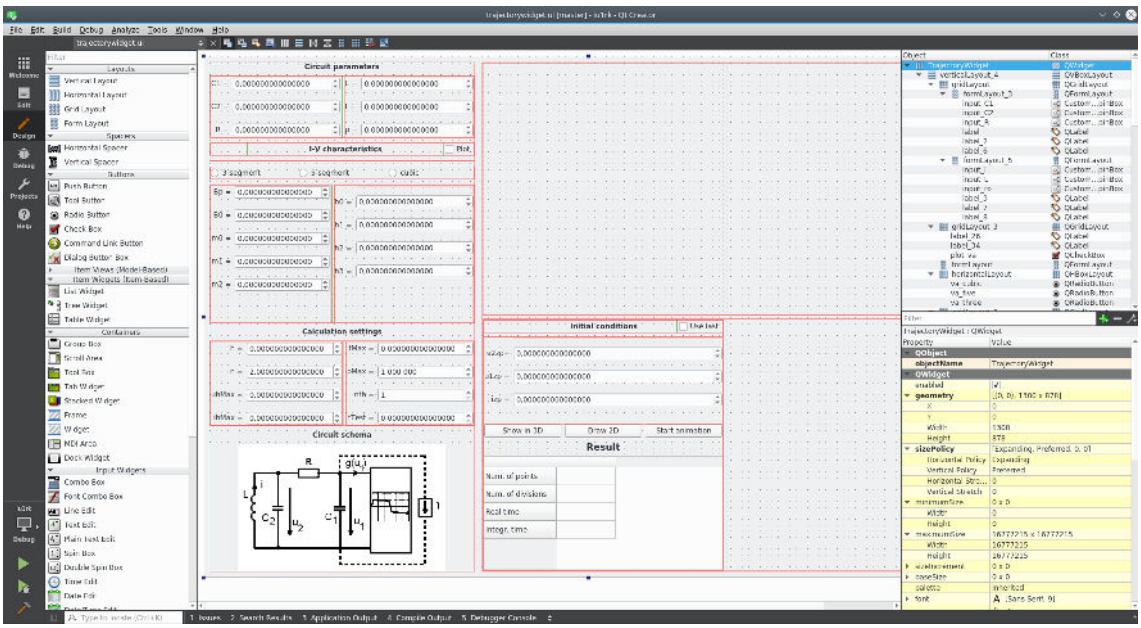
- *Farebný prechod* – nová funkcia `getColor(int i)` slúži na generovanie farebných prechodov, napr. pri pestrofarebnom zobrazení trajektórie.
- *Farebná škála* – rozšírenie legendy o nový prvok, ktorá zobrazí farebnú škálu použitú pri pestrofarebnej krivke.
- *Skrytie legendy* – je možné skryť legendu cez kontextové menu, čo pôvodná verzia knižnice neumožnila.

Na vykreslenie trajektórie v 3D je interne použité OpenGL, vďaka čomu výkon zobrazenia scény využíva hardvérové zrýchlenie dané grafickou kartou. Pre správne zobrazenie je potrebné mať grafickú kartu, ktorá podporuje OpenGL so správne nainštalovanými ovládačmi. Modifikovaná verzia knižnice *QPlot3D* je distribuovaná spolu s aplikáciou *Chuaviz* v priečinku `qplot3d`.

4.6.5 Návrh používateľského rozhrania

Pre návrh používateľského rozhrania aplikácie *Chuaviz* bolo použité integrované vývojové prostredie *Qt Creator*. Prostredie poskytuje tzv. design mód, v ktorom je možné používateľské rozhranie poskladať z widgetov. Okrem iných nastavení, umožňuje meniť rozloženie okna, priraďovať funkcie k interakciám s widgetmi, a upravovať ich atribúty. Tiež podporuje vývoj a prácu s vlastnými widgetmi.

Náhľad na hlavné okno aplikácie *Chuaviz* zobrazený v design móde programu *Qt Creator* je uvedený na Obr. 4 – 6. Podobným spôsobom boli navrhnuté ďalšie dve okná aplikácie, ktoré sú spolu s funkcionalitou prvkov hlavného okna predstavené v nasledujúcej kapitole.



Obr. 4 – 6: Hlavné okno aplikácie *Chuaviz* v *Qt Creator*

4.7 Funkcionalita konečnej aplikácie

Používateľské rozhranie aplikácie *Chuaviz* pozostáva z jedného okna, ktoré poskytuje tri prepínateľné pohľady. V týchto podkapitolách predstavíme funkcionalitu jednotlivých pohľadov.

4.7.1 Výpočet trajektórie

Prvý pohľad, ktorý je zobrazený hneď po otvorení programu, je určený pre skúmanie jednej trajektórie ZB. Umožňuje zadanie parametrov obvodu, vykonanie výpočtu trajektórie a jej vizualizáciu. Vizualizácia je možná pomocou Mongeovej projekcie v 2D do troch rovín alebo pomocou perspektívnej 3D projekcie. Projekcie do 2D sú priamo umiestnené v okne, podporujú posúvanie, priblíženie a postupné vykresľovanie trajektórie v čase. 3D projekcia sa zobrazí v novom okne, tá okrem posúvania a priblíženia podporuje aj otáčanie a niekoľko ďalších možností prispôsobenia zobrazenia, ktoré budú predstavené nižšie.

Snímok obrazovky používateľského rozhrania pohľadu pre výpočet a vizualizáciu trajektórie je uvedený na Obr. 4 – 7. Jednotlivé časti rozhrania sú označené nasledujúcimi číslami:

1. *Parametre obvodu* – vstupné polia, do ktorých sa zadávajú parametre obvodu.

2. *VA charakteristika* – pomocou týchto vstupných polí je možné zadať parametre VA charakteristiky. Prepínacie políčko na vrchu tejto časti umožňuje vybrať typ VA charakteristiky, žiadaného pre výpočet. Keďže každý typ VA charakteristiky vyžaduje iné parametre, podľa výberu typu sa nižšie číselné vstupy aktivujú/deaktivujú.

3. *Vykreslenie VA charakteristiky* – zaškrtávacie tlačidlo slúži na vykreslenie grafu aktuálne zvolenej VA charakteristiky do 2D projekcie v rovine i, u_1 (12b. na Obr. 4 – 7). Pre vykreslenie grafu je potrebné najprv vykresliť trajektóriu.

4. *Parametre výpočtu* – v tejto časti sa zadávajú parametre na dolaďovanie výpočtu, ktoré sú opísané v kapitole 4.6.1.

5. *Schéma obvodu* – ilustruje schému simulovaného Chuaovho obvodu.

6. *Začiatočné podmienky* – do týchto polí sa zadávajú začiatočné hodnoty u_2, u_1 a i použité pre výpočet.

7. *Použiť posledné ZP* – zaškrtávacie tlačidlo, ktoré pri zaškrtnutí automaticky nahradí hodnoty začiatočných podmienok v 6) posledným bodom vypočítanej trajektórie. Táto funkcionalita umožňuje vynechať prechodný dej pri zobrazení atraktora, alebo počítanie trajektórie po častiach.

8. *Zobraziť v 3D* – tlačidlo, ktoré spustí výpočet podľa zadaných parametrov, a výslednú trajektóriu zobrazí v perspektívnom 3D pohľade umiestneného v novom okne. Informácie o prebehnutom výpočte sú zapísané do tabuľky v 11). Bližší popis ovládania 3D projekcie, ako aj ukážkový snímok je uvedený

na Obr. 4 – 8.

9. *Vykresliť v 2D* – spustí výpočet a výslednú trajektóriu vykreslí do oblastí 12a, 12b a 12c (Obr. 4 – 7) v 2D projekcii. Informácie o výpočte, podobne ako pri 8), sú zapísané do tabuľky v 11).

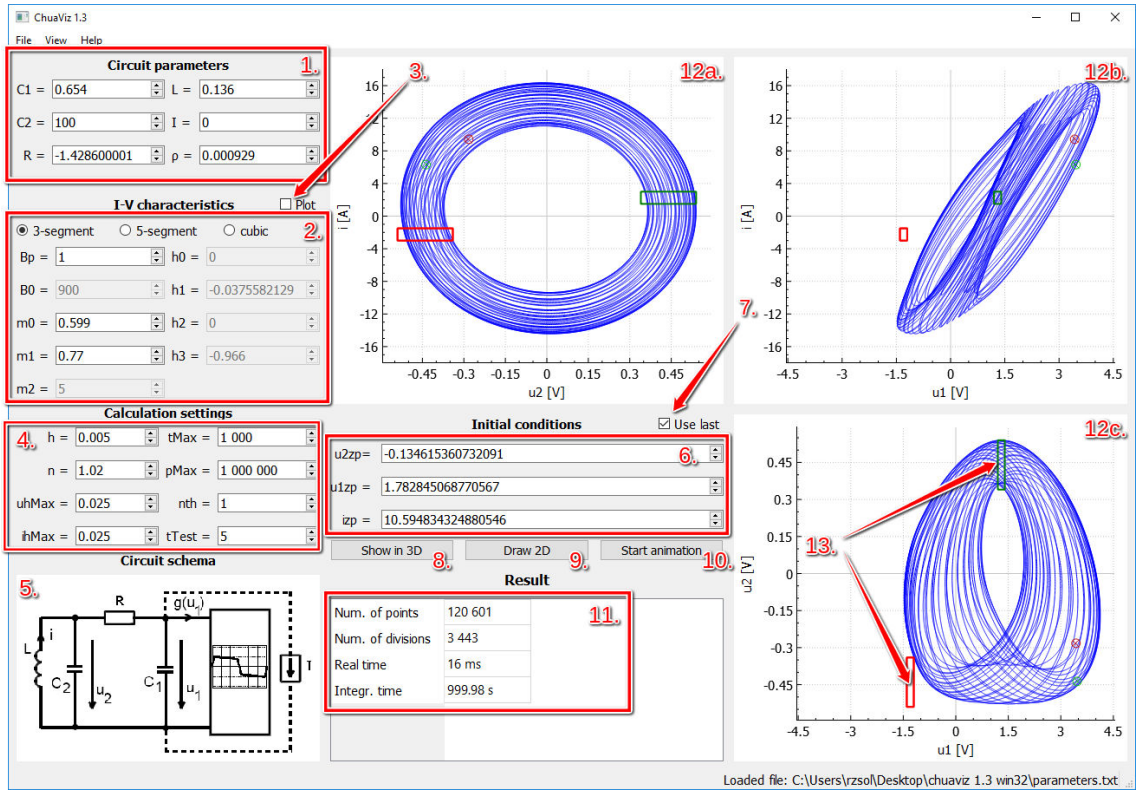
10. *Začať animáciu* – správa sa podobne, ako tlačidlo 9), vypočítanú trajektóriu avšak vykresľuje postupom času, v 10x zrýchlení vzhľadom k integračnému času. Po spustení animácie je text tohto tlačila zmenený na *„Stop animation“* (zastaviť animáciu) a po opätovnom kliknutí zastaví vykresľovanie.

11. *Informácie o výpočte* – do tejto tabuľky sú zapísané informácie o prebehnutom výpočte – počet vypočítaných bodov, počet delení, reálny čas spotrebovaný výpočtom a integračný čas v poslednom vypočítanom bode.

12. *Plochy pre 2D* – do týchto oblastí je vizualizovaná vypočítaná trajektória pomocou Mongeovej projekcie do troch rovín: 12a – i, u_2, 12b – i, u_1 a 12c – u_2, u_1. Začiatok trajektórie je označený malým zeleným krúžkom, kým koniec červeným. Zobrazenú oblasť je možné posúvať pomocou ťahania myšou a priblížiť skrolovacím kolieskom myši. Pravým klikom na jednu z plôch, sa vyvolá kontextové menu, ktoré umožňuje export trajektórie, resp. konkrétnej projekcie vo formátoch: CSV, TXT, PLY a PNG.

13. *Testovacie hranoly* – do oblastí 12a, 12b a 12c sa okrem trajektórie vykreslia aj všetky definované testovacie hranoly pre CHA, potrebné pre výpočet rezu HP. Testovacie hranoly sa definujú v pohľade pre výpočet rezov HP, ktorý je opísaný v kapitole 4.7.2.

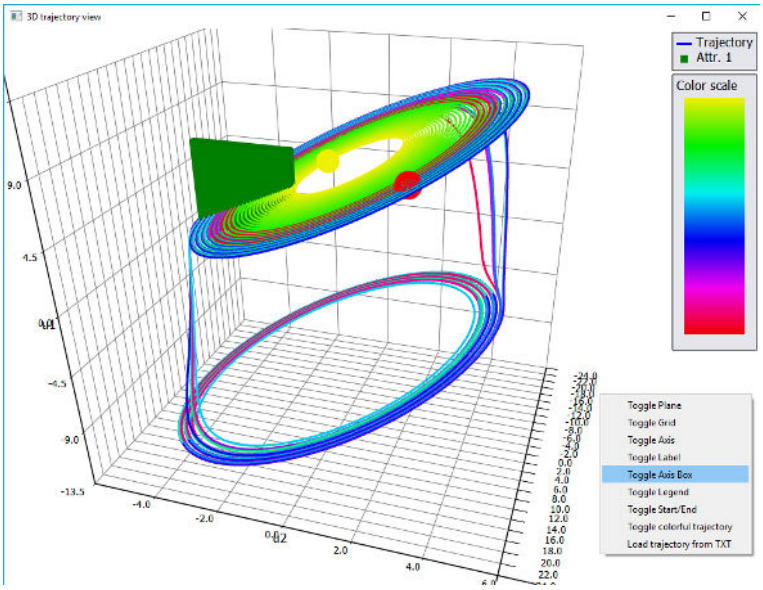
Ako bolo aj vyššie spomenuté, vypočítanú trajektóriu je možné vizualizovať aj v 3D kliknutím na tlačidlo *„Show in 3D“*. Po kliknutí sa otvorí nové okno v ktorom je zobrazená trajektória v 3D perspektívnom pohľade. Okrem trajektórie sa zobrazia aj všetky definované testovacie hranoly v ich príslušnej farbe, a voliteľne aj body



Obr. 4 – 7: Pohľad pre výpočet a vizualizáciu jednej trajektórie

začiatku/konca trajektórie, osi s popisom a legenda. Vykreslené objekty môžeme približovať, posúvať, ale aj otáčať. Snímka okna je uvedená na Obr. 4 – 8.

Okno 3D pohľadu na trajektóriu poskytuje niekoľko možností prispôsobenia zobrazenia. Po pravom kliknutí na ľubovoľné miesto v okne sa zobrazí menu, v ktorom si môžeme medzi možnosťami vyberať. Všetky položky v menu, okrem posledného sa správajú ako prepínacie tlačidlá – po prvom kliknutí zmenia stav a po druhom sa vrátia do pôvodného stavu. Pomocou položiek *„Toggle Plane“*, *„Toggle Grid“* a *„Toggle Axis Box“* je možné zvýrazniť plochy, ktoré tvoria hranice vykreslenia 3D objektov. Položka *„Toggle Axis“* zobrazí osi s číslovaním, kým *„Toggle Label“* názvy osí. Legendu, ktorá sa nachádza na pravej strane okna (Obr. 4 – 8) je možné prepínať pomocou *„Toggle Legend“*. Pomocou *„Toggle Start/End“* môžeme zapnúť zvýraznenie prvého a posledného bodu trajektórie vo forme farebných guľ. *„Toggle Colorful*



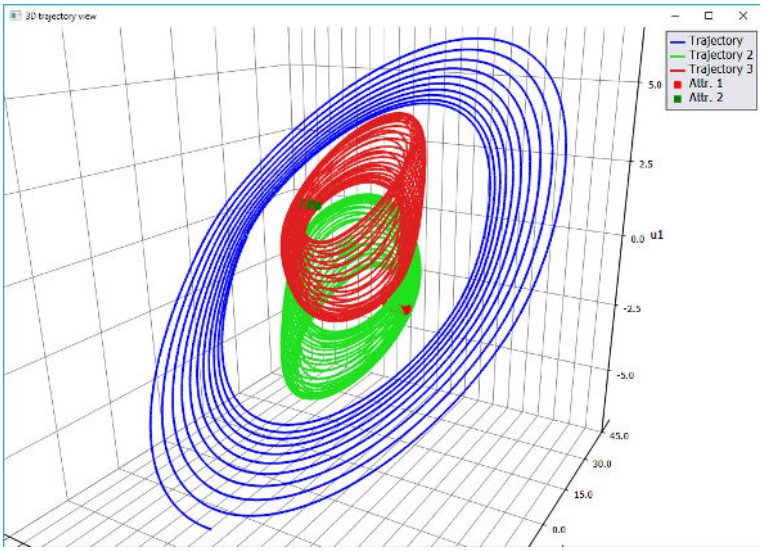
Obr. 4 – 8: 3D zobrazenie jednej trajektórie

trajectory“ umožňuje prepnúť medzi jednofarebným a pestrofarebným zobrazením trajektórie, pri pestrofarebnom zobrazení je zobrazená aj farebná škála v legende, podľa ktorého sú priradené farby jednotlivým úsekom vykreslenej trajektórie.

Posledná položka v menu *„Load trajectory from TXT“* umožňuje pridanie ďalších trajektórií do 3D okna. Ďalšia trajektória sa načíta z textového súboru, ktorý musí obsahovať údaje v rovnakom formáte, ako je formát použitý pri exporte trajektórie (formát je popísaný v používateľskej príručke). Po kliknutí na položku je zobrazené dialógové okno, v ktorom sa vyberá vstupný súbor. Ak je vypnutá možnosť pestrofarebného zobrazenia, trajektórie sú zobrazené odlišnými farbami, v opačnom prípade sú všetky pestrofarebné. Aplikácia umožňuje načítať až 6 ďalších trajektórií. Ukážka 3D zobrazenia viacerých trajektórií v aplikácii *Chuaviz* je viditeľná na Obr. 4 – 9.

4.7.2 Výpočet rezu HP

Druhý pohľad, znázornený na obrázku 4 – 10 umožňuje výpočet a vizualizáciu rezov HP. V ľavej hornej časti môžeme vidieť parametre obvodu, s ktorými budeme po-

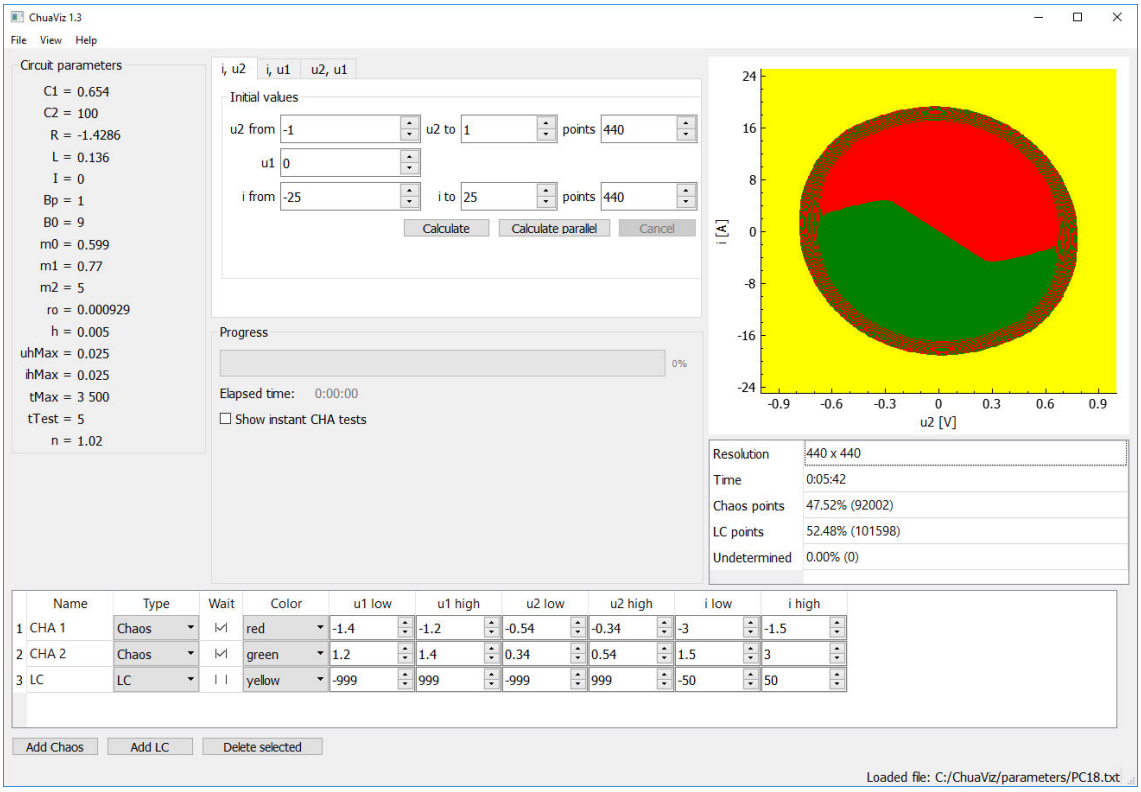


Obr. 4 – 9: 3D zobrazenie viacerých trajektórií

čítať. Parametre sa nastavujú v predošlom pohľade pre výpočet jednej trajektórie. V strednej hornej časti okna je možné nastaviť sieť bodov – raster. Môžeme si vybrať z troch rovín: (u_2, i), (u_1, i) a (u_1, u_2), pričom pri každom sa nastavuje rozsah a požadovaný počet vypočítaných bodov (rozlíšenie) na dvoch osí. Tretia súradnica počas výpočtu jedného rezu ostáva konštantná.

V ľavej dolnej časti sa nachádza tabuľka, v ktorom je možné definovať testovacie hranoly určené pre zistenie typu trajektórie. V tabuľke je možné zadať názov, typ podmienky (Chaos alebo LC), farbu, ktorá bude zapísaná do výsledného rezu pri kladnom vyhodnotení danej podmienky a mierku na jednotlivých osiach. Parameter *„Wait“* v tabuľke určuje, či sa má podmienka vyhodnocovať až po uplynutí integračného času t_{test} alebo hneď od nultej sekundy simulácie. Na testovacie hranoly, resp. podmienky s vypnutým parametrom *„Wait“* sa ďalej budeme odvolávať, ako rýchle hranoly. Každá podmienka je vyhodnotená (ak je rýchla alebo uplynul čas t_{test}) pre každý vypočítaný bod trajektórie, a pri prvom kladnom vyhodnotení je výpočet trajektórie zastavený a poradové číslo testu uložený do výsledku.

V závislosti od typu podmienky sú testovacie hranoly vyhodnotené rozdielnymi



Obr. 4 – 10: Pohľad pre počítanie rezu hraničnou plochou

spôsobmi. Testovacia podmienka typu *LC* je vyhodnotená, ako kladná, ak sa nájde bod na trajektórii, ktorý na niektorej osi presahuje hodnoty *low* a *high*. Test typu *Chaos* je kladný v opačnom prípade: ak bod zapadá do daných intervalov na všetkých osiach. Do tabuľky je možné pridať novú podmienku kliknutím na tlačidlo *„Add Chaos“*, resp. *„Add LC“*. Po kliknutí sa objaví nový riadok daného typu v tabuľke, v ktorom sa priamo nastavujú hodnoty podmienky. Vymazať podmienku je možné označením riadku (kliknutím na poradové číslo riadku) a stlačením tlačidla *„Delete selected“*.

Výpočet je možné spustiť kliknutím na tlačidlo *„Calculate“*, resp. *„Calculate parallel“*, v závislosti od toho, či si prajeme využiť paralelizáciu pri výpočte. Po spustení výpočtu sa v strede okna aktivuje ukazateľ postupu, ktorý informuje o počte vypočítaných stĺpcov v percentách. Algoritmus výpočtu je uvedený na Obr. 4 – 4.

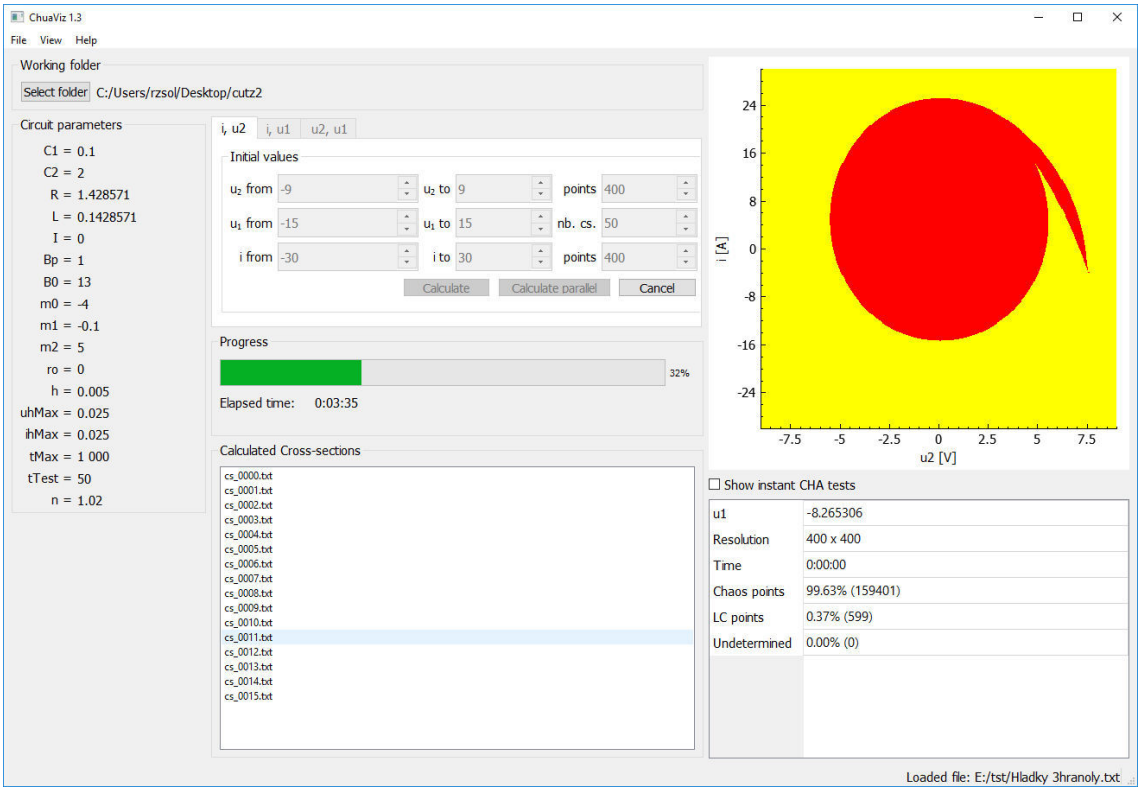
V pravej časti okna sú zobrazené výsledky výpočtu – vyobrazený rez HP a tabuľka s údajmi o výpočte. Príslušný rez HP je vykreslený priebežne, kým tabuľka až po skončení výpočtu. Do rezu HP je možné vykresliť kontúry rýchlych testovacích hranolov typu *„Chaos“* zaškrtnutím políčka *„Show instant CHA tests“*, ktoré sa nachádza pod ukazateľom postupu. Tabuľka obsahuje nasledujúce údaje: rozlíšenie výsledného rezu, celkový čas výpočtu, počet bodov pritiahnutých atraktorom CHA, počet bodov pritiahnutých atraktorom SLC a počet bodov, v ktorých sa nedospelo k výsledku ani po uplynutí času $t_{\max}$. Po skončení výpočtu pravým kliknutím na vypočítaný rez je možné vyvolať kontextové menu, ktoré povoľuje nasledujúce operácie: exportovať rez v TXT alebo CSV formáte, exportovať ZP nezistených trajektórií a uložiť rez ako PNG obrázok.

4.7.3 Výpočet série rezov HP

Tretí pohľad, slúži na výpočet postupnej série rezov HP. Keďže pri tomto výpočte sa jedná o dlhší výpočet a väčšie množstvo údajov, medzivýsledky (hotové rezy) pri výpočte sú zapísané priamo na úložisko. Preto ako prvý, musí byť vybraný pracovný adresár, do ktorého budú novovypočítané rezy zapísané, resp. z ktorého budú existujúce rezy načítané. Ak vo vybranom adresári sa už nachádzajú výsledky predchádzajúceho výpočtu, aplikácia ich rozpozná a umožní ich vizualizáciu. Keďže výpočet série rezov v závislosti ich rozlíšení a počtu môže tvrať viac dní, ale aj týždňov, aplikácia umožňuje otvoriť aj adresár s nedokončeným výpočtom a pokračovať v ňom aj po reštartovaní aplikácie, resp. počítača. Pri pokračovaní výpočet pokračuje ďalej od posledného kompletne vypočítaného rezu.

Obr. 4 – 11 uvádza pohľad výpočtu série rezov. V hornej časti okna je možné vybrať pracovný adresár (*„Working folder“*) klikom na tlačidlo *„Select folder“*. Kým nie je vybraný adresár, aplikácia nepovoľuje zadať parametre alebo spustiť výpočet v tomto pohľade. V ľavej časti sú uvedené parametre obvodu, ktoré boli nastavené

v pohľade pre výpočet jednej trajektórie. V strednej hornej časti okna je možné vybrať rovinu (i, u_2, i, u_1 alebo u_2, u_1), rozsah, rozlíšenie a počet rezov HP.



Obr. 4 – 11: Pohľad pre počítanie série rezov HP

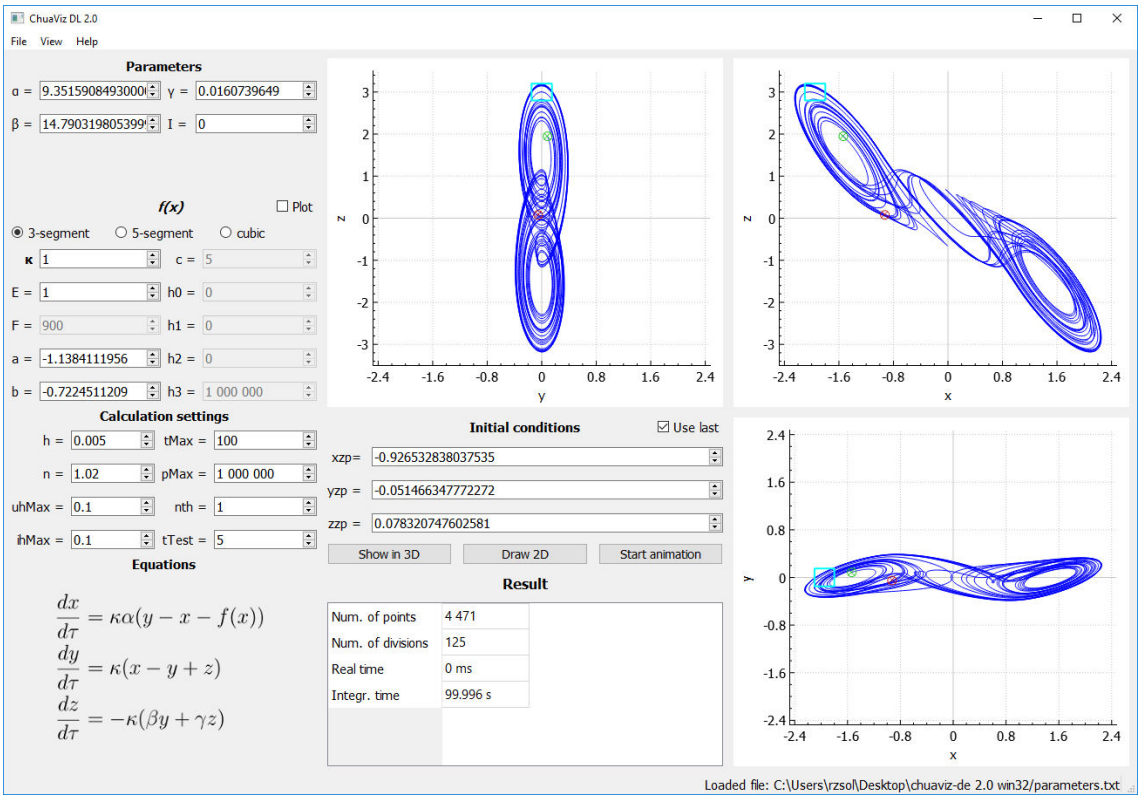
Oproti predchádzajúcemu pohľadu s výpočtom jedného rezu HP, tu sa nastavuje rozsah pri všetkých troch osiach. Výpočet je možné spustiť kliknutím na tlačidlo *„Calculate“* alebo *„Calculate Parallel“*. Po spustení výpočtu sa v pracovnom adresári vytvorí súbor s názvom *info.txt*, do ktorého sú uložené rozsahy výpočtu. Následne už nie je možné meniť rozsahy, iba po vybraní nového adresára. Po dopočítaní každého rezu je súbor s rezom uložený na úložisko, do pracovného adresára. Taktiež sa pridá do zoznamu *„Calculated cross-sections“*, ktorý sa nachádza v strednej časti okna. Po kliknutí na ľubovoľný rez HP v zozname sa daný rez načíta z úložiska a zobrazí sa na ploche v pravej časti pohľadu.

4.8 Modifikácia aplikácie pre bezrozmerný systém rovníc

Chuaov obvod v publikáciach je často skúmaný pomocou bezrozmerného systému diferenciálnych rovníc (2.2). Aby bolo možné skúmať aj tieto obvody, bola vytvorená upravená verzia aplikácie *Chuaviz*, s názvom *Chuaviz DE* (dimensionless equations). Hlavné zmeny modifikovanej aplikácie reprezentujú:

- Použitie bezrozmerných diferenciálnych rovníc (2.2) pre výpočet trajektórie;
- Výpočet pomocou bezrozmerných charakteristík diódy (2.4), (2.6) a (2.8);
- Prispôsobené používateľské rozhranie novým parametrom a všeobecnej notácii osí x, y, z.

Hlavné okno s pohľadom na výpočet trajektórie je znázornené na Obr. 4 – 12. Na obrázku je možné vidieť zmenené parametre obvodu (*Parameters*) a Chuaovej diódy ($f(x)$). Osi sú označené všeobecným x, y, z namiesto u_1, u_2, i pri zadávaní začiatočných podmienok a projekciách trajektórie, ale aj vo všetkých ostatných častiach programu. Na uvedenej snímke obrazovky je vykreslený atraktor DE1 z publikácie [5].



Obr. 4 – 12: Hlavné okno aplikácie *Chuaviz DE*

5 Optimalizácia rýchlosti výpočtov

V predchádzajúcej kapitole sme uviedli aplikáciu *Chuaviz*, resp. *Chuaviz DE*, jej funkcionalitu, a spôsob vykonania výpočtov. Môžeme z nej vidieť, že implementované výpočty trajektórií, rezov HP a sérií rezov HP sú výkonovo náročné operácie. V tejto kapitole sú uvedené možnosti urýchlenia výpočtov realizovaných pomocou predstavených aplikácií. Poukážeme na to, že zvolenie vhodných parametrov výpočtu je kritické z hľadiska výkonu, uvedieme možnosti optimalizácie dané prekladačom a tiež zrýchlenia dosiahnuté pomocou paralelizácie.

Uvádzame aj výsledky výkonnostných testov, ktoré boli vykonané na počítačových zostavách s rôznymi parametrami. Zoznam počítačových zostáv použitých pri testovaní spolu s ich parametrami je uvedený v Tab. 5 – 1. V ďalších častiach kapitoly sa budeme na nich odvolávať podľa ich názvu (1. stĺpec tabuľky).

5.1 Zmena parametrov výpočtu

Výber vhodných parametrov pri výpočte je kľúčový nielen pre dosiahnutie správneho výsledku, ale aj z hľadiska spotrebovaného času. Na nasledujúcich stranách uvádzame experimenty s variáciou hodnôt parametrov n a $hMax$, rôznym umiestnením testovacích hranolov pre detekciu atraktorov, ako aj výpočty HP s viacerými testovacími hranolmi.

5.1.1 Parametre *n* a *hMax*

Parametrom n sa násobí integračný krok h v metóde Runge-Kutta (kapitola 2.4.2) dovtedy, kým nie je prekročená hodnota $hMax$ definovaná na danej osi. V aplikácii *Chuaviz* používame parametre $uhMax$ pre osi u_1, u_2 a $ihMax$ pre os i, kým v aplikácii *Chuaviz DE* $xyhMax$ pre osi x, y a $zhMax$ pre os z. Týmito parametrami

Tabuľka 5 – 1: Počítačové zostavy použité pre testy

Názov	Model procesora	Op. systém	jadrá / vlákna
PC1	Intel i5-5200U, 2.2 GHz	Win 8.1, 64 bit	2 / 4
PC1L	Intel i5-5200U, 2.2 GHz	Linux Mint 18, 64 bit	2 / 4
PC2	Intel Q9550, 2.83 GHz	Win 7, 64 bit	4 / 4
PC3	Intel i7-2670QM, 2.2 GHz	Win 7, 64 bit	4 / 8
PC4	Intel E5200 2.5 GHz	Win 7, 32 bit	2 / 2
PC5	AMD Athlon II 215, 2.7 GHz	Win 7, 32 bit	2 / 2
PC6	Intel E6600, 2.4 GHz	Win 7, 32 bit	2 / 2
PC7	Intel i7-3770, 3.4 GHz	Win 7, 64 bit	4 / 8
PC8	Intel G3250T, 2.8GHz	Win 10, 64 bit	2 / 2
PC8L	Intel G3250T, 2.8GHz	Linux Mint 18, 64 bit	2 / 2
PC9	Intel i7-4790K, 4 GHz	Linux Mint 18, 64 bit	4 / 8
PC9V	Intel i7-4790K, 4 GHz	Win 10, 64 bit[1]	4 / 8
PC10	Intel E5300, 2.6 GHz	Win 7, 32 bit	2 / 2
PC11	Intel i7-7700K, 4.2 GHz	Win 10, 64 bit	4 / 8

sa testuje vzdialenosť vypočítaného bodu od predchádzajúceho. Podstatnú úlohu zohráva aj ich veľkosť, keďže presnosť výpočtu do istej miery určujú práve tieto parametre. Preto boli urobené testy pre rôzne variácie týchto hodnôt [31].

Za účelom testovania bol vykonaný výpočet 1 trajektórie v programe *Chuaviz*, preloženého s optimalizačnou úrovňou prekladača `O2` (optimalizácie `Od` a `O1` boli pomalšie). Tab. 5 – 2 uvádza výsledky testovania, pričom boli použité štyri hodnoty parametra n uvedené v stĺpcoch tabuľky, a štyri variácie parametrov $uhMax$ a $ihMAx$:

a) $uhMax = 0{,}05\,\mathrm{V}$; $ihMax = 0{,}05\,\mathrm{A}$;

[1] Virtualizácia pod Linuxom, pomocou programu VirtualBox.

b) $uhMax = 0{,}05\,\mathrm{V}$; $ihMax = 0{,}01\,\mathrm{A}$;

c) $uhMax = 0{,}01\,\mathrm{V}$; $ihMax = 0{,}05\,\mathrm{A}$;

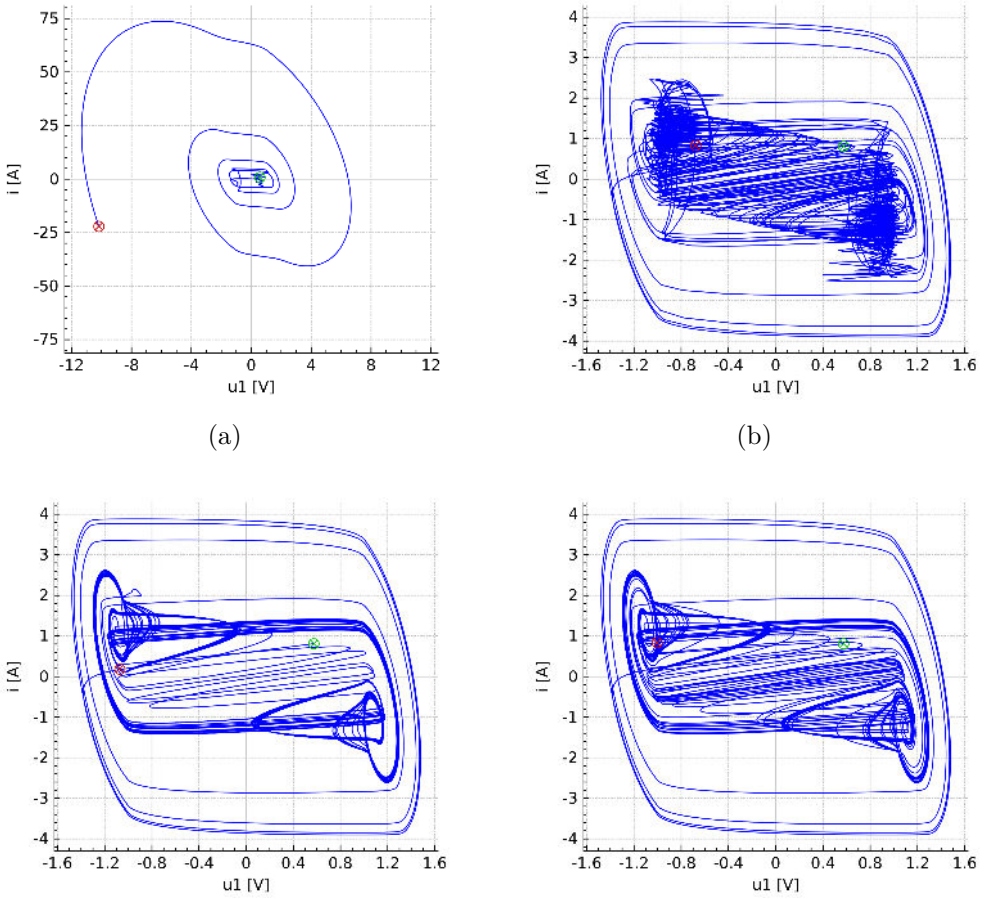
d) $uhMax = 0{,}01\,\mathrm{V}$; $ihMax = 0{,}01\,\mathrm{A}$.

Z Tab. 5 – 2 sú taktiež je zrejmé, že najrýchlejší výpočet je pre prípad a) a $n = 1{,}02$ resp. 1,002. Dôležité je avšak poznamenať, že vyššie hodnoty parametrov $uhMax$ a $ihMax$ nie sú vždy žiadané a pri ich nevhodnom zvolení môže dôjsť k výraznej nepresnosti výpočtu. Túto skutočnosť ilustruje Obr. 5 – 1, v ktorom je vykreslená trajektória pohybu ZB pri štyroch rôznych hodnotách $uhMax$ a $ihMax$.

Tabuľka 5 – 2: Časy výpočtov 1 trajektórie pre rôzne $ihMax$, $uhMax$ a n [31].

	n	**2**	**1.2**	**1.02**	**1.002**
a)	Počet bodov	266 561	265 660	261 619	285 382
	Počet delení	266 561	69 879	7 474	825
	Čas výpočtu [ms]	58.5	45.1	38.7	39.8
b)	Počet bodov	1 317 582	1 276 267	1 275 883	1 299 908
	Počet delení	1 317 584	335 706	36 455	3 751
	Čas výpočtu [ms]	294.2	216.7	188.2	187.8
c)	Počet bodov	1 365 083	1 327 870	1 327 870	1 327 300
	Počet delení	1 365 085	349 279	37 915	3 830
	Čas výpočtu [ms]	298	228.3	192.9	193.6
d)	Počet bodov	505 903	485 011	479 195	474 347
	Počet delení	505 903	127 576	13 692	1 369
	Čas výpočtu [ms]	121.2	88.5	79.6	79.3

Dĺžky výpočtov pri jednotlivých prípadoch ilustrovaných na Obr. 5 – 1 boli nasledovné: a) $< 1\,\mathrm{ms}$, b) 1 ms, c) 2 ms, d) 95 ms. Je zrejmé, že veľkosť hodnôt $uhMax$ a $ihMax$ značne ovplyvňuje výsledky, ale aj dĺžku výpočtu. Najvýraznejší je roz-



Obr. 5 – 1: Trajektória ZB obvodu PC191 pri rôznych hodnotách $uhMax$ a $ihMax$. a) $uhMax = 1\,\mathrm{V}$; $ihMax = 1\,\mathrm{V}$. b) $uhMax = 0{,}5\,\mathrm{V}$; $ihMax = 0{,}5\,\mathrm{V}$. c) $uhMax = 0{,}1\,\mathrm{V}$; $ihMax = 0{,}1\,\mathrm{V}$. d) $uhMax = 0{,}001\,\mathrm{V}$; $ihMax = 0{,}001\,\mathrm{V}$.

diel v prípade Obr. 5 – 1a, kde boli použité najvyššie hodnoty parametrov $uhMax$ a $ihMax$. Nepresnosť výpočtu v tomto prípade bola až taká veľká, že sa ZB dostal mimo RP CHA a trajektória nakoniec chybne skonvergovala do SLC. Na ďalších obrázkoch vidíme postupné zvýšenie detailnosti vypočítanej trajektórie.

Preskok ZB do iného RP má pri výpočte rezov HP vážny následok. Atraktor trajektórie môže byť chybne identifikovaný, čo vedie k nesprávnemu výsledku. Pri každom obvode (parametroch) je teda potrebné opatrne zvážiť veľkosť parametrov

$uhMax$ a $ihMax$, kým nízke hodnoty vedú k výraznému poklesu výkonu, vysoké zvyšujú riziko vzniku viesť k chybných výsledkov.

Pre výpočty trajektórií na Obr. 5 – 1 boli použité parametre obvodu PC191 z knihy [5]. Textové súbory obsahujúce parametre výpočtov sú priložené v CD prílohe G, v priečinku `parameters/chuaviz/test-hmax`. Súbory sú otvoriteľné v aplikácii *Chuaviz*.

5.1.2 Testovacie hranoly

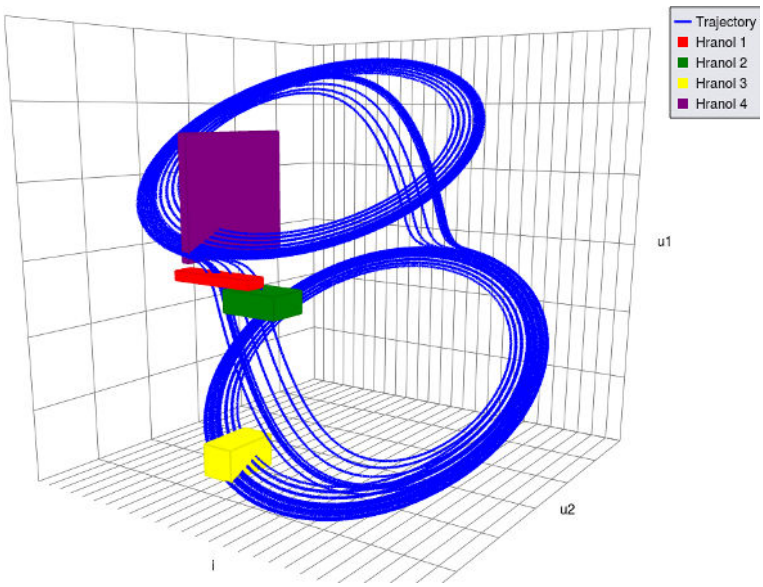
V ďalšom kroku sme počítali 1 rez HP v rozlíšení 200×200 bodov. Skúmali sme vplyv umiestnenia testovacieho hranola na rýchlosť detekcie CHA. Po vstupe ZB dovnútra hranola sa výpočet aktuálne počítanej trajektórie zastaví, a pokračuje sa ďalšou ZP. Boli zvolené 4 rôzne hranoly umiestnené v 3D.

Pre lepšiu predstavu, Obr. 5 – 2 ilustruje polohu všetkých štyroch testovacích hranolov, pričom hranol červenej farby bol označený číslom 1, zelený číslom 2, žltý 3 a fialový 4. Pri testovaní naraz bol použitý vždy iba jeden hranol.

Dĺžka výpočtu jedného rezu HP je zrejmá z Tab. 5 – 3. Môžeme z neho vidieť, že výpočet prebehol najrýchlejšie pri použití štvrtého hranola. Pri voľbe parametra n sa znovu potvrdzuje, že najrýchlejší výpočet je pre $n = 1{,}02$ a 1,002.

Tabuľka 5 – 3: Časy výpočtov 1 rezu HP (200 × 200 bodov) pre rôzne testovacie hranoly a hodnoty n [31].

$n =$	**2**	**1.2**	**1.02**	**1.002**
Hranol 1 [min]	1:14	0:49	0:42	0:44
Hranol 2 [min]	0:58	0:39	0:33	0:33
Hranol 3 [min]	1:06	0:44	0:36	0:38
Hranol 4 [min]	0:55	0:37	0:31	0:32



Obr. 5 – 2: Štyri testované hranoly zobrazené v 3D pomocou aplikácie *Chuaviz*.

Vstupné súbory s parametrami obsahujúce jednotlivé hranoly sú priložené v CD prílohe G, v priečinku `parameters/chuaviz/cuboids`. Súbory je možné otvoriť aplikáciou *Chuaviz*.

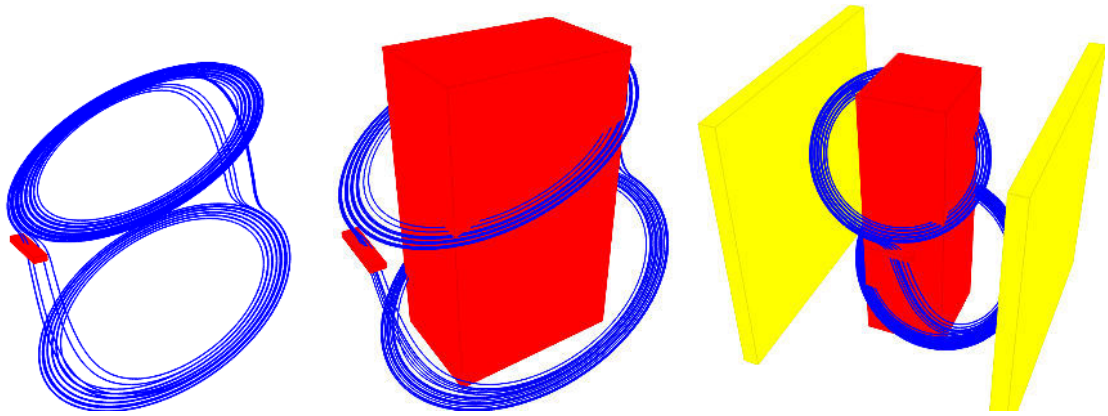
5.1.3 Umiestnenie viacerých hranolov

Ďalšie výkonnostné testy boli vykonané aplikáciou *Chuaviz*, pričom meraný bol čas výpočtu série rezov HP. Zámerom testovania bol, či je možné výpočet zrýchliť vhodným umiestnením ďalších testovacích hranolov v stavovom priestore.

Počítalo sa 50 rezov HP v rovine i, u_2 v rozlíšení 400×400 bodov. Pri výpočte boli použité tri sady testovacích hranolov, ktoré sú ilustrované na Obr. 5 – 3. V prvom prípade (A) – podobne, ako v predchádzajúcej kapitole – je pre detekciu CHA definovaný iba jeden „štandardný“ testovací hranol. V druhom prípade (B) je ku hranolu z A pridaný ďalší, „rýchly“ testovací hranol, ktorý je umiestnený v prostriedku CHA. V poslednom prípade (C) sú pridané ďalšie dve „rýchle“ hranoly mimo CHA, ktoré slúžia na detekciu limitného cyklu. Pri každom prípade bola prítomná

podmienka pre „nekonečno“ ($\pm 999\,\mathrm{V}, \pm 50\,\mathrm{A}$), ktorá bola identifikovaná ako LC.

„Rýchle“ hranoly boli zvolené po osobitnom vypočítaní prvého, posledného a stredného rezu série s podmienkou A. Predpokladalo sa, že prienik oblastí príťažlivosti ku CHA z týchto troch rezov bude rovnakou oblasťou príťažlivosti aj pri ostatných rezoch. Keďže pri výpočte s „rýchlymi“ testovacími podmienkami sa pri simulácii trajektórie nečaká na t_{test}, testovacej podmienke môže vyhovieť aj priamo začiatočná podmienka. V takomto prípade je určená farba bodu v reze pred začatím samotnej simulácie trajektórie.



(a) (b) (c)

Obr. 5 – 3: Poloha testovacích hranolov a CHA v 3D.

Výsledky testovania sú zhrnuté v Tab. 5 – 4. V prvom stĺpci tabuľky je uvedený názov počítačovej zostavy z Tab. 5 – 1, na ktorej bol test vykonaný. Testované boli 32 a 64 bitové verzie aplikácie, druhý stĺpec uvádza verziu použitú pre testy v danom riadku. Stĺpce A, B a C uvádzajú dĺžku výpočtu vo formáte `h:mm:ss` pri troch rôznych prípadoch testovacích hranolov a posledný stĺpec pomer rýchlostí vykonania výpočtov.

Z Tab. 5 – 4 vidíme, že v prípade C, pomocou troch pridaných „rýchlych“ testovacích hranolov bolo možné dosiahnuť až 14,58 násobné zrýchlenie výpočtu. Dôležité je však poznamenať, že tieto testovacie podmienky prinášajú isté riziko chybného

Tabuľka 5 – 4: Porovnanie časov výpočtu série 50 rezov HP rozlíšenia 400×400

Počítač	Verzia	A	B	C	Rýchlosť (A : B : C)
PC1L	64 bit	3:02:19	1:09:34	0:15:35	1 : 2,62 : 11,70
PC2	32 bit	4:14:34	1:36:21	0:20:32	1 : 2,64 : 12,40
PC2	64 bit	4:02:07	1:29:48	0:19:09	1 : 2,70 : 12,64
PC8	32 bit	6:39:43	2:22:03	0:28:15	1 : 2,81 : 14,15
PC8	64 bit	6:22:48	2:20:03	0:27:25	1 : 2,81 : 14,58
PC9	64 bit	1:44:58	0:39:05	0:07:53	1 : 2,68 : 13,31
PC10	32 bit	9:15:49	3:27:59	0:43:29	1 : 2,67 : 12,78
PC11	32 bit	0:58:21	0:22:13	0:05:16	1 : 2,62 : 11,08
PC11	64 bit	0:48:14	0:18:06	0:04:12	1 : 2,66 : 11,48

vypočítania rezov HP. Môže sa stať, že kvôli nevhodne umiestnenému testovaciemu hranolu isté trajektórie budú označené nesprávnym atraktorom alebo skryté atraktory obvodu ostanú neobjavené. Použitie ďalších testovacích hranolov preto vyžaduje dôkladné poznatky o morfológii HP obvodu.

Parametre vykonaných výpočtov vo formáte TXT sa nachádzajú v CD prílohe G v priečinku `parameters/chuaviz/cuboids-multi`. Súbory je možné načítať v aplikácii *Chuaviz*.

5.2 Optimalizačné úrovne prekladača

Ďalšia možnosť, ako zvýšiť rýchlosť výpočtu je optimalizácia prekladačom. Ako už bolo spomenuté v kapitole 4.6, na preklad programu bol použitý GCC (systém Linux) a Visual Studio od Microsoftu (systém Windows). Obidva prekladače poskytujú veľké množstvo možností optimalizácie. Ich výber je však zjednodušený pomocou optimalizačných úrovní, ktoré môžu byť chápané ako množiny optimalizačných nasta-

vení. V tejto časti práce skúmame optimalizačné úrovne prekladačov, ako možnosť zrýchlenia výpočtov vyvinutých aplikácií.

GCC ponúka nasledujúce úrovne optimalizácie [12]:

- O0 – žiadna optimalizácia (predvolené).
- O1 – mierna optimalizácia, ktorá nezvyšuje výrazne čas prekladu.
- O2 – plná optimalizácia, ktorá zvýši výkon programu na úkor času prekladu.
- O3 – plná optimalizácia (ako v O2), s povolením agresívnejších optimalizačných techník, ako inlinovanie funkcií (náhrada volania funkcie s jej telom) alebo vektorizáciou cyklov. Výstupný súbor pri O3 je zvyčajne väčší ako pri ostatných úrovniach, čo môže pôsobiť za istých okolností aj spomalenie.
- Os – optimalizuje veľkosť výsledného programu.

Visual Studio umožňuje aplikovať nasledujúce optimalizačné úrovne [26]:

- Od – žiadna optimalizácia.
- O1 – optimalizuje program pre čo najmenší výsledný kód.
- O2 – optimalizuje pre najvyšší výkon.

Rýchlosť výpočtu rezu HP v aplikácii *Chuaviz* bol testovaný na desiatich počítačových zostavách, pričom aplikácia bola preložená na 32 a 64 bitovú architektúru s troma rôznymi optimalizačnými úrovňami: Od (O0), O1 a O2. Výsledné časy sú uvedené v Tab. 5 – 5. Pri všetkých prípadoch bol počítaný jeden rez HP v rozlíšení 200×200, na nečinnom počítači. Každý výpočet bol zopakovaný aspoň trikrát, a do tabuľky bol zapísaný čas najkratšieho výpočtu. Posledný stĺpec pre porovnanie uvádza čas rovnakého výpočtu s využitím jedného procesorového jadra na optimalizačnej úrovni O2. Súbor s parametrami výpočtu otvoriteľný v aplikácii *Chuaviz*, je

uvedený v CD prílohe G v priečinku `parameters/cs-optim.txt`.

Tabuľka 5 – 5: Porovnanie časov výpočtu rôznymi optimalizačnými úrovňami [31]

Počítač	32 / 64 bitový operačný systém			
	Od	O1	O2	1 CPU, O2
	t [min]	t [min]	t [min]	t [min]
PC1	2:17 / 1:45	0:57 / 0:28	0:30 / 0:29	1:23 / 1:15
PC2	2:51 / 1:42	1:22 / 0:38	0:39 / 0:38	2:28 / 2:24
PC3	1:43 / 1:35	0:53 / 0:50	0:43 / 0:44	2:24 / 2:29
PC4	6:06 / -	3:24 / -	1:29 / -	3:55 / -
PC5	4:55 / -	2:52 / -	1:27 / -	2:54 / -
PC6	7:40 / -	5:30 / -	2:10 / -	5:19 / -
PC7	0:51 / 0:43	0:24 / 0:18	0:17 / 0:18	1:22 / 1:20
PC8	2:47 / 2:04	1:13 / 0:58	0:59 / 0:57	1:54 / 1:52
PC9	- / 0:35	- / 0:14	- / 0:14	- / 1:06
PC9V	0:49 / 0:38	0:24 / 0:16	0:15 / 0:15	1:18 / 1:17

Ako je z Tab. 5 – 5 zrejmé, 64 bitové výpočty sú vo väčšine prípadov mierne rýchlejšie ako 32 bitové výpočty. Zaujímavé je pritom zrýchlenie 32 bit/64 bit pre PC2 pri optimalizácii Od a O1. Najrýchlejšie CPU sú Intel i7 pre PC7 a PC9. Ide o stolové PC, takže pochopiteľne PC3 s i7, keďže je to mobilný procesor, je pomalší. Zaujímavé je vidieť aj zrýchlenie PC1 (5. generácia CPU) oproti PC3 (2. generácia CPU).

Z tabuľky vidíme aj, že paralelizácia priniesla výrazné zvýšenie výkonu. Najväčšia miera zrýchlenia výpočtu je spozorovateľná na PC9, kde ide o 4,71 násobný nárast výkonu. Ide o štvorjadrový procesor typu Intel i7, ktorý pomocou technológie Hyperthreading poskytuje osem virtuálnych jadier. Z výsledkov je zrejmé, že s Hyperthreading je zrýchlenie viacnásobné, ako je počet procesorových jadier, čo by bez využití virtuálnych jadier nebolo možné. O efektivite paralelizácie svedčí aj fakt, že

na štvorjadrovom procesore bez podpory Hyperthreadingu (PC2) bolo s ňou dosiahnuté 3,79 násobné zrýchlenie pri 64 bitovej verzii, čo sa blíži teoretickému maximu 4. Pri testovaných dvojjadrových procesoroch je toto číslo okolo 2.

Na PC5 bol nainštalovaný aj Windows 10 32 bit, avšak ten neovplyvnil dĺžku výpočtov oproti Windows 7. Zaujímavé zistenie z Tab. 5 – 5 je aj to, že PC2, hoci má staršie CPU, ako PC1 a PC3, je porovnateľne rýchly vo výpočtoch rezu HP. Zároveň je zaujímavé zistenie pre PC9 a 64 bitovej aplikácii, že virtuálny stroj v PC9V je len o málo pomalší pri výpočtoch, ako priamo na fyzickom počítači s operačným systémom Linux Mint.

Celkovo si z výsledkov môžeme usúdiť, že neoplatí sa počítať s aplikáciou preloženou bez optimalizácie. Pri testoch aj nižšia optimalizačná úroveň O1 priniesla výrazné zrýchlenie (pri PC9 až 2,5 násobné). Nárast výkonu pomocou úrovne O2 je spozorovateľný najmä pri starších procesoroch a 32 bitovej verzii aplikácie. Rozdiely vo výkone medzi 32 a 64 bitovými verziami pri optimalizačnej úrovni O2 sú zanedbateľné. Každopádne úroveň O2 ani v jednom prípade neznížila výkon výpočtu, oplatí sa teda ju využívať, najmä ak sú dlhšia doba a vyššie pamäťové nároky prekladu akceptovateľné.

Optimalizovanie prekladačom je efektívny spôsob zrýchlenia výpočtov bez zmeny programového kódu, avšak pri istých trajektóriách bolo spozorované, že niektoré optimalizačné úrovne prekladača vedú k mierne rozličným výsledkom pri dlhšej simulácii trajektórie, oproti iným optimalizačným úrovniam. Tie sú pravdepodobne spôsobené nepresnosťou aritmetiky čísel v pohyblivej rádovej čiare. Ich konkrétne okolnosti a vplyv na výpočet HP však v rámci tejto práce neboli skúmané.

5.3 Preklad pre konkrétnu mikroarchitektúru

Prekladače umožňujú nastaviť aj cieľovú inštrukčnú sadu procesora. V základnom nastavení je zvolená minimálna inštrukčná sada pre danú architektúru (napr. x86-64 pre 64 bitovú), čiže pri preklade sa používajú iba tie inštrukcie, ktoré sú podporované všetkými procesormi v rámci danej architektúry. Zabezpečuje sa tým prenosnosť vytvoreného programu medzi počítačmi. Rôzne procesory však poskytujú rôzne ďalšie inštrukčné sady nad rámec štandardnej, slúžiace na ďalšie zrýchlenie špecifických výpočtov, napr. rozšírenia SSE a AVX, ktoré umožňujú vykonať operácie nad viacerými číslami súčasne (inštrukcie typu SIMD) alebo AES určené na kryptografické účely. Zvolením cieľovej mikroarchitektúry je možné využívať aj tieto inštrukcie.

V tejto časti práce skúmame vplyv použitia rôznych inštrukčných sád na rôznych procesoroch za účelom urýchlenia výpočtu v aplikácii *Chuaviz*. Prekladač GCC poskytuje tri možnosti nastavenia cieľovej inštrukčnej sady [13]:

1. Prepínač `march` – udáva cieľovú mikroarchitektúru, pre ktorú má byť program optimalizovaný. Pri preklade je dostupná celá sada inštrukcií danej mikroarchitektúry, je teda možné, že výsledný program nebude funkčný na všetkých procesoroch.

2. Prepínač `mtune` – optimalizuje výsledný program pre danú mikroarchitektúru, avšak bez použitia nových inštrukcií. Neovplyvňuje teda prenosnosť programu medzi rôznymi procesormi.

3. Sú k dispozícii aj ďalšie prepínače, ktoré umožňujú kontrolovať použitie konkrétnych inštrukcií.

Boli urobené merania rýchlosti výpočtu na rôznych počítačoch s rôznymi optimalizačnými úrovňami prekladača a optimalizáciami pre rôzne mikroarchitektúry. Testoval sa čas potrebný na výpočet jednej trajektórie v dĺžke integračného času

10 000 s. Výkonnostné testy boli vykonané na troch počítačových zostavách: PC8L s procesorom Pentium mikroarchitektúry *Ivy Bridge*, PC9 s procesorom Core i7 architektúry *Haswell* a PC1L s procesorom Core i5 architektúry *Broadwell*. Výpočty, ale aj preklady programu boli uskutočnené pod 64 bitovým systémom Linux.

Bolo vytvorených 20 rôzne preložených verzií programu, ktoré vznikli kombinovaním štyroch hodnôt prepínača `march`, konkrétne: `x86-64`, `ivybridge`, `haswell` a `broadwell`, a piatich optimalizačných úrovní: O0, O1, O2, O3 a Os. Prepínač `march` pri hodnote `x86-64` znamená použitie inštrukčnej sady štandardnej x86-84 architektúry bez rozšírení. Takto prekladaný program je teda spustiteľný na všetkých procesoroch. *Haswell* je v porovnaní s *Ivy Bridge* novšia mikroarchitektúra od firmy Intel, a je s ňou plne kompatibilná, čo však neplatí opačne. Rovnako to je pri mikroarchitektúrach *Haswell* a *Broadwell*, z ktorých je *Broadwell* novšia. Preto na PC8L nebolo možné testovať verzie pre *Haswell* a *Broadwell*, a na PC9 verziu pre *Broadwell*. Dôležité je poznamenať aj to, že pri verzií optimalizovanej pre *Ivy Bridge* bol použitý dodatočný prepínač `mno-avx`, ktorý vypol inštrukcie AVX. Z modelov Pentium bola totiž vynechaná táto sada inštrukcií, ktorá je dostupná vo všetkých vyšších modeloch mikroarchitektúry *Ivy Bridge*. Tab. 5 – 6 uvádza namerané časy pri jednotlivých výpočtoch.

Vo výslednej Tab. 5 – 6 vidíme zaujímavé zistenia. Opäť sa potvrdil prínos zapnutia optimalizácie prekladačom, medzi jednotlivými optimalizačnými úrovňami však nie je výrazný rozdiel. Vidíme, že najlepšie výsledky boli dosiahnuté úrovňou O2, avšak v istých prípadoch (napr. PC1L s verziou `x86-64`) bol Os najrýchlejší medzi všetkými možnosťami. O3 bol najrýchlejší iba v prípade PC1L a `x86-64`, resp. `ivybridge`. Optimalizácia pre konkrétnu mikroarchitektúru priniesla najviac 4,5%-né zrýchlenie výpočtu v prípade PC9 pri úrovni Os a mikroarchitektúre *Haswell*, v niektorých prípadoch však rýchlosť výpočtu znížil. Napr. pri PC8L, O0 a `ivybridge` trval výpočet o 9% dlhšie, ako bez optimalizácie pre mikroarchitektúru.

Tabuľka 5 – 6: Čas výpočtu trajektórie pri optimalizácii pre mikroarchitektúry.

Opt.	Počítač	Čas výpočtu pri rôznych hodnotách `march`.			
		`x86-64`	`ivybridge`	`haswell`	`broadwell`
		t [ms]	t [ms]	t [ms]	t [ms]
O0	PC8L	11 995	13 141	-	-
	PC9	7 241	8 008	7 908	-
	PC1L	11 337	11 242	11 227	11 256
O1	PC8L	7 354	7 487	-	-
	PC9	4 454	4 536	4 391	-
	PC1L	4 843	4 812	4 804	4 810
O2	PC8L	7 252	7 146	-	-
	PC9	4 342	4 319	4 192	-
	PC1L	4 825	4 827	4 759	4 758
O3	PC8L	7 303	7 175	-	-
	PC9	4 389	4 324	4 197	-
	PC1L	4 812	4 823	4 761	4 766
Os	PC8L	7 408	7 424	-	-
	PC9	4 456	4 470	4 258	-
	PC1L	4 737	4 769	4 799	4 800

Môžeme si teda usúdiť, že preklad aplikácie *Chuaviz* s prekladačom GCC pre konkrétnu mikroarchitektúru od Intelu neprináša zreteľné zrýchlenie výpočtov trajektórie. V istých prípadoch môže aj znížiť výkon a v neposlednom rade stráca sa prenosnosť programu medzi rôznymi počítačmi.

Vstupný súbor s parametrami vykonaných testov, pre aplikáciu *Chuaviz*, je dostupný v CD prílohe G v súbore `parameters/traj-long.txt`.

6 Výstupy z aplikácie

V tejto kapitole sú uvedené výstupy výpočtov z aplikácií *Chuaviz* a *Chuaviz DE*, ktoré potvrdzujú ich použiteľnosť. Pri výpočtoch sme vychádzali z parametrov uvedených v knihe [5] označenými skratkami PC pre fyzikálne obvody (physical circuit), DE pre bezrozmerné rovnice (dimensionless equations) a CE pre kubické rovnice (cubic equations).

Vyvinuté aplikácie boli použité v iných prácach už počas písania tejto práce, uvedieme aj ukážku z dosiahnutých výsledkov.

6.1 Atraktory

Hlavným motivátorom tejto práce bola kniha [5], ktorá uvádza niekoľko sto chaotických atraktorov spolu s parametrami obvodov. Na nasledujúcich stranách uvádzame vizuálne porovnanie vybraných atraktorov z knihy, s grafickým výstupom aplikácií *Chuaviz* a *Chuaviz DE*. Táto malá galéria atraktorov slúži na poukázanie správnosti aplikácií *Chuaviz* a *Chuaviz DE*, a možnosť ich použitia na výpočet HP obvodov prezentovaných v knihe [5], resp. iných obvodov.

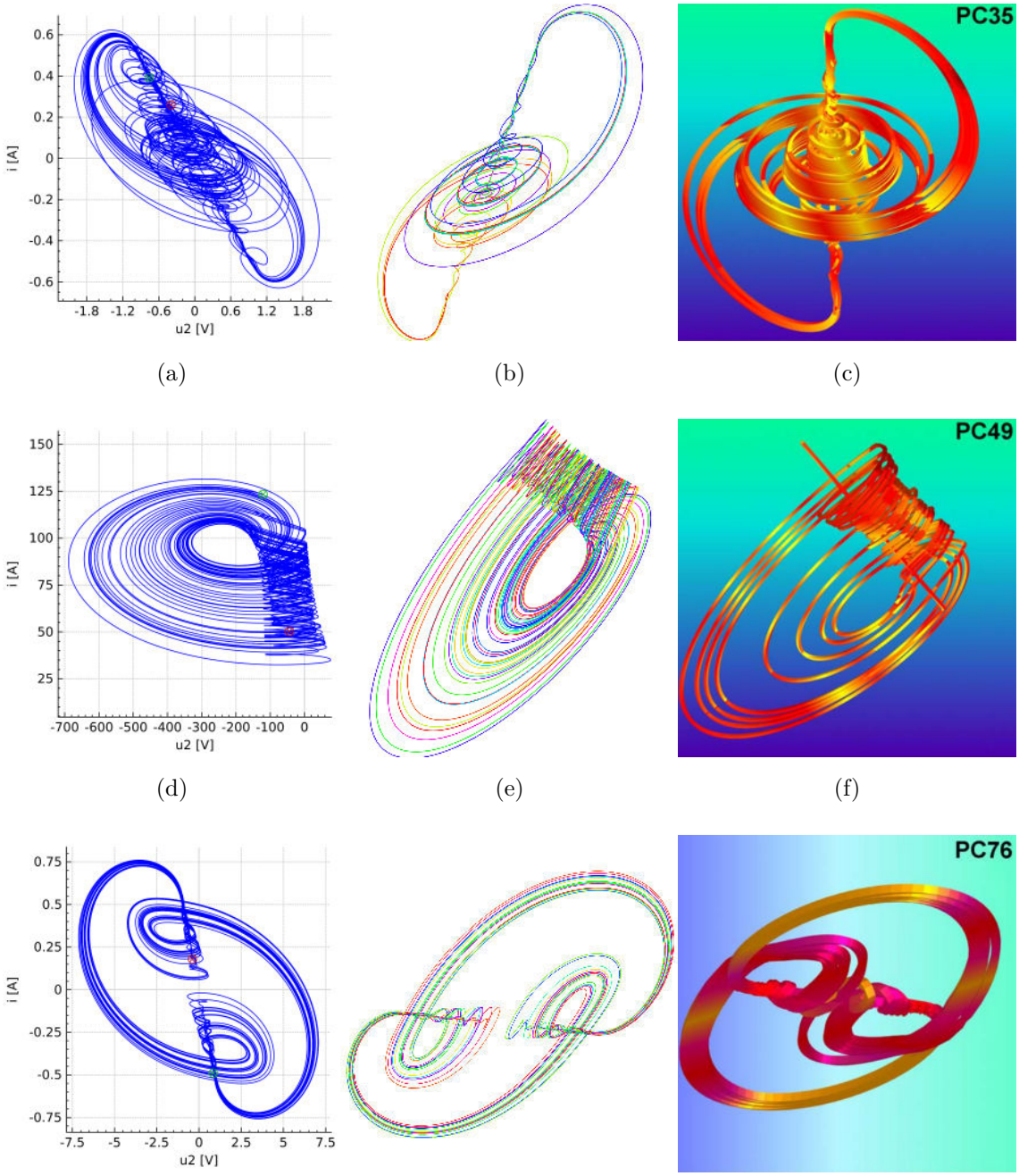
Počítané boli atraktory fyzikálnych obvodov (označené podľa knihy ako PC) s 3-segmentovou VA charakteristikou s aplikáciou *Chuaviz*, bezrozmerných modelov obvodov s 3-segmentovou funkciou $f(x)$ (ozn. ako DE) a bezrozmerných modelov obvodov s kubickou funkciou $f(x)$ (ozn. ako CE) s aplikáciou *Chuaviz DE*. Ku všetkým atraktorom sú uvedené aj parametre obvodov.

6.1.1 Fyzikálne obvody

Na Obr. 6 – 1 môžeme vidieť chaotické atraktory Chuaovho obvodu pre tri rôzne parametre uvedené v Tab. 6 – 1. V každom riadku je zobrazený jeden atraktor, pričom prvý a druhý obrázok sú snímky z aplikácie *Chuaviz*, kým tretí je z knihy [5]. Prvý riadok (Obr. 6 – 1a, 6 – 1b, 6 – 1c) zobrazuje atraktor obvodu PC35, druhý riadok (Obr. 6 – 1d, 6 – 1e, 6 – 1f) PC49 a tretí riadok (Obr. 6 – 1g, 6 – 1h, 6 – 1i) PC49. Súbory s parametrami pre aplikáciu *Chuaviz* je možné nájsť v priečinku `parameters/chuaviz/pc` na pripojenom CD (príloha G).

Tabuľka 6 – 1: Parametre fyzikálnych obvodov (PC)

	PC35	**PC49**	**PC76**
C_1 [F]	1	1	1
C_2 [F]	−1,0837	−1,0837	−1,0837
R [Ω]	0,03	0,02947	0,02947
L [H]	−1,49	−10	−9,7136
ρ [Ω]	2,228	2,228	4,65
m_0 [S]	−0,51	−100	−0,5
m_1 [S]	0,0064	−0,003	0,0064
$u_{1_{zp}}$ [V]	−0,7585631	−113,5619	0,848773
$u_{2_{zp}}$ [V]	−0,75197965	−120,41357	0,84331024
i_{zp} [A]	0,39119202	123,32288	−0,48951969
E [V]	1	1	1



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

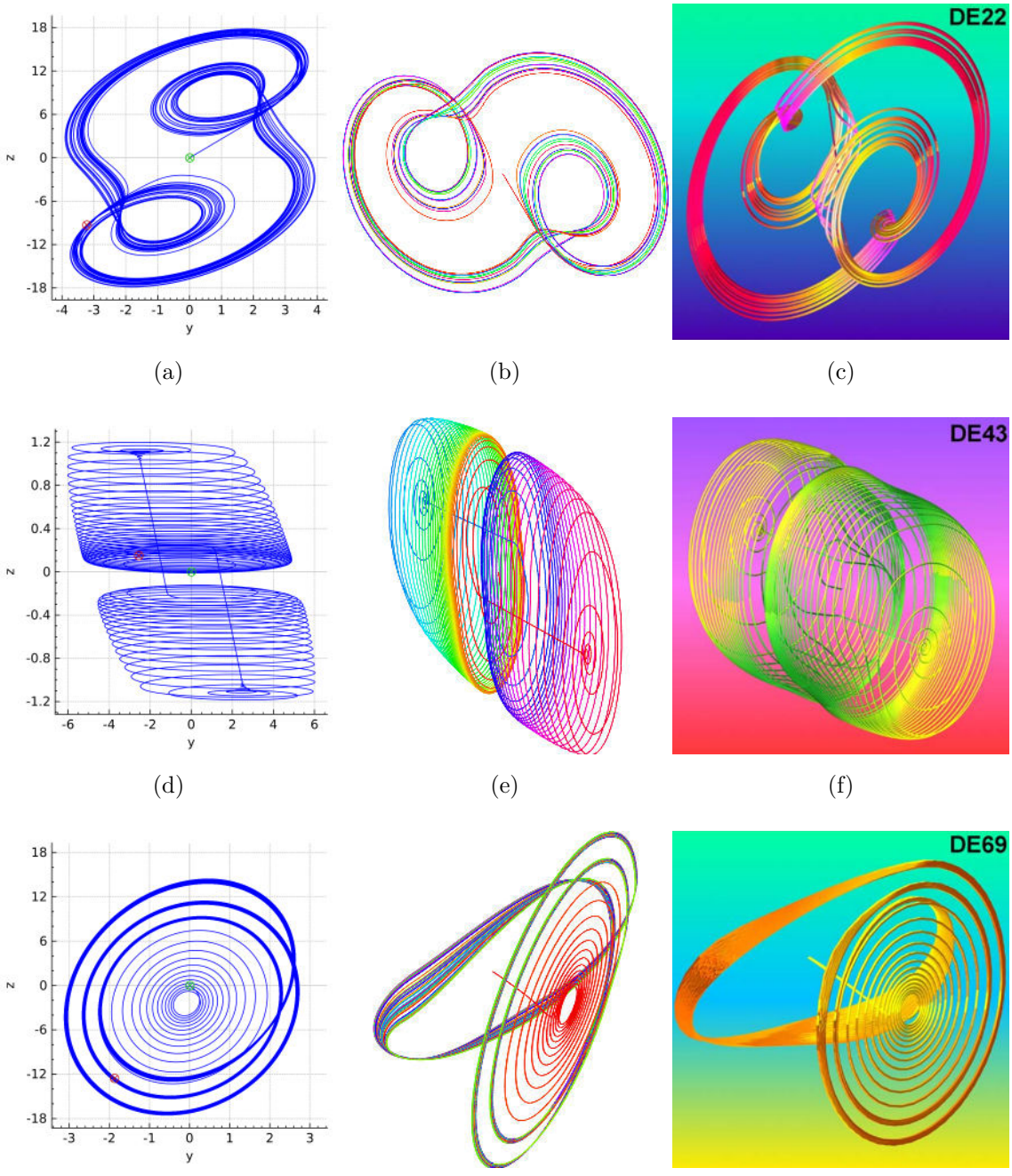
Obr. 6 – 1: Vybrané chaotické atraktory fyzikálnych odvodov s 3-segmentovou funkciou diódy. a), d), g) – zobrazenie v rovine (i, u_2) v *Chuaviz*; b), e), h) – perspektívne 3D zobrazenie v *Chuaviz*, c), f), i) – pohľad na atraktor z publikácie [5].

6.1.2 Bezrozmerné modely obvodov s 3-segmentovou funkciou *f(x)*

Obr. 6 – 2 uvádza chaotické atraktory bezrozmerného modelu Chuaovho obvodu s 3-segmentovou funkciou Chuaovej diódy. Podobne, ako pri fyzikálnych obvodov v predchádzajúcej kapitole, obrázky sú zobrazené v mriežke, pričom prvý a druhý obrázok v danom riadku sú snímky z aplikácie *Chuaviz DE* a tretí z knihy [5]. Prvý riadok (Obr. 6 – 2a, 6 – 2b, 6 – 2c) zobrazuje atraktor obvodu DE22, druhý riadok (Obr. 6 – 2d, 6 – 2e, 6 – 2f) DE43 a tretí riadok (Obr. 6 – 2g, 6 – 2h, 6 – 2i) DE69. Parametre obvodov sú uvedené v Tab. 6 – 2. Súbory s parametrami pre aplikáciu *Chuaviz DE* je možné nájsť v priečinku `parameters/chuaviz-de/de` na pripojenom CD (príloha G).

Tabuľka 6 – 2: Parametre bezrozmerných obvodov s 3-segmentovou funkciou (DE)

	DE22	**DE43**	**DE69**
α	−4,898979	−1,515	3,7091002664
β	−3,624135	−0,0136073	24,0799705758
γ	0,364	−0,02969968	−0,779
a	−2,501256	0,1690817	−2,7647222013
b	−0,9297201	−0,4767822	0,1805569489
κ	1	1	1
x_{zp}	0,02	0,02	0,02
y_{zp}	0,01	0,01	0,01
z_{zp}	0	0	0
E	1	1	1



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Obr. 6 – 2: Vybrané chaotické atraktory s 3-segmentovou funkciou diódy. a), d), g) – zobrazenie v rovine (z, y) v *Chuaviz DE*; b), e), h) – perspektívne 3D zobrazenie v *Chuaviz DE*, c), f), i) – pohľad na atraktor z publikácie [5].

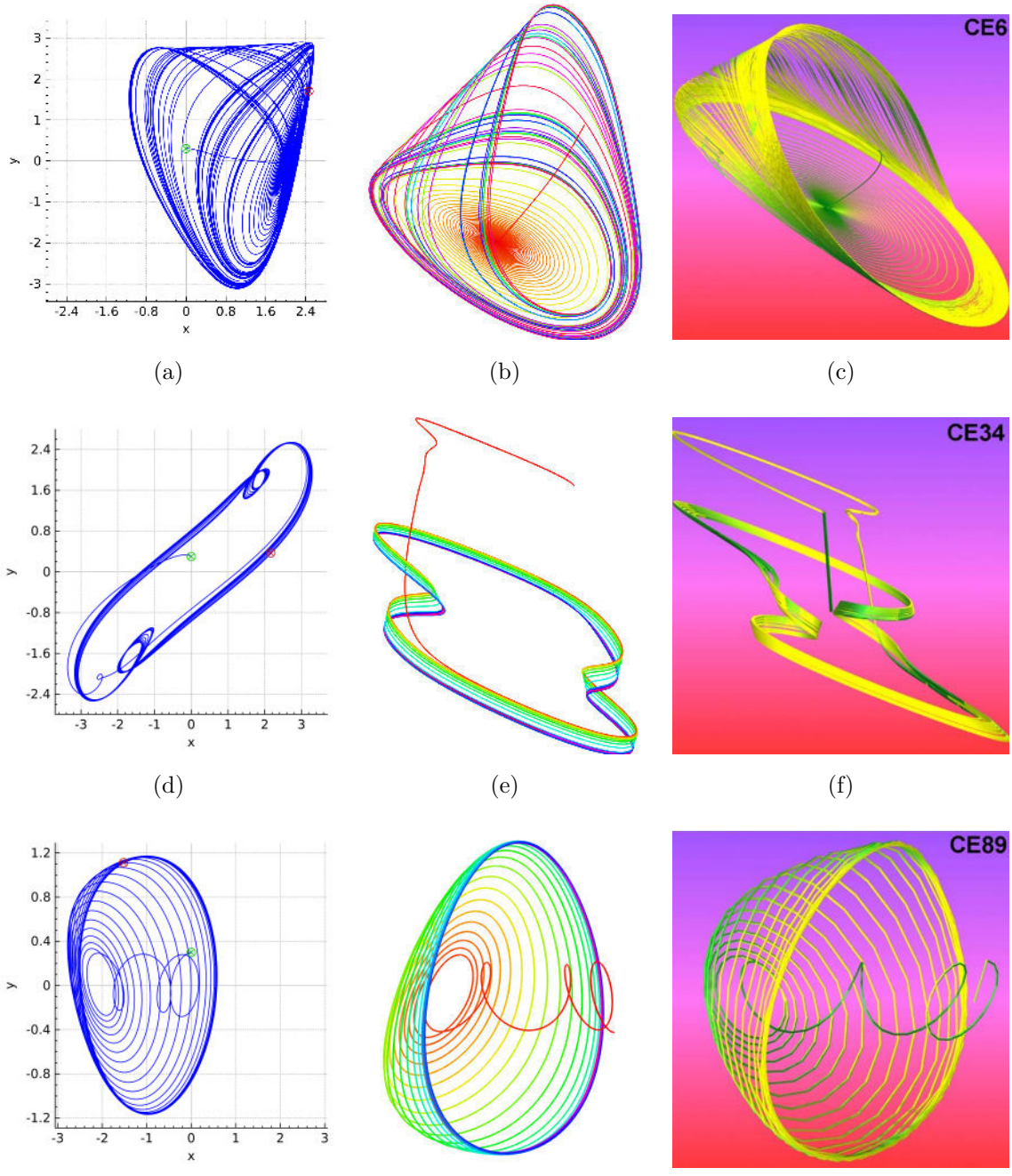
6.1.3 Bezrozmerné modely obvodov s kubickou funckciou *f(x)*

Atraktory bezrozmerného modelu Chuaovho obvodu s kubickou funkciou Chuaovej diódy sú zobrazené na Obr. 6 – 3. Spôsob uvádzania výsledkov je rovnaký, ako v predchádzajúcej kapitole pri obvodoch s 3-segmentovou funkciou diódy.

Prvý riadok (Obr. 6 – 3a, 6 – 3b, 6 – 3c) zobrazuje atraktor obvodu CE6, druhý riadok (Obr. 6 – 3d, 6 – 3e, 6 – 3f) CE34 a tretí riadok (Obr. 6 – 3g, 6 – 3h, 6 – 3i) CE89. Parametre obvodov sú uvedené v Tab. 6 – 3. Pri všetkých výpočtoch bola použitá kubická funkcia Chuaovej diódy s hodnotami $h_0 = 0$, $h_2 = 0$ a h_1, h_3 uvedenými v Tab. 6 – 3. Súbory s parametrami pre aplikáciu *Chuaviz DE* je možné nájsť v priečinku `parameters/chuaviz-de/ce` na pripojenom CD (príloha G).

Tabuľka 6 – 3: Parametre bezrozmerných obvodov s kubickou funkciou (CE)

	CE6	**CE34**	**CE89**
α	2.96	−1.301814	12.141414
β	24.07997058	−0.0136073	95.721132
γ	−0.859255678	−0.02969968	−0.8982235
h_1	−2.9315446532	0.163	−0.8415410391
h_3	0.4530092443	−0.0534372394	−0.0375582129
κ	1	1	−1
x_{zp}	0	0	0
y_{zp}	0.3	0.3	0.3
z_{zp}	0.5	0.5	0.5



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Obr. 6 – 3: Vybrané atraktory s kubickou funkciou diódy. a), d), g) – zobrazenie v rovine (y, x) v *Chuaviz DE*; b), e), h) – perspektívne 3D zobrazenie v *Chuaviz DE*, c), f), i) – pohľad na atraktor z publikácie [5].

6.2 Rezy hraničnými plochami

Pomocou aplikácie *Chuaviz* bol robený výskum morfológie HP pre rôzne parametre fyzikálnych obvodov z knihy [5]. Výskum bol vedený konzultantom tejto práce. Na začiatku sme sa na základe doterajších výsledkov domnievali, že HP bude v 3D stále mať tvar rúry s výskytom (ostrého) „zubu“ pozdĺž jej boku podobne, ako bolo predstavené na Obr. 2 – 7, alebo zvitkovitý tvar, pri dvoch single-scroll CHA. Túto úvahu potvrdzovali vypočítané rezy pre parametre PC1 až PC4 HP predstavené v publikácii [14].

Na začiatku výpočtu rezov HP pomocou aplikácie *Chuaviz*, nebolo nič zaujímavé. Počítalo sa pre parametre PC1 až PC6 a morfológia HP bola taká, akú sme čakali podľa [14]. Pre PC5 vypočítaný rez HP (Obr. 6 – 4a) ukázal zvitkovitý tvar HP s dvoma chaotickými RA naznačenými červenou a zelenou farbou, pretože v obvode sa vyskytujú 2 single-scroll CHA. Pre PC6 (Obr. 6 – 4b) sú RA jednofarebné podľa očakávania. Rezy HP sa avšak nevyznačovali (ostrým) „zubom“ tak, ako to vidieť na Obr2 – 7. Podobné typy rezov rozmerovo menšie alebo väčšie boli pozorované pre parametre uvedené v [33] od PC7 do PC15.

Prvým prekvapením netypického rezu HP boli parametre PC16 až PC19. Morfológia RA bola iná ako to bolo doteraz publikované. Ako je z Obr. 6 – 4d pre PC17 a z Obr. 6 – 4e pre PC18 zrejmé, ide o nejaký mix celistvého RA jednej farby (pre double-scroll CHA) a zvitkovitej štruktúry odpovedajúcej výskytu dvoch single-scroll CHA. Doteraz takáto morfológia BS nebola známa. Ešte väčšie prekvapenie priniesli parametre PC20 (Obr. 6 – 4f). Tento rez bol niekoľkokrát preverovaný zmenou veľkosti integračného kroku aj presnosti výpočtu, avšak charakter rezu HP ostal zachovaný.

Z rezu HP pre parametre PC20 (Obr. 6 – 4f) vidíme nasledujúce charakterisitky:

- vo vnútri červeného RA pre CHA sa objavili štyri malé žlté regióny. Žltá farba

je vyhradená pre také IC, ktoré smerujú do nekonečna;

- celý červený región je popretkávaný žltými bodmi.

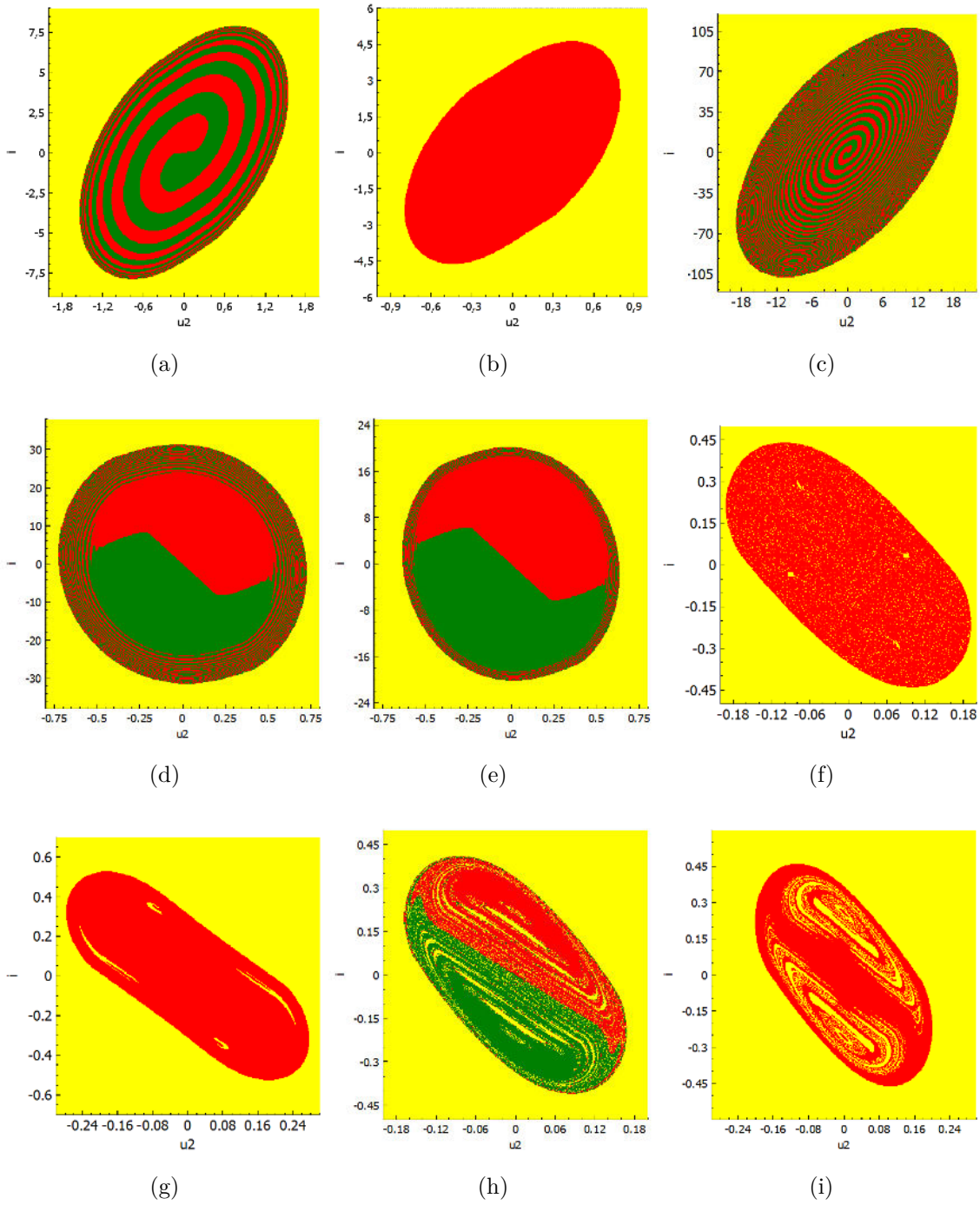
Keď bol overovaný pohyb ZB pomocou simulácie trajektórie v aplikácii *Chuaviz* pri zvolených ZP ležiacich vedľa seba v žltom alebo červenom regióne, tie boli pritiahnuté do nekonečna alebo ku CHA podľa očakávania. Zdá sa teda, že Obr. 6 – 4f bol vypočítaný správne.

RP ilustrované na Obr. 6 – 4f však neboli jediným prekvapením. Z nasledujúcich obrázkov na Obr. 6 – 4 pre parametre PC31 (Obr. 6 – 4g), PC32 (Obr. 6 – 4h) a PC36 (Obr. 6 – 4i) je zrejmé prečo. Pri výskyte jediného CHA sme očakávali iba dva jednofarebné RP – žltý a červený a pri výskyte dvoch CHA zasa červený, zelený a žltý RP. Uvedené RP sa síce nachádzajú na obrázkoch, ale sú opäť popretkávané žltou farbou odpovedajúcej atraktoru nekonečno. Takéto prípady doteraz neboli známe, pravdepodobne je to prvá zmienka o takomto nezvyčajnom type rezu HP.

Prípady HP pri konvenčnej morfológii, ako pri PC5 (Obr. 6 – 4a) a PC6 (Obr. 6 – 4b) je možné zrekonštruovať z vypočítaných 2D rezov HP. Príklady rezov uvedené na Obr. 6 – 4, od parametrov PC15 však kvôli narušeniu celistvosti červeného alebo zeleného RA pre CHA, nie je možné rekonštruovať a vizualizovať v 3D, metódami uvedenými v prácach [15] [17] a [20].

Parametre obvodu, s ktorými boli rezy HP vypočítané uvádza Tab. 6 – 4. Pri výpočte bola použitá 3-segmentová VA charakteristika s hodnotou $E = 0$.

Spomenuté výsledky boli uvedené aj v článku [16], ktorý je aktuálne na recenzii IEEE konferencie TSP 2017, Barcelona. Prezentované rezy HP narúšajú doterajšiu konvenčnú predstavu o morfológii HP a prinášajú nové, doteraz nepublikované typy rezov HP. Spoluautorom článku je aj autor tejto práce. Článok v plnom znení je priložený v prílohe F.



Obr. 6 – 4: Nové tvary HP objavené aplikáciou *Chuaviz* – rezy v rovine i, u_2. a) – PC5, b) – PC6, c) – PC15, d) – PC17, e) – PC18, f) – PC20, g) – PC 31, h) – PC32, i) – PC36 [16].

Tabuľka 6 – 4: Parametre pre prípady výpočtu rezov HP. Parametre sú vyjadrené nasledovne: C_1 a C_2 v [F], R v [Ω], L v [H], G_a a G_b v [S] [16]

	C_1	C_2	R	L	ρ	G_a	G_b
PC1	0,10443	1	0,989119	0,0625	0	−1,143	−0,714
PC2	0,10443	0,981	1	0,0625	0	−1,143	−0,714
PC3	0,10443	0,85	1	0,0625	0	−1,143	−0,714
PC4	0,10443	1	1	0,0625	0	−1,2	−0,714
PC5	−0,1333	8,2	1	0,31	−0,1	−0,98	−2,4
PC6	−0,1333	11	1	0,31	−0,1	−0,98	−2,4
PC13	−0,1333	10	0,786782	0.31	−0,1	−0,98	−2,4
PC15	−0,1333	10	0,778816	0,31	−0,1	−0,98	−2,4
PC17	0,06	15,35	−1	0,00667	0,000651	0,856	1,1
PC18	0,06	10	−1	0,0075	0,000651	0,856	1,1
PC19	0,06	10	−1	0,0068	0,000651	0,856	1,1
PC20	1	−1,0837	33,33333	−1,49	2.228	−0,5	0,0064
PC31	1	−1,163	33,33333	−1,49	2,228	−0,5	0,0064
PC32	1,01	−1,0837	33,33333	−1,49	2,228	−0,5	0,0064
PC36	1	−1,0837	33,33333	−1,49	2,228	−0,52	0,0064

6.3 3D projekcia hraničných plôch

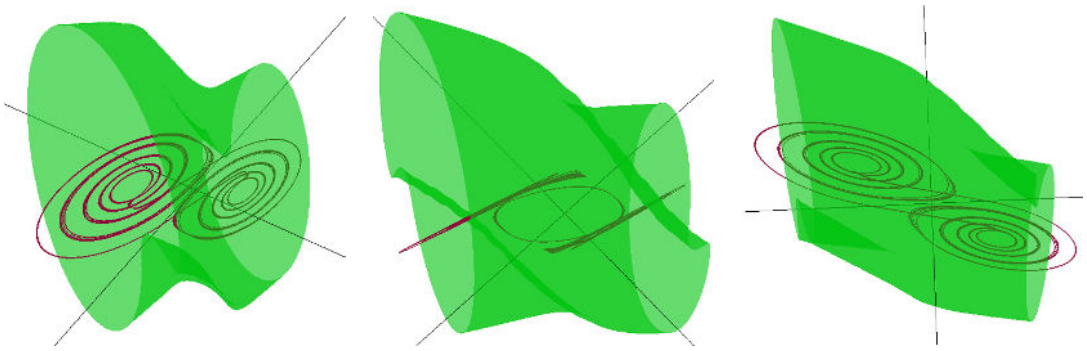
Ako to bolo spomenuté aj v kapitole 2.6, pospájaním série paralelných rezov HP, je možné rekonštruovať HP v 3D. Aplikácia *Chuaviz* umožňuje jednoducho vypočítať sériu rezov, ktoré okrem možnosti priamej vizualizácie sú aj ukladané na úložisko v textovom formáte. Tieto výstupné súbory boli ďalej spracované v práci [17], ktorá sa zaoberá s 3D vizualizáciou HP.

Postup pri 3D vizualizácii HP v práci [17] bol nasledovný:

1. Výpočet série rezov HP v aplikácii *Chuaviz*.
2. Preformátovanie výstupného súboru (zmena poradia stĺpcov, vymazanie nepotrebných informácií, atď).
3. Spracovanie súborov pomocou aplikácie *ShapeMaker* z práce [20], resp. s inými nástrojmi. Výstupom tohto kroku je jeden PLY súbor.
4. Načítanie do programu *3ds Max*.
5. Úprava v programe *3ds Max* (pospájanie kontúr, aplikovanie možností surface, relax, multires, atď.).

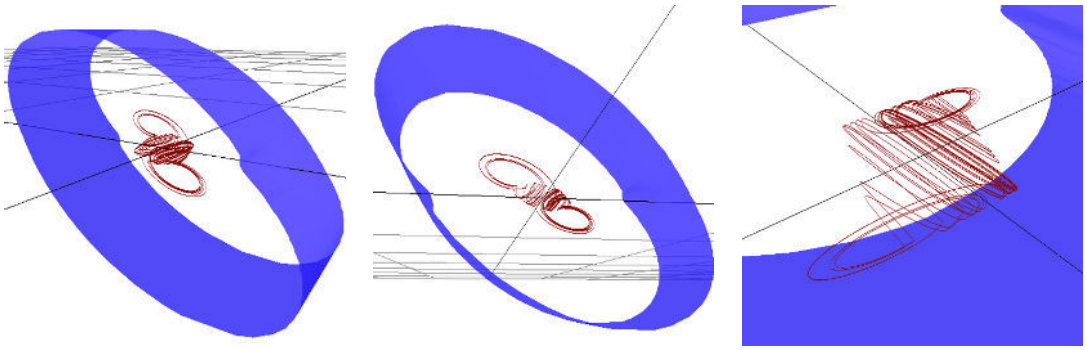
Ukážku z výsledkov práce [17] uvádzame na Obr. 6 – 5, Obr. 6 – 6 a Obr. 6 – 7, na ktorých môžeme vidieť HP a chaotickú trajektóriu pohybu ZB pre parametre Chuaovho obvodu PC4, PC9 a PC26, umiestnené v 3D priestore, z pohľadu rôznych uhlov. Pre vizualizáciu bol použitý program *3ds Max*. Všetky HP zobrazené na spomenutých obrázkoch boli vypočítané pomocou aplikácie *Chuaviz*, vyvinutej v rámci tejto práce. Vstupné súbory s parametrami pre aplikáciu *Chuaviz* sú umiestnené v CD prílohe G, v priečinku `parameters/chuaviz/pc`.

Okno s 3D projekciou trajektórie v aplikácii *Chuaviz* umožňuje načítať ľubovoľnú množinu bodov z textového súboru a zobraziť ich pri vykreslenej trajektórii. Táto



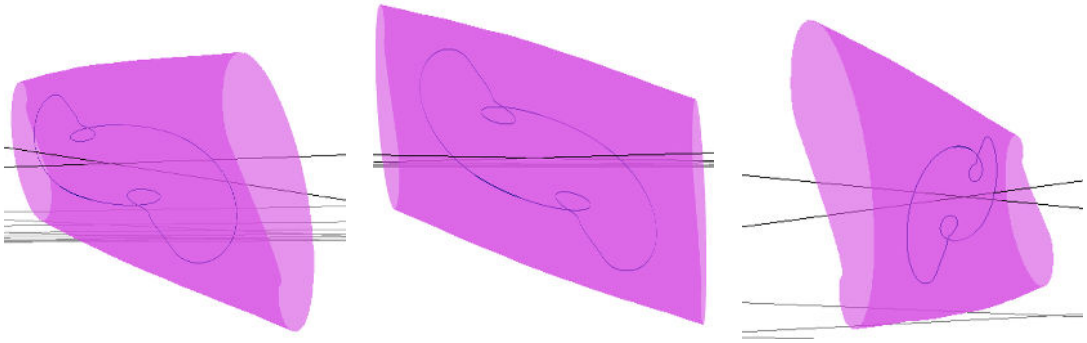
(a) (b) (c)

Obr. 6 – 5: 3D pohľady na hraničnú plochu pre parametre PC4 [17]



(a) (b) (c)

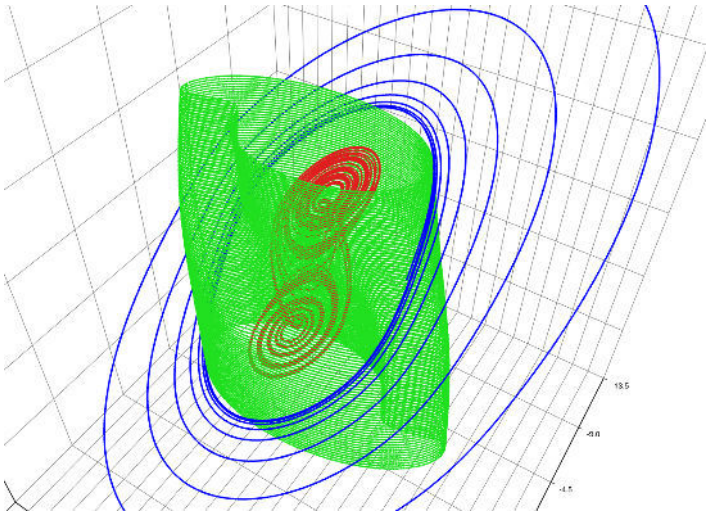
Obr. 6 – 6: 3D pohľady na hraničnú plochu pre parametre PC9 [17]



(a) (b) (c)

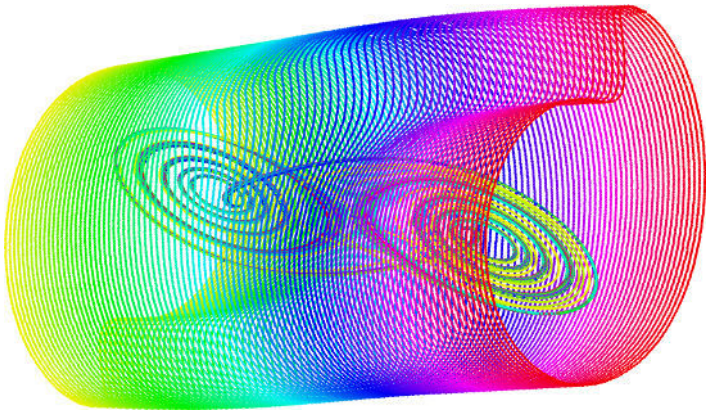
Obr. 6 – 7: 3D pohľady na hraničnú plochu pre parametre PC26 [17]

možnosť pôvodne slúžila na zobrazenie viacerých trajektórií súčasne, avšak zaujímavou bolo zobrazenie HP v tejto projekcii. Autor práce [17] poskytol spracované rezy HP v súboroch formátu PLY. Mierne upravené súbory boli skúšobne importované do aplikácie *Chuaviz*. Výsledok je uvedený na Obr. 6 – 8, kde je súčasne zobrazená chaotická trajektória, SLC a úsek HP pre parametre PC4.



Obr. 6 – 8: 3D zobrazenie HP a dvoch atraktorov pre parametre PC4

Z výsledku je zreteľne vidno vzájomnú polohu HP a atraktorov. Názornosť vylepšuje možnosť otáčania a skúmania objektov z každého uhla. Ešte zaujímavejší pohľad sme dostali po zapnutí pestrofarebného zobrazenia uvedeného na Obr. 6 – 9.



Obr. 6 – 9: 3D zobrazenie HP a CHA pre parametre PC4 v pestrofarebnom móde

7 Záver

Hlavným cieľom tejto práce bolo poskytnúť softvérové riešenie pre výskum Chuaovho obvodu. V rámci práce boli preštudované a popísané oblasti ako: teória chaosu, Chuaov obvod a matematické podklady pre výpočet HP a trajektórií. Ďalej boli porovnané existujúce programové riešenia zaoberajúce sa výpočtom alebo vizualizáciou Chuaovho obvodu. Pri porovnaní však bola zistená potreba vytvoriť nový softvér, keďže existujúce nespĺňali všetky požiadavky.

Preštudované poznatky boli použité pri návrhu a implementácii aplikácie, ktorý dostal názov *Chuaviz*. Aplikácia umožňuje počítať trajektórie v stavovom priestore a vizualizovať ich pomocou 2D a 3D projekcií. Ďalej poskytuje možnosť paralelného výpočtu rezov hraničnými plochami a ich 2D vizualizácie. Ponúka široké možnosti nastavenia parametrov výpočtu, ako zmenu parametrov obvodu, VA charakteristík, adaptívnu kontrolu integračného kroku a mnoho ďalších.

Bola vytvorená aj modifikácia aplikácie *Chuaviz*, ktorá dostala názov *Chuaviz DE*. Aplikácia je funkčne totožná s *Chuaviz*, avšak namiesto fyzických veličín počíta s bezrozmerným modelom Chuaovho obvodu ku čomu je prispôsobené aj jej grafické rozhranie.

Samostatná kapitola sa venuje optimalizácii rýchlosti výpočtov. Skúmané sú parametre výpočtu, optimalizácie prekladačom a paralelizácia. Všetky navrhnuté optimalizácie boli otestované na rôznych počítačových zostavách, a výsledky testov sú prezentované v práci. Posledná kapitola práce uvádza grafické výstupy z aplikácií *Chuaviz* a *Chuaviz DE*, tá poukazuje na ich funkčnosť a použiteľnosť prí výskume.

Zároveň na základe výsledkov diplomovej práce boli vyhotovené dva vedecké články, z ktorých jeden bol publikovaný a prezentovaný autorom práce na medzinárodnej konferencii IEEE *Radioelektronika* v Brne. Druhý je podmienečne schválený na IEEE konferenciu *TSP* v Barcelone. Články sú uvedené v Prílohe E a v Prílohe F.

Výhoda navrhnutej aplikácie spočíva v tom, že aplikácia umožnila výrazné zrýchlenie výpočtu na akomkoľvek bežnom, dnes používanom viac-vláknovom počítači či notebooku. Používaním aplikácie boli nájdené nové, doteraz nepublikované atraktory a morfológie hraničných plôch. Časť poznatkov z tejto práce už bola premietnutá do už spomínaných konferenčných príspevkov sprístupnených v databáze IEEE.

Navrhnutá aplikácia tiež umožnila nielen zobrazenie galérie chaotických atraktorov, ale aj ich rýchle zobrazenie bez prechodných dejov pôsobených začiatočnými podmienkami uvedenými v [5]. Spolu s vypočítanými rezmi HP tak pripravila základ pre rekonštrukciu HP v 3D a následné zobrazenie ako atraktora aj HP prostriedkami virtuálnej reality.

V budúcnosti je možné aplikáciu rozvinúť napr. do oblasti výpočtu vlastných čísel či Ljapunovských exponentov, na výpočet ktorých je paralelizácia nutnosťou. Taktiež je možné navrhnutú aplikáciu rozšíriť o nové sústavy diferenciálnych rovníc generujúcich chaos. Postup pri takomto rozširovaní je uvedený v prílohe D.

Literatúra

[1] Andrews, T. *Computation Time Comparison Between Matlab and C++ Using Launch Windows* [online]. California Polytechnic State University, 2012. Dostupné z: <http://digitalcommons.calpoly.edu/aerosp/78>.

[2] Assencio, D. *The double pendulum: Lagrangian formulation* [online]. 2014. [cit. 2017.3.22]. Dostupné z: <https://diego.assencio.com/?index=1500c66ae7ab27bb0106467c68feebc6>.

[3] Bača, P. The utilization of multi-core CPU for computation of boundary surfaces. Diplomová práca, Technická univerzita v Košiciach, Košice, 2014.

[4] Bene, G. – Gruiz, M. *Kaotikus dinamika* [online]. BME TTK, 2012. [cit. 2017.4.4]. Dostupné z: <http://tankonyvtar.ttk.bme.hu/pdf/156.pdf>.

[5] Bilotta, E. – Pantano, P. *Gallery of Chua Attractors*. 61 / *A*. World Scientific Publishing Co. Pte. Ltd., 2008. 533 s. ISBN 978-981-279-062-0.

[6] Chroboczek, M. *Grafická uživatelská rozhrácí v Qt a C++*. Computer Press, 2013. ISBN 978-80-251-4124-3.

[7] Contreras, G. – Martonosi, M. Characterizing and improving the performance of Intel Threading Building Blocks. In *2008 IEEE International Symposium on Workload Characterization*, s. 57–66, Sept 2008. doi: 10.1109/IISWC.2008.4636091.

[8] Eichhammer, E. QCustomPlot (verzia 1.3.2), 2015. Dostupné z: <http://www.qcustomplot.com>.

[9] Farnham, K. *TBB Parallel_for Grain Size Experiments* [online]. Intel Developer Zone, 2007. [cit. 2017.1.23]. Dostupné z:

<https://software.intel.com/en-us/blogs/2007/08/07/tbb-parallel_for-grain-size-experiments>.

[10] Farnham, K. *Hierarchically Tiled Arrays and Threading Building Blocks* [online]. Intel Developer Zone, 2007. [cit. 2017.1.25]. Dostupné z: <https://software.intel.com/en-us/blogs/2007/09/28/hierarchically-tiled-arrays-and-threading-building-blocks>.

[11] Fourment, M. – Gillings, M. R. A comparison of common programming languages used in bioinformatics. *BMC Bioinformatics*. 2008, 9, 1, s. 82. doi: 10.1186/1471-2105-9-82. Dostupné z: <http://dx.doi.org/10.1186/1471-2105-9-82>.

[12] GCC Team. *GNAT User's Guide for Native Platforms: Optimization Levels* [online]. Free Software Foundation, Inc., 2017. [cit. 2017.4.25]. Dostupné z: <https://gcc.gnu.org/onlinedocs/gnat_ugn/Optimization-Levels.html>.

[13] GCC Team. *x86 Options - Using the GNU Compiler Collection (GCC)* [online]. Free Software Foundation, Inc., 2017. [cit. 2017.4.25]. Dostupné z: <https://gcc.gnu.org/onlinedocs/gcc-5.4.0/gcc/x86-Options.html>.

[14] Guzan, M. Variations of Boundary Surface in Chua's Circuit. *Radioengineering*. 2015, 24, 3, s. 814–823. ISSN 1210-2512.

[15] Guzan, M. – Sobota, B. *Vizualizácia stavového priestoru nelineárnych autonómnych obvodov*. JUPOS, 2015. 138 s. ISBN 978-80-87321-31-7.

[16] Guzan, M. et al. Cross-Sections of Boundary Surface for Variations of Parameters in Chua's Circuit. *International Conference on Telecommunications and Signal Processing*. 2017, 40, 1. (na recenzii).

[17] Halachan, P. 3D vizualizácia hraničných plôch Chuaovho obvodu. Diplomová práca, Technická univerzita v Košiciach, Košice, 2017.

[18] Heczko, S. Teorie chaosu a chování otevřených systémů. *Marathon*. 2004, 47, s. 29–24. ISSN 1211-8591.

[19] Intel® Software. *Algorithms* [online]. Intel Developer Zone, 2017. [cit. 2016.11.11]. Dostupné z: <https://software.intel.com/en-us/node/506140>.

[20] Koreny, A. Vizualizácia 3D hraničných plôch z 2D rezov. Diplomová práca, Technická univerzita v Košiciach, Košice, 2015. 99 s.

[21] Krakovská, A. Teória chaosu a fyziológia srdca. *Československá fyziologie*. 2001, 50, 4, s. 192–200. ISSN 1210-6313.

[22] Link, T. Vizualizácia mračna bodov v reálnom čase. Diplomová práca, Technická univerzita v Košiciach, Košice, 2016. 91 s.

[23] Lorenz, E. Predictability: Does the Flap of a Butterfly's wings in Brazil Set Off a Tornado in Texas? American association for the advancement of science, 139th meeting, 1972. Dostupné z: <http://eaps4.mit.edu/research/Lorenz/Butterfly_1972.pdf>.

[24] Madan, R. N. *Chua's circuit: A Paradigm for chaos*. World Scientific Publishing Co. Pte. Ltd., 1993. ISBN 981-02-1366-2.

[25] Merah, L. et al. A Pseudo Random Number Generator Based on the Chaotic System of Chua's Circuit, and its Real Time FPGA Implementation. *Applied Mathematical Sciences*. 2013, 7, 55, s. 2719–2734. ISSN 1314-7552.

[26] Microsoft. *O Options (Optimize Code)* [online]. Microsoft Developer Network, 2017. [cit. 2017.4.25]. Dostupné z: <https://msdn.microsoft.

com/en-us/library/k1ack8f1.aspx>.

[27] Mital, P. B. – Kumar, U. – Prasad, R. S. Chua's Circuit – A Universal Paradigm for Generating and Studying Chaos. *J. of Active and Passive Electronic Devices*. 2008, 3, s. 51–63. Dostupné z: <http://www.oldcitypublishing.com/FullText/JAPEDfulltext/JAPED3.1fulltext/JAPEDv3n1p51-63Mital.pdf>.

[28] Mocnár, R. Vizualizácia chaotického atraktora v stavovom priestore. Diplomová práca, Technická univerzita v Košiciach, Košice, 2006. 51 s.

[29] Nezin, C. – So, B. *Chua's Circuit* [online]. brendabrandy at GitHub.com, 2016. [cit. 2017.04.5]. Dostupné z: <https://github.com/brendabrandy/chuacircuit/wiki>.

[30] Prechelt, L. An empirical comparison of C, C++, Java, Perl, Python, Rexx, and Tcl for a search/string-processing program. Technical Report 2000-5, Fakultät für Informatik, Univ. Karlsruhe, Germany, March 2000.

[31] Rácz, Zs. – Guzan, M. – Sobota, B. Parallelizing boundary surface computation of Chua's circuit. *Radioelektronika*. 2017, 27, 1.

[32] Simsik, D. et al. Wearable Non-Invasive Computer Controled System for Improving of Seniors Gait. In *Proceedings of the 10th Congress of the European Federation for Research in Rehabilitation – EFRR*, s. 19–24, Riga, Latvia, sept. 2009.

[33] Spany, V. – Galajda, P. – Guzan, M. The state space mystery in multiple-valued logic circuit with load plane – part I. *Acta Electrotechnica et Informatica*. 2001, 1, 1, s. 17–22. ISSN 1335-8243.

[34] Stanford Complexity Group. *Double Pendulum Simulation* [online]. Stanford University, 2014. [cit. 2017.3.17]. Dostupné z: <http://complexity.

stanford.edu/blog/double-pendulum-simulation>.

[35] Strömbäck, P. QPlot3D, 2014. Dostupné z: <https://github.com/pstrom77/QPlot3D>.

[36] The Qt Company. *All modules* [online]. Qt Documentation, 2017. [cit. 2016.11.11]. Dostupné z: <http://doc.qt.io/qt-5/qtmodules.html>.

[37] The Qt Company. *Supported Platforms* [online]. Qt Documentation, 2016. [cit. 2016.11.10]. Dostupné z: <http://doc.qt.io/qt-5/supported-platforms.html>.

[38] Tél, T. – Gruiz, M. Mi a káosz? *Természet Világa*. 2002, 133, 7, s. 296–298. ISSN 0040-3717. Dostupné z: <http://theorphys.elte.hu/tel/magyar/pdf_pub/Mi%20a%20k%C3%A1osz.pdf>.

[39] Vasilovici, O. – Irimiciuc, S. A. – Dimitriu, D. G. Signal Encryption Using Chua's Chaotic Circuits. *Journal of Advanced Research in Physics*. 2003, 4, 1. Dostupné z: <http://stoner.phys.uaic.ro/jarp/index.php/jarp/article/download/134/83>.

[40] W. Kahan. *IEEE Standard 754 for Binary Floating-Point Arithmetic* [online]. University of California, Berkeley, 1997. [cit. 2016.10.25]. Dostupné z: <https://people.eecs.berkeley.edu/~wkahan/ieee754status/IEEE754.PDF>.

[41] Zavacký, M. Nelineárne dynamické systémy – riešenie a vizualizácia. Diplomová práca, Technická univerzita v Košiciach, Košice, 2011. 66 s.

Zoznam príloh

Príloha A Výpočet systému troch rovníc metódou Runge-Kutta

Príloha B Výpočet trajektórie v C++

Príloha C Používateľská príručka

Príloha D Systémová príručka

Príloha E Článok publikovaný na IEEE konferencii Radioelektronika, Brno, apríl 2017

Príloha F Článok na recenzii IEEE konferencie TSP, Barcelona, júl, 2017

Príloha G CD médium – záverečná práca v elektronickej podobe, zdrojové kódy vytvorených aplikácií, parametre výpočtov, nástroje k prekladu.

Príloha A – Výpočet systému troch rovníc metódou Runge-Kutta

Táto príloha uvádza výpočet systému troch diferenciálnych rovníc metódou Runge-Kutta. Nahradením funkcií fx, fy, fz je uvedený postup priamo aplikovateľný pre výpočet trajektórie Chuaovho obvodu.

$$
\begin{aligned}
a_x &= fx(x_n, y_n, z_n) \\
a_y &= fy(x_n, y_n, z_n) \\
a_z &= fz(x_n, y_n, z_n)
\end{aligned}
$$

$$
\begin{aligned}
b_x &= fx(x_n + \frac{h}{2}a_x, y_n + \frac{h}{2}a_y, z_n + \frac{h}{2}a_z) \\
b_y &= fy(x_n + \frac{h}{2}a_x, y_n + \frac{h}{2}a_y, z_n + \frac{h}{2}a_z) \\
b_z &= fz(x_n + \frac{h}{2}a_x, y_n + \frac{h}{2}a_y, z_n + \frac{h}{2}a_z)
\end{aligned}
$$

$$
\begin{aligned}
c_x &= fx(x_n + \frac{h}{2}b_x, y_n + \frac{h}{2}b_y, z_n + \frac{h}{2}b_z) \\
c_y &= fy(x_n + \frac{h}{2}b_x, y_n + \frac{h}{2}b_y, z_n + \frac{h}{2}b_z) \\
c_z &= fz(x_n + \frac{h}{2}b_x, y_n + \frac{h}{2}b_y, z_n + \frac{h}{2}b_z)
\end{aligned}
$$

$$
\begin{aligned}
d_x &= fx(x_n + \frac{h}{2}c_x, y_n + \frac{h}{2}c_y, z_n + \frac{h}{2}c_z) \\
d_y &= fy(x_n + \frac{h}{2}c_x, y_n + \frac{h}{2}c_y, z_n + \frac{h}{2}c_z) \\
d_z &= fz(x_n + \frac{h}{2}c_x, y_n + \frac{h}{2}c_y, z_n + \frac{h}{2}c_z)
\end{aligned}
$$

$$
\begin{aligned}
x_{n+1} &= x_n + h\frac{a_x + 2b_x + 2c_x + d_x}{6} \\
y_{n+1} &= y_n + h\frac{a_y + 2b_y + 2c_y + d_y}{6} \\
z_{n+1} &= z_n + h\frac{a_z + 2b_z + 2c_z + d_z}{6}
\end{aligned}
$$

Príloha B – Výpočet trajektórie v C++

V tejto prílohe uvádzame okomentovaný úryvok programového kódu v jazyku C++, z triedy `TrajectoryCalculator` aplikácie *Chuaviz*. Kód je určený na výpočet trajektórie ZB metódou Runge-Kutta.

V kóde sú použité nasledujúce pomocné konštrukcie:

- `Point3DT` – údajová štruktúra, ktorá uchováva tri priestorové súradnice bodu a čas.
- `Trajectory` – trieda, ktorá reprezentuje trajektóriu ZB. Uchováva pole bodov trajektórie, a počet delení, ku ktorému došlo počas výpočtu.

Funkcia `calculateTrajectory` v úryvku uvedenom nižšie je zodpovedná za výpočet trajektórie. Argumenty `i0`, `u1_0` a `u2_0` reprezentujú začiatočné podmienky výpočtu. Pomocou argumentu `saveNth` je možné špecifikovať, každý koľkáty bod chceme do výsledku uložiť. Argument `maxPoints` slúži na obmedzenie maximálneho počtu bodov uložených do výsledku. Výsledkom funkcie je smerník na nový objekt triedy `Trajectory`.

```
Trajectory* TrajectoryCalculator::calculateTrajectory(
    double i0, double u1_0, double u2_0, int saveNth, int pMax)
{
    // Inicializácia výsledneho poľa
    std::vector<Point3DT>* points = new std::vector<Point3DT>();

    // Deklarácie i, u1, u2 a nastavenie ich začiatočných hodnôt
    double i = i0;
    double u1 = u1_0;
    double u2 = u2_0;

    // Nastavenie začiatočného integračného kroku
    double h = h0;
```

```
int divisionCount = 0;
int pointCount = 0;
int savedPointCount = 0;

// Uloženie začiatocného bodu trajektórie
points->push_back(Point3DT(i0, u1_0, u2_0, 0));

/* Cyklus je vykonaný až kým čas t nepresiahne maximálny čas
   výpočtu a počet uložených bodov hodnotu pMax */
for (double t = h;
            t <= tMax && savedPointCount <= pMax;
            t += h)
{
            // Výpočet metódou Runge-Kutta
    double a_i = fi(u2, i);
    double a_u1 = fu1(u1, u2, i);
    double a_u2 = fu2(u1, u2, i);
    double hdiv2 = 0.5 * h;
    double hdiv6 = h/6;

    double b_i = fi(u2 + hdiv2 * a_u2, i + hdiv2 * a_i);
    double b_u1 = fu1(u1 + hdiv2 * a_u1,
        u2 + hdiv2 * a_u2, i + hdiv2 * a_i);
    double b_u2 = fu2(u1 + hdiv2 * a_u1,
        u2 + hdiv2 * a_u2, i + hdiv2 * a_i);

    double c_i = fi(u2 + hdiv2* b_u2, i + hdiv2 * b_i);
    double c_u1 = fu1(u1 + hdiv2 * b_u1,
        u2 + hdiv2 * b_u2, i + hdiv2 * b_i);
    double c_u2 = fu2(u1 + hdiv2 * b_u1,
        u2 + hdiv2 * b_u2, i + hdiv2 * b_i);

    double d_i = fi(u2 + h * c_u2, i + h * c_i);
    double d_u1 = fu1(u1 + h * c_u1, u2 + h * c_u2, i + h * c_i);
```

```
        double d_u2 = fu2(u1 + h * c_u1, u2 + h * c_u2, i + h * c_i);

        // Prírastok uložíme do premenných iDiff , u1Diff , u2Diff
        double iDiff = hdiv6*(a_i + 2 * b_i + 2 * c_i + d_i);
        double u1Diff = hdiv6*(a_u1 + 2 * b_u1 + 2 * c_u1 + d_u1);
        double u2Diff = hdiv6*(a_u2 + 2 * b_u2 + 2 * c_u2 + d_u2);

        // Presahuje prírastok maximálne povolenú hodnotu?
        if(abs(iDiff) > ihMax ||
           abs(u1Diff) > uhMax ||
           abs(u2Diff) > uhMax)
        {
            // Ak áno, odčítame z času integračný krok
            t = t - h;
            // Integračný krok predelíme číslom 2
            h = h / 2;
            // Zaznamenáme delenie integračného kroku
            ++divisionCount;
        }else{
            // Ak nie, pripočítame prírastky k i, u1, u2
            i = i + iDiff;
            u1 = u1 + u1Diff;
            u2 = u2 + u2Diff;

            // Integračný krok vynasobíme číslom n
            h = h * n;

            /* Ak je aktuálny bod "n-tý", vypočítané súradnice
               uložíme do poľa points  */
            if(pointCount % saveNth == 0){
                points->push_back(Point3DT(i, u1, u2, t));
                ++savedPointCount;
            }
            ++pointCount;
        }
```

```
    }

    return new Trajectory(points, divisionCount);
}
// Funkcia Chuaovej diódy - 5 segmentová
double fu1(double u1, double u2, double i) {
    return ((u2 - u1) / R - (m2 * u1 + 0.5 * (m1 - m0)*(abs(u1 - Bp)
    - abs(u1 + Bp)) + 0.5 * (m2 - m1)*(abs(u1 - B0) - abs(u1 + B0)))
    - I) / C1;
}

// Pomocná funkcia pre výpočet Runge-Kutta
double fu2(double u1, double u2, double i) {
    return ((u1 - u2) / R + i) / C2;
}

// Pomocná funkcia pre výpočet Runge-Kutta
double fi(double u2, double i) {
    return (-u2 - ro * i) / L;
}
```

Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky

Paralelné výpočty pre spracovanie a vizualizáciu stavového priestoru nelineárnych obvodov

Príloha C

Používateľská príručka programu Chuaviz

Diplomová práca

2017 **Zsolt Rácz**

Inštalácia programu

V nasledujúcej časti sa nachádzajú inštalačné inštrukcie pre aplikácie *Chuaviz*. Aplikácia je distribuovaná pre 32 a 64 bitové verzie operačných systémov Windows a Linux. Pre iné platformy aleob CPU architektúry je potrebné aplikáciu preložiť zo zdrojového kódu. Inštrukcie pre preklad sa nachádzajú v systémovej príručke.

Hardvérové požiadavky

Pre korektný beh aplikácie je potrebné, aby boli nasledujúce minimálne hardvérové požiadavky splnené:

- Procesor (CPU) – Intel Pentium 4, alebo novší, s frekvenciou 1,3 GHz a vyššou (alebo ekvivalent);
- Veľkosť systémovej pamäti (RAM) – 1 GB alebo viac;
- Voľný priestor na úložisku aspoň 100 MB;
- Obrazovka s minimálnym rozlíšením 1440x900;
- Grafická karta s podporou OpenGL 2.0.

Softvérové požiadavky

Aplikácia *Chuaviz* je distribuovaná pre nasledujúce operačné systémy:

- Windows – verzia 7 a vyššie;
- Linux – verzia kernelu 2.6 a vyššie, 64 bitová architektúra.

Ďalej je potrebné, aby operačný systém bežal v grafickom móde, s rozlíšením aspoň 1440×900. Pri nižších rozlíšeniach môže byť okno aplikácie orezané. Pre využitie možností 3D vizualizácie je potrebné mať nainštalované grafické ovládače. Ostatné softvérové závislosti sú distribuované spolu s aplikáciou.

Na operačnom systéme Linux je potrebné mať nainštalované zdieľané knižnice *Qt* verzie aspoň 5.7. Ak distribúcia poskytuje iba staršie verzie, je ich potrebné nainštalovať ručne. Detailný manuál inštalácie je uvedený na stránke: `https://wiki.qt.io/Install_Qt_5_on_Ubuntu`.

Inštalácia aplikácie

Inštalácia *Chuaviz* prebieha ručným skopírovaním distribuovanej aplikácie z CD do počítača. Spustiteľné verzie aplikácie je možné nájsť v priečinku `chuaviz/dist`, resp. v `chuaviz-de/dist` pre verziu DE. Tieto priečinky obsahujú verzie pre rôzne platformy: Windows 32 bitová verzia, Windows 64 bitová verzia a Linux 64 bitová verzia. Dôležité je, aby pri inštalácii boli prekopírované všetky súbory nachádzajúce sa v tomto priečinku.

Na operačnom systéme Linux je potrebné pre hlavný súbor aplikácie (`chuaviz`) nastaviť oprávnenie *execute*. Je to možné docieliť vykonaním príkazu `chmod +x chuaviz` v priečinku nainštalovanej aplikácie.

Spúšťanie aplikácie

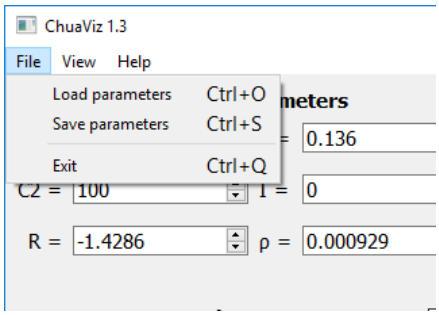
Spúšťanie aplikácie *Chuaviz* nevyžaduje žiadne špeciálne kroky. Aplikáciu stačí spustiť dvojklikom na spustiteľný súbor (`chuaviz.exe` alebo `chuaviz`) z používateľského rozhrania alebo z príkazového riadku bez žiadnych parametrov. Pred spustením je dôležité sa uistiť, že v priečinku aplikácie sa nachádza súbor `parameters.txt`.

Použitie aplikácie

Funkcionalita používateľského rozhrania a možnosti aplikácie *Chuaviz* sú detailne uvedené v hlavnej časti práce. V tejto časti uvádzame teda doplnok k už predstavenému možnostiam.

Menu aplikácie

Globálne akcie v aplikácii sú dostupné cez aplikačné menu (alebo hlavnú ponuku). Ide o štandardný prvok používateľského rozhrania, ktoré sa nachádza v pravej hornej časti okna. Menu je ilustrované na Obr. C – 1.



Obr. C – 1: Hlavná ponuka aplikácie *Chuaviz*

V menu sú dostupné nasledujúce možnosti:

- *File* – Súbor
  - *Load parameters* – po zvolení položky je zobrazené dialógové okno na vyhľadanie súboru so vstupnými parametrami.
  - *Save parameters* – umožňuje ukladať stav okna (parametre) do súboru.
  - *Exit* – zatvorí okno a ukončí beh aplikácie.
- *View* – Pohľad

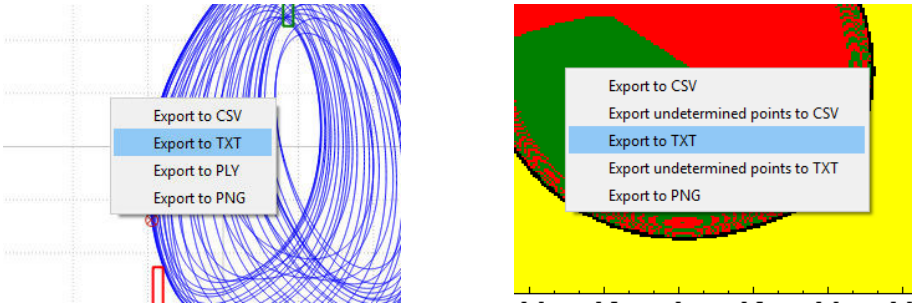
- *Trajectory* – prepnutie na pohľad s výpočtom trajektórie.
- *Cross-section* – prepnutie na pohľad s výpočtom jedného rezu HP.
- *CS serie* – prepnutie na pohľad s výpočtom série rezov HP.

- *Help* – Pomoc. Obsahuje jedinú položku *About*, ktorá poskytuje informácie o aplikácii a autorovi.

Exportovanie údajov

Výsledky výpočtov je možné preniesť do iných programov pomocou možnosti exportovania. Exportovať je možné vypočítanú trajektóriu, alebo rez HP. Po kliknutí pravým tlačidlom myši na 2D projekciu rezu, zobrazí sa kontextové menu, ako to je ilustrované na Obr. C – 2a. Poskytnuté sú možnosti: CSV, TXT, PLY alebo PNG. Po vybraní niektorej z možností sa zobrazí dialógové okno pre výber cesty pre ukladanie súboru.

Podobne to je aj pri reze HP, ktoré je zobrazené na Obr. C – 2b. Poskytnuté sú možnosti exportovania do CSV, TXT a PNG súboru, avšak pri vybraní voľby *„Export undetermined points“* sa ukladajú iba tie body rezu HP, v ktorých nebol atraktor identifikovaný.



(a) Exportovanie trajektórie (b) Exportovanie rezu HP

Obr. C – 2: Možnosti exportovania údajov

Klávesové skratky

Aplikácia poskytuje nasledujúce klávesové skratky:

- `Ctrl+1` – prepnúť na výpočet trajektórie ZB *(Menu → View → Trajectory)*;
- `Ctrl+2` – prepnúť na výpočet rezu HP *(Menu → View → Cross-section)*;
- `Ctrl+3` – prepnúť na výpočet série rezov HP *(Menu → View → CS serie)*;
- `Ctrl+O` – načítať parametre zo súboru *(Menu → File → Load parameters)*;
- `Ctrl+S` – uložiť parametre do súboru *(Menu → File → Save parameters)*;
- `Ctrl+Q` – ukončiť program*(Menu → File → Exit).*

Vstupné a výstupné súbory

Aplikácia pracuje s rôznymi textovými súbormi za účelom načítania a ukladania výpočtových parametrov, resp. vypočítaných výsledkov. V tejto časti príručky predstavíme ich formáty.

Parameters.txt

Je umožnené ukladanie parametrov výpočtu, zadaných v používateľskom rozhraní. Formát súboru je nasledovný:

C_1	C_2	L	ρ	m_0	m_1		i_{zp}		u_{2zp}	u_{1zp}
m_2	I	B_P	B_0	t_{max}	$uhMax$	$ihMax$		t_{test}	n	nth	p_{max}

kde jednotlivé čísla reprezentujú hodnoty parametrov výpočtu, ktoré sú zobrazené v pohľade pre výpočet trajektórie. Ďalšie riadky môžu byť v súbore prítomné,

z ktorých sa načítajú intervaly výpočtu rezu HP, resp. série rezov HP a testovacie podmienky pre detekciu typu trajektórie.

Trajektória vo formáte CSV a TXT

Pri ukladaní trajektórie vo formáte CSV, ako aj pri načítaní trajektórie v okne s 3D zobrazením je formát riadku nasledovný:

$$i, \quad u_2, \quad u_1, \quad t$$

kde:

i, u_2 a u_1	sú súradnice ZB v 3D priestore,
t	je integračný čas pre ktorý bol ZB určený.

Formát riadku v TXT súbore je podobný formátu CSV, avšak ako oddelovač hodnôt namiesto čiarky je použitý tabulátor.

Trajektória vo formáte PLY

Je ukladaný v štandardnom formáte PLY, používa sa verzia 1.0 a ASCII (textový) zápis. Do súboru sú uložené všetky vypočítané 3D body ako vrcholy (vertex), bez určenia farby.

Rez HP vo formáte CSV a TXT

Aplikácia umožňuje ukladanie rezov HP v textovom formáte CSV alebo TXT. Rozdiel medzi dvoma formátmi je v oddelovači hodnôt – pri CSV sa používa čiarka, kým v TXT tabulátor. Výstupný súbor obsahuje M×N počet riadkov, kým M reprezentuje počet stĺpcov rezu a N počet riadkov. Jeden riadok reprezentuje jeden bod HP, teda jednu vypočítanú trajektóriu. Formát ukladaného riadku je nasledovný (CSV):

u_2, u_1, i, *result* *test_idx* t, *division_count*

kde:

i, u_2 a u_1	sú začiatočné podmienky,
result	je typ trajektórie,
test_idx	je poradové číslo vyhovenej podmienky,
t	je integračný čas kedy bol výpočet trajektórie ukončený,
division_count	je počet delení integračného kroku.

Možné hodnoty štvrtého stĺpca *(result)* sú: `0` – SLC, `1` – CHA a `6` – nezistený. V piatom stĺpci *(test_idx)* sú podmienky číslované od 1 a hodnota `0` znamená nezistený typ. Testovacie podmienky nie sú ukladané do výstupného súboru rezu HP, iba v súbore vstupných parametrov. Preto pre uchovanie plnohodnotnosti údajov je dôležité uchovať aj tento súbor.

Neskonvergované body v TXT a CSV

Pre ukladanie zoznamu neskonvergovaných bodov sa používa rovnaký formát riadku, ako pri ukladaní celého rezu. Hodnota *result* má v tomto prípade v každom riadku hodnotu `6` a *test_idx* hodnotu `0`.

Info.txt

Súbor `info.txt` je textový súbor, ktorý sa vytvára automaticky v priečinku zvolenom pre výpočet série rezov HP. Formát súboru je nasledovný:

type
columnMin, *columnMax*, *columnCount*
rowMin, *rowMax*, *rowCount*
depthMin, *depthMax*, *depthCount*

kde:

type	je typ rezu (0 – i,u_1, 1 – i,u_2, 2 – u_2, u_1),
columnMin, *Max* a *Count*	je rozsah a počet stĺpcov (x-ová os),
rowMin, *Max* a *Count*	je rozsah a počet riadkov (y-ová os),
depthMin, *Max* a *Count*	je rozsah a počet rezov (z-ová os).

Chybové hlásenia

Aplikácia pri vyskytnutí problémoch môže vypísať chybové hlásenia. Najčastejšie vyskytujúce sa hlásenia sú popísané v tejto časti.

Cannot load parameters file

Chybová správa znamená, že aplikácia nenašla v jej priečinku súbor s názvom `parameters.txt`. Tento súbor slúži na načítanie predvolených parametrov výpočtu a musí existovať v priečinku aplikácie.

This program can't start because MSVCP120.dll is missing from your computer.

Správa sa môže vyskytnúť na systémoch Windows. Do priečinku aplikácie je potrebné pridať súbor `msvcp120.dll`, ktorý je dodávaný spolu s aplikáciou, alebo musia byť nainštalované distribuovateľné balíčky jazyka Visual C++.

Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky

Paralelné výpočty pre spracovanie a vizualizáciu stavového priestoru nelineárnych obvodov

Príloha D
Systémová príručka

Diplomová práca

2017 **Zsolt Rácz**

Úvod

Systémová príručka aplikácie *Chuaviz* obsahuje návod na prekladanie aplikácie pod systémom Windows a Linux. Ďalej uvádza architektúru aplikácie pomocou vývojových diagramov a popisom navrhnutých tried.

Preklad aplikácie

Aplikácia *Chuaviz* bola vyvíjaná v programovacom jazyku C++ s využitím frameworku *Qt*. Keďže sa jedná o prekladaný jazyk, pred spustením aplikácie je ju potrebné vhodnými nástrojmi preložiť. V tejto časti uvedieme postup prekladu aplikácie na platformách Windows a Linux.

Požiadavky na technické prostriedky pri preklade

Pre bezproblémový preklad aplikácie je odporúčané používať hardvér, ktorý spĺňa nasledujúce nároky:

- Procesor (CPU) – radu Pentium 4 a vyššie;
- Veľkosť systémovej pamäti (RAM) – 1 GB alebo viac;
- Voľný priestor na úložisku aspoň 500 MB;

Preklad pod systémom Windows

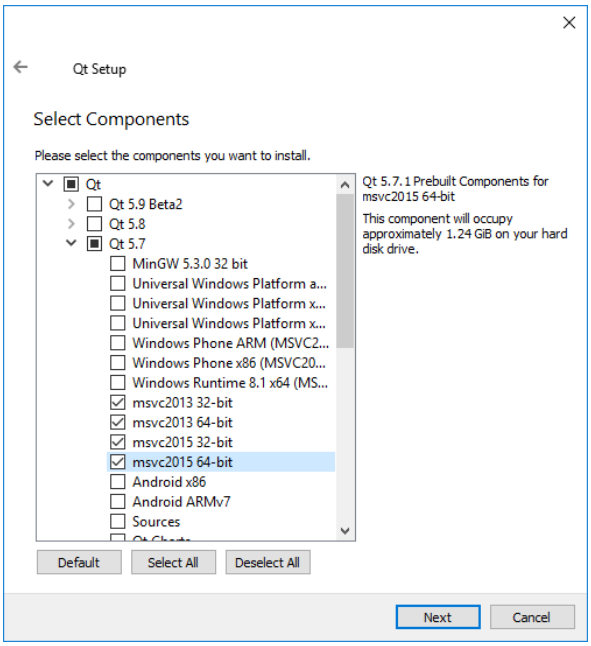
Pri vývoji bolo použité integrované vývojové prostredie *QtCreator*, ktoré umožňuje spustiť preklad priamo z užívateľského rozhrania. K prekladu však je najprv potrebné nainštalovať prekladač, keďže *QtCreator* ho neposkytuje.

Krok 1 – Inštalácia prekladača

Program *QtCreator* podporuje množstvo prekladačov, avšak pri vývoji aplikácie *Chuaviz* bol použitý Microsoft Visual Studio 2013. Je odporúčané teda jeho využitie. Prekladač je dostupný z nasledujúcej stránky: `https://www.visualstudio.com/downloads/`, pričom za účelom prekladu aplikácie postačuje základná verzia *Community*, ktorá je k dispozícii zadarmo.

Krok 2 – Inštalácia *Qt*

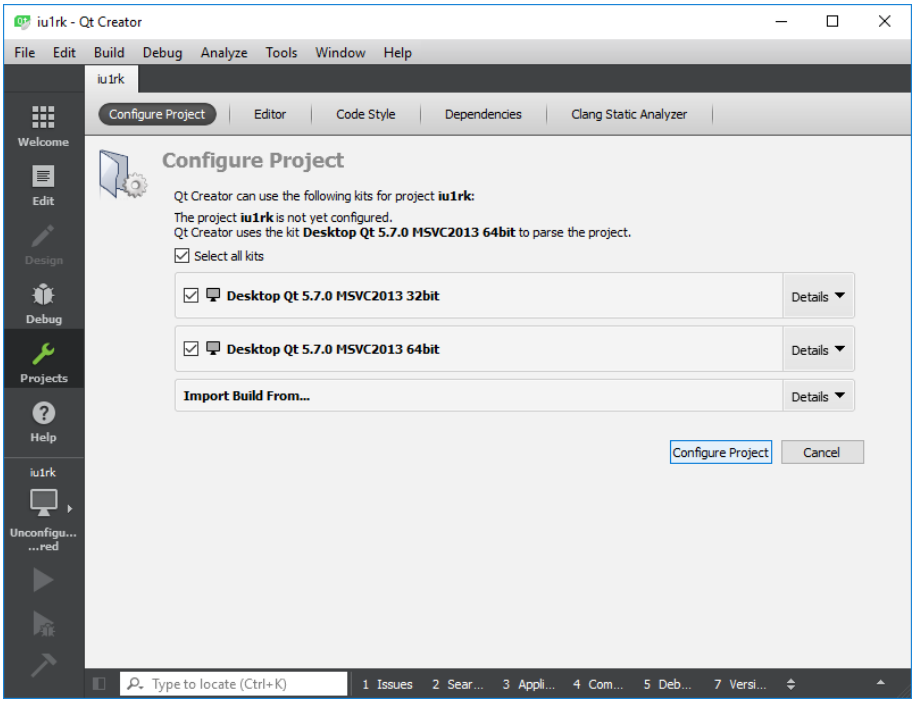
Ďalším krokom po inštalácii prekladača je inštalácia knižníc *Qt* spolu s programom *QtCreator*. V CD prílohe G v priečinku `tools` je pripojený grafický inštalátor, pomocou ktorého je možné tento proces uskutočniť. Po spustení súboru `qt-unified-online.exe` sa objaví grafický sprievodca. V kroku výberu inštalovaných komponentov, ktorý je zobrazený na Obr. D – 1 je potrebné vybrať *Qt 5.7* a podporu pre nainštalovaný prekladač. V našom prípade to je *msvc2013*.



Obr. D – 1: Výber komponentov pri inštalácii *Qt*

Krok 3 – Výber prekladača v *QtCreator*

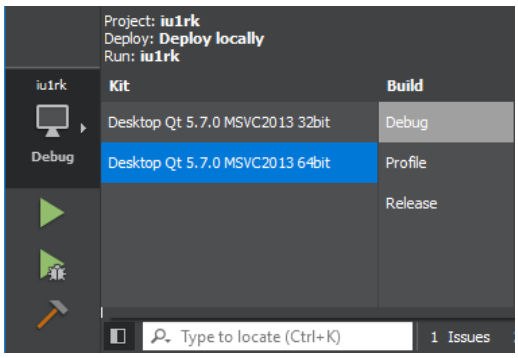
Po úspešnej inštalácii frameworku *Qt* a programu *QtCreator* je možné otvoriť projekt. Zdrojové súbory pre aplikáciu *Chuaviz* sa nachádzajú v priečinku `chuaviz/src` a pre aplikáciu *Chuaviz DE* v `chuaviz-de/src`. V priečinku je potrebné nájsť súbor s rozšírením `.pro` a otvoriť ho v *QtCreator*. Mali by sme vidieť podobné okno, ako je zobrazené na Obr. D – 2. V tomto kroku je potrebné si vybrať prekladač, resp. architektúru, ktorú si prajeme použiť pri preklade.



Obr. D – 2: Výber prekladača v programe *QtCreator*

Krok 4 – Spustenie prekladu

Ovládacie tlačidlá pre spustenie prekladu sa nachádzajú v ľavom dolnom rohu okna programu *QtCreator* (viď. Obr. D – 3). Klikom na zelené tlačidlo sa spustí preklad aplikácie.



Obr. D – 3: Spustenie prekladu v programe *QtCreator*

Preklad pod systémom Linux

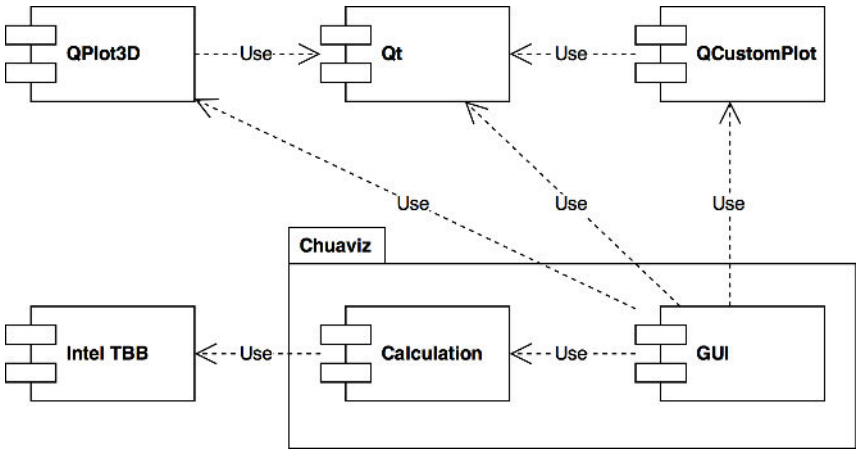
Preklad na systéme Linux je možné spustiť podobným spôsobom. Grafický inštalačný sprievodca je umiestnený na ceste `tools/qt-unified-online.run` v CD prílohe G. Pri preklade pod systémom Linux bol používaný prekladač GCC.

Popis riešenia

Aplikácie *Chuaviz* bola logicky rozdelená na dve hlavné moduly: *Calculation* a *GUI*. Kým prvý zahrňuje všetky poskytované výpočty, v druhom je implementované grafické rozhranie. Ich vzájomný vzťah, resp. vzťah k iným použitým softvérovým knižniciam je znázornený na Obr. D – 4.

Popis tried, algoritmov a údajových štruktúr

Najpodstatnejšou časťou aplikácie je modul *Calculation*, v ktorom sú zahrnuté všetky výpočty, ktoré aplikácia poskytuje. Je to nezávislý komponent využiteľný aj v iných projektoch. Diagram tried modulu *Calculation* je uvedený na Obr. D – 5. Popis jednotlivých tried je uvedený v nasledujúcich častiach.



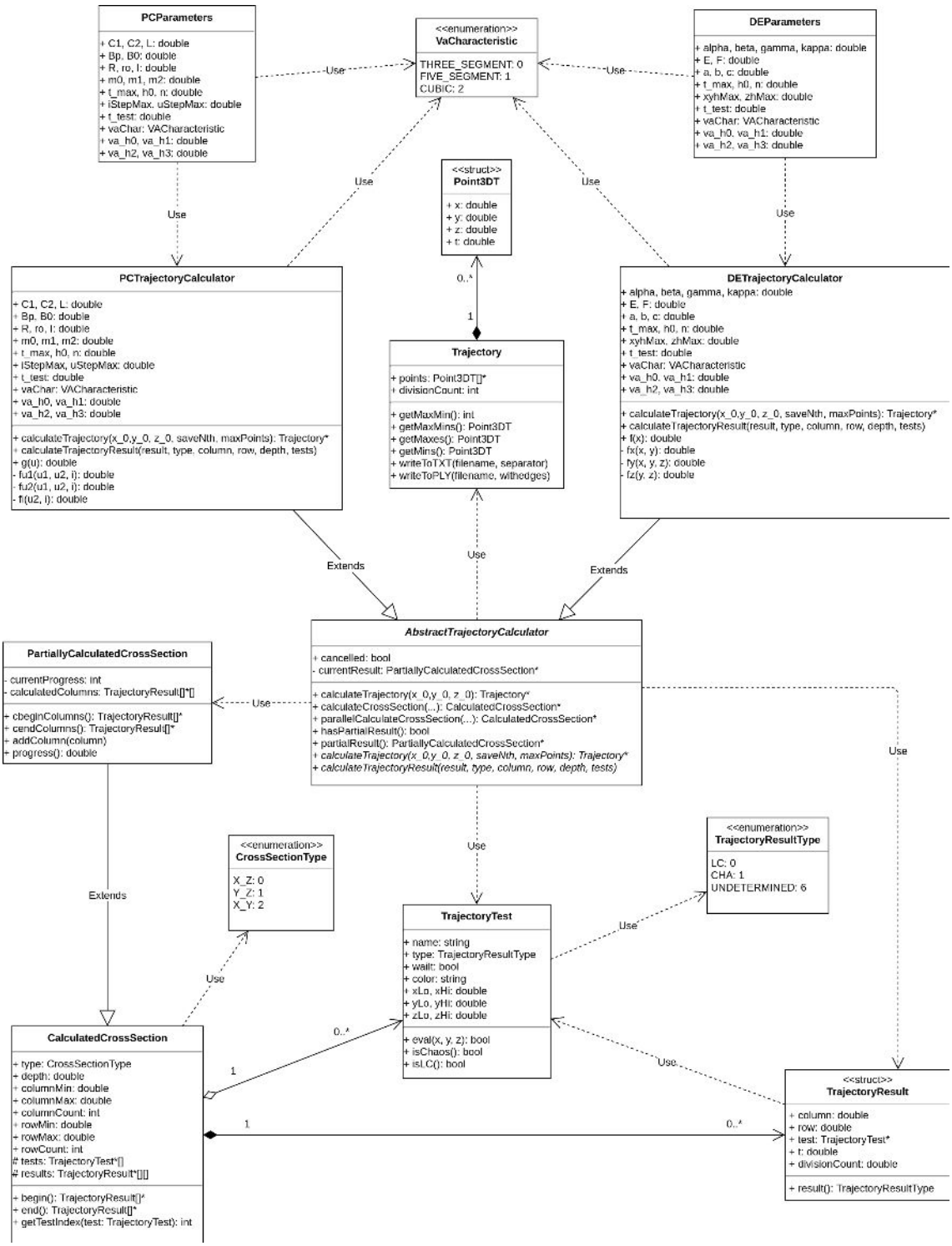
Obr. D – 4: Diagram závislostí a komponentov

Trieda AbstractTrajectoryCalculator

Abstraktná trieda, ktorá je jadrom modulu *Calculation*. Slúži ako základ pre výpočet HP. Obsahuje dve abstraktné metódy, ktoré musia byť implementované v potomkoch.

Atribúty:

Názov	**Typ**	**Viditeľnosť**
cancelled	`bool`	`public`
Označuje, či bol aktuálny výpočet zrušený.		
currentResult	`PartiallyCalCulatedCross-Section*`	`private`
Uchováva smerník na rez HP, ktorý sa aktuálne počíta. Pred výpočtom a po výpočte má hodnotu `NULL`.		



Obr. D – 5: Diagram tried modulu *Calculation*

Metódy:

Názov	**Návratový typ**	**Viditeľnosť**
calculateCrossSection	`CalculatedCrossSection*`	`public`
Je určená pre výpočet jedného rezu HP. Argumentmi metódy sú rozsahy, v ktorom sa má rez počítať a množina testovacích podmienok.		
parallelCalculateCrossSection	`CalculatedCrossSection*`	`public`
Vykonáva paralelný výpočet jedného rezu HP, pričom použije všetky jadrá CPU. Argumentmi metódy sú rozsahy, v ktorom sa má rez počítať a množina testovacích podmienok.		
hasPartialResult	`bool`	`public`
Ak je k dispozícii priebežný výsledok, metóda vráti hodnotu `true`.		
partialResult	`PartiallyCalculated-CrossSection*`	`public`
Ak je k dispozícii priebežný výsledok, metóda vráti naňho smerník, v opačnom prípade vráti `NULL`.		
calculateTrajectory	`Trajectory*`	`public`
Abstraktná metóda pre vykonanie výpočtu trajektórie.		
calculateTrajectoryResult	`void`	`public`
Abstraktná metóda pre zistenie typu trajektórie.		

Enumeračný typ VACharacteristic

Tento enumeračný typ reprezentuje typy VA charakteristiky, resp. funkcie $f(x)$ diódy. Možné hodnoty:

- `THREE_SEGMENT` – 3-segmentový typ funkcie;
- `FIVE_SEGMENT` – 5-segmentový typ funkcie;

- `CUBIC` – kubický typ funkcie.

Štruktúra Point3DT

Údajová štruktúra, ktorá uchováva polohu ZB v istom časovom okamihu. Obsahuje verejné atribúry x, y a z typu `double`, ktoré udávajú polohu bodu, a t, ktorý reprezentuje čas v ms. Pri fyzikálnom obvode $x \Leftrightarrow u_2$, $y \Leftrightarrow u_1$ a $z \Leftrightarrow i$.

Trieda DEParameters

Reprezentuje množinu parametrov pre výpočet bezrozmerného modelu Chuaovho obvodu.

Atribúty:

Názov	**Typ**	**Viditeľnosť**
alpha, beta, gamma, kappa	`const double`	`public`
Parametre $\alpha, \beta, \gamma, \kappa$ bezrozmerného modelu obvodu.		
t_max	`const double`	`public`
Maximálny čas simulácie.		
h0	`const double`	`public`
Udáva veľkosť počiatočného integračného kroku.		
xyhMax	`const double`	`public`
Maximálny odskok nasledujúceho bodu na osiach x a y pri simulácii trajektórie.		
zhMax	`const double`	`public`
Maximálny odskok nasledujúceho bodu na osi z pri simulácii trajektórie.		
n	`const double`	`public`

Číslo s ktorým je po každom vypočítanom bode integračný krok násobený.		
t_test	`const double`	`public`
Integračný čas, do ktorého má simulácia trajektórie bežať bez aplikovania testovacích podmienok pre CHA, pri výpočte rezov HP.		
vaChar	`const VACharacteristic`	`public`
Typ funkcie $f(x)$, s ktorým sa má počítať.		
E	`const double`	`public`
Parameter E 5 a 3 segmentovej funkcie diódy.		
F	`const double`	`public`
Parameter F 5 segmentovej funkcie diódy.		
a, b	`const double`	`public`
Parametre a, b 5 a 3 segmentovej funkcie diódy.		
c	`const double`	`public`
Parameter c 5 segmentovej funkcie diódy.		
va_h0, va_h1, va_h2, va_h3	`const double`	`public`
Parametre kubickej funkcie $f(x)$.		

Trieda DETrajectoryCalculator

Potomok triedy `AbstractTrajectoryCalculator`, ktorý je určený pre počítanie trajektórií a HP bezrozmerného modelu Chuaovho obvodu. Pri inštancovaní triedy sú odovzdané parametre výpočtu, ktoré nie je možné následne zmeniť.

Atribúty:

Obsahuje rovnaké atribúty, ako trieda `DEParameters`. Tie reprezentujú parametre obvodu. Pri vytváraní inštancie `DETrajectoryCalculator` sú hodnoty paramet-

rov skopírované do nového objektu.

Metódy:

Názov	**Návratový typ**	**Viditeľnosť**
calculateTrajectory	`Trajectory*`	`public`
Implementácia abstraktnej metódy z `AbstractTrajectoryCalculator` pre bezrozmerné modely obvodov.		
calculateTrajectoryResult	`void`	`public`
Implementácia abstraktnej metódy z `AbstractTrajectoryCalculator` pre bezrozmerné modely obvodov.		
f	`double`	`public`
Implementuje funkciu $f(x)$ diódy.		
fx	`double`	`private`
Pomocná funkcia pri výpočte Runge-Kutta – vyjadruje dx z prvej rovnice v (2.2).		
fy	`double`	`private`
Pomocná funkcia pri výpočte Runge-Kutta – vyjadruje dy z druhej rovnice v (2.2).		
fz	`double`	`private`
Pomocná funkcia pri výpočte Runge-Kutta – vyjadruje dz z tretej rovnice v (2.2).		

Trieda PCParameters

Reprezentuje množinu parametrov pre výpočet fyzikálneho modelu Chuaovho obvodu.

Atribúty:

Názov	**Typ**	**Viditeľnosť**
C1, C2, L, R, ro, I	`const double`	`public`
Parametre C_1, C_2, L, R, ρ, I fyzikálneho modelu obvodu.		
t_max	`const double`	`public`
Maximálny čas simulácie.		
h0	`const double`	`public`
Udáva veľkosť počiatočného integračného kroku.		
uStepMax	`const double`	`public`
Maximálny odskok nasledujúceho bodu na osiach u_1 a u_2 pri simulácii trajektórie.		
iStepMax	`const double`	`public`
Maximálny odskok nasledujúceho bodu na osi i pri simulácii trajektórie.		
n	`const double`	`public`
Číslo s ktorým je po každom vypočítanom bode integračný krok násobený.		
t_test	`const double`	`public`
Integračný čas, do ktorého má simulácia trajektórie bežať bez aplikovania testovacích podmienok pre CHA, pri výpočte rezov HP.		
vaChar	`const VACharacteristic`	`public`
Typ VA charakteristiky Chuaovej diódy, s ktorým sa má počítať.		
Bp	`const double`	`public`
Parameter B_p 5 a 3 segmentovej funkcie diódy.		
B0	`const double`	`public`
Parameter B_0 5 segmentovej funkcie diódy.		
m0, m1	`const double`	`public`
Parametre m_0, m_1 5 a 3 segmentovej funkcie diódy.		
m2	`const double`	`public`
Parameter m_2 5 segmentovej funkcie diódy.		

va_h0, va_h1, va_h2, va_h3	`const double`	`public`
Parametre kubického typu VA charakteristiky.		

Trieda PCTrajectoryCalculator

Potomok triedy `AbstractTrajectoryCalculator`, ktorý je určený pre počítanie trajektórií a HP fyzického modelu Chuaovho obvodu. Pri inštancovaní triedy sú odovzdané parametre výpočtu, ktoré nie je možné následne zmeniť.

Atribúty:

Obsahuje rovnaké atribúty, ako trieda `PCParameters`. Tie reprezentujú fyzikálne parametre obvodu a parametre výpočtu. Pri vytváraní inštancie `PCTrajectory-Calculator` sú parametre výpočtu skopírované do novovytvoreného objektu.

Metódy:

Názov	**Návratový typ**	**Viditeľnosť**
calculateTrajectory	`Trajectory*`	`public`
Implementácia abstraktnej metódy z `AbstractTrajectoryCalculator` pre fyzikálne modely obvodov.		
calculateTrajectoryResult	`void`	`public`
Implementácia abstraktnej metódy z `AbstractTrajectoryCalculator` pre fyzikálne modely obvodov.		
g	`double`	`public`
Implementuje VA charakteristiku Chuaovej diódy.		

fu1	`double`	`private`
Pomocná funkcia pri výpočte Runge-Kutta – vyjadruje du_1 z prvej rovnice v (2.1).		
fu2	`double`	`private`
Pomocná funkcia pri výpočte Runge-Kutta – vyjadruje du_2 z druhej rovnice v (2.1).		
fi	`double`	`private`
Pomocná funkcia pri výpočte Runge-Kutta – vyjadruje *di* z tretej rovnice v (2.1).		

Trieda Trajectory

`Trajektory` reprezentuje jednu trajektóriu – výsledok výpočtu. Obsahuje pole vypočítaných bodov trajektórie, ktoré sú usporiadané podľa času t.

Atribúty:

Názov	**Typ**	**Viditeľnosť**
points	`vector<Point3DT>*`	`public`
Smerník na pole vypočítaných bodov trajektórie.		
divisionCount	`int`	`public`
Reprezentuje počet delení, ku ktorým došlo pri výpočte trajektórie v dôsledku parametrov $iStepMax$ a $uStepMax$, resp. $xyhMax$ a $zhMax$.		

Metódy:

Názov	**Návratový typ**	**Viditeľnosť**
getMaxMin	`int`	`public`
Vracia najvyššiu absolútnu hodnotu súradníc ZB zo všetkých osí.		

getMaxMins	`Point3DT`	`public`
Vracia najvyššiu absolútnu hodnotu súradníc ZB pre všetky osi osobitne pomocou údajovej štruktúry `Point3DT`.		
getMaxes	`Point3DT`	`public`
Vracia najvyššiu súradníc ZB pre všetky osi osobitne pomocou údajovej štruktúry `Point3DT`.		
getMins	`Point3DT`	`public`
Vracia najnižšiu súradníc ZB pre všetky osi osobitne pomocou údajovej štruktúry `Point3DT`.		
writeToTXT	`void`	`public`
Vypíše súradnice bodov trajektórie do textového súboru. Argument *separator* udáva oddelovač medzi číslami v riadku.		
writeToPLY	`void`	`public`
Vypíše súradnice bodov trajektórie do súboru vo formáte PLY.		

Enumeračný typ TrajectoryResultType

Tento enumeračný typ reprezentuje typ trajektórie. Používa sa pri výpočte rezu HP. Možné hodnoty:

- `LC` – trajektória typu limitný cyklus;
- `CHA` – chaotický atraktor;
- `UNDETERMINED` – neidentifikovaný typ trajektórie.

Trieda TrajectoryTest

Reprezentuje testovaciu podmienku, pomocou ktorej sa identifikuje atraktor pri výpočte rezu HP. Medzi iných uchováva súradnice testovacieho hranola.

Atribúty:

Názov	**Typ**	**Viditeľnosť**
name	`string`	`public`
Názov testovacej podmienky.		
type	`string`	`public`
Typ testovacej podmienky. Podmienka `CHA` vyhovuje, keď sa ZB objaví vo vnútri testovacieho hranola, pri `LC` opačne – keď sa ZB objaví mimo hranola.		
wait	`bool`	`public`
Označuje, či má sa čakať do času $tTest$ pri vyhodnotení podmienok typu `CHA`. Ak je $false$, vyhodnocuje sa od času $t = 0$.		
color	`string`	`public`
Názov farby, akou sa majú v reze HP označovať ZP, pri ktorých podmienka vyhovela.		
xLo, xHi, yLo, yHi, zLo, zHi	`string`	`public`
Označujú hranice testovacieho hranola na jednotlivých priestorových osiach.		

Metódy:

Názov	**Návratový typ**	**Viditeľnosť**
eval	`bool`	`public`
Vyhodnotí podmienku pre ZB daný v argumentoch.		
isChaos	`bool`	`public`

Vracia `true`, ak podmienka je typu CHA.		
isLc	`bool`	`public`
Vracia `true`, ak podmienka je typu LC.		

Štruktúra TrajectoryResult

Táto údajová štruktúra je použitá pre uchovanie výsledku výpočtu jedného bodu v reze HP.

Atribúty:

Názov	**Typ**	**Viditeľnosť**
column	`double`	`public`
Súradnica ZP na osi, ktorému prislúcha x-ová os rezu (stĺpec).		
row	`double`	`public`
Súradnica ZP na osi, ktorému prislúcha z-ová os rezu (riadok).		
test	`TrajectoryTest*`	`public`
Smerník na testovaciu podmienku, ktorá identifikovala typ trajektórie. Ak nebola identifikovaná, táto hodnota je `NULL`.		
t	`double`	`public`
Integračný čas, pri ktorom bol typ trajektórie identifikovaný.		
divisionCount	`int`	`public`
Počet delení, ku ktorým došlo pri výpočte trajektórie v dôsledku parametrov $iStepMax$ a $uStepMax$, resp. $xyhMax$ a $zhMax$.		

Enumeračný typ CrossSectionType

`CrossSectionType` slúži na označenie roviny rezu HP. Možné hodnoty sú:

- `X_Z` – rovina i, u_1 pri fyzikálnom obvode a z, x pri bezrozmernom;
- `Y_Z` – rovina i, u_2, resp. z, y;
- `X_Y` – rovina u_2, u_1, resp. y, x.

Trieda CalculatedCrossSection

Trieda reprezentuje jeden rez HP. Uchováva dvojrozmerné pole s výsledkami výpočtu.

Atribúty:

Názov	**Typ**	**Viditeľnosť**
type	`CrossSectionType`	`public`
Reprezentuje typ (rovinu) rezu HP.		
columnMin, columnMax	`double`	`public`
Rozsah ZP na priestorovej osi reprezentovanou zvislou osou rezu.		
columnCount	`double`	`public`
Počet počítaných stĺpcov rezu HP.		
rowMin, rowMax	`double`	`public`
Rozsah ZP na priestorovej osi reprezentovanou vodorovnou osou rezu.		
rowCount	`double`	`public`
Počet počítaných riadkov rezu HP.		
tests	`vector<TrajectoryTest>*`	`protected`
Pole s testovacími podmienkami, s ktorými bol rez počítaný.		

results	`vector<vector<TrajectoryResult>>`	`protected`
Dvojrozmerné pole uchovávajúce výsledné body rezu HP.		

Metódy:

Názov	**Návratový typ**	**Viditeľnosť**
begin	`vector<vector<TrajectoryResult>>::iterator`	`public`
Vracia smerník (iterátor) na prvý prvok (stĺpec) poľa *results*.		
end	`vector<vector< TrajectoryResult>>::iterator`	`public`
Vracia smerník (iterátor) na adresu za posledným prvkom (stĺpcom) poľa *results*.		
getTestIndex	`int`	`public`
Zistí poradové číslo testu poskytnutého v argumente.		

Trieda PartiallyCalculatedCrossSection

Reprezentuje čiastočne vypočítaný rez HP. Uchováva jeho vypočítané stĺpce za účelom ich priebežného zobrazenia.

Atribúty:

Názov	**Typ**	**Viditeľnosť**
currentProgress	`int`	`public`
Uchováva aktuálny stav výpočtu v percentách.		
calculatedColumns	`list<vector<TrajectoryResult>*>`	`public`

Zoznam, ktorý uchováva smerníky na vypočítané stĺpce rezu HP.

Metódy:

Názov	**Návratový typ**	**Viditeľnosť**
cbeginColumns	`vector<vector<TrajectoryResult>>::iterator`	`public`
Vracia smerník (konštantný iterátor) na prvý prvok v poli vypočítaných stĺpcov *results*.		
cendColumns	`vector<vector<TrajectoryResult>>::iterator`	`public`
Vracia smerník (konštantný iterátor) na adresu za posledným prvkom v poli vypočítaných stĺpcov *results*.		
addColumn	`void`	`public`
Pridá nový stĺpec do poľa vypočítaných stĺpcov rezu HP.		
progress	`double`	`public`
Vracia stav výsledku vo formáte `double` s hodnotou od 0 do 1.		

Zmena aplikácie pre iné systémy diferenciálnych rovníc

Pri budúcom výskume s aplikáciou môže dôjsť k potrebe iných výpočtov. Pomocou nasledujúcich krokov je možné vytvoriť modifikovanú verziu aplikácie pre počítanie ľubovoľnej sústavy diferenciálnych rovníc:

1. Vytvoriť novú triedu v rámci modulu *Calculation*, ktorá rozširuje triedu `AbstractTrajectoryCalculator`. Pri tomto kroku je potrebné implemen-

tovať metódy `calculateTrajectory` a `calculateTrajectoryResult`. Tie slúžia na výpočet jednej trajektórie, pričom prvá metóda ukladá a vracia vypočítané body, kým druhá vracia iba zistený typ atraktora pre dané ZP podľa poskytnutého parametra *tests*.

2. Upraviť grafické rozhranie, aby bolo možné zadať parametre výpočtu, s ktorými novovytvorená trieda z prvého kroku pracuje. Ako základ môže byť použité grafické rozhranie aplikácie *Chuaviz*, či už *Chuaviz DE*. Je potrebné v prvom rade prispôsobiť hlavné okno s výpočtom trajektórie, ktoré obsahuje vstupné polia s parametrami, ale aj widget *CircuitParameters*, ktorý zobrazuje zadané parametre pri výpočte rezu a série rezov HP. Tieto úpravy je možné vykonať aj graficky v programe *QtCreator*.

3. Ďalej je potrebné upraviť súbor `trajectorywidget.cpp`, ktorý tieto parametre posúva triedam v module *Calculation*.

4. Posledným krokom je úprava súboru `mainwindow.cpp`, v ktorom sa nachádza kód pre ukladanie parametrov výpočtu do súboru, resp. kód pre ich načítanie.

Príloha E – Článok publikovaný na IEEE konferencii Radioelektronika, Brno, apríl 2017

V rámci diplomovej práce bol publikovaný článok *Parallelizing Boundary Surface Computation of Chua's Circuit* na medzinárodnej konferencii *27th International Conference Radioelektronika 2017* pod záštitou československej sekcie *IEEE*.

Článok predstavuje princíp paralelizácie výpočtu HP, uvádza aplikáciu *Chuaviz* a výsledky testovania rýchlosti výpočtu aplikáciou za rôznych okolností.

Konferencia sa konala v Brne od 19 do 21. apríla 2017 na VUT, na ktorom sa zúčasnil aj autor tejto práce a článok úspešne odprezentoval. Na ďalších stranách prílohy uvádzame článok v plnom znení.

Parallelizing Boundary Surface Computation of Chua's Circuit

Zsolt Racz, Branislav Sobota*
Dept. of Computers and Informatics
Technical University of Košice
Košice, Slovakia
zsolt.racz@student.tuke.sk,
branislav.sobota@tuke.sk*

Milan Guzan
Dept. of Theoretical and
Industrial Electrical Engineering
Technical University of Košice
Košice, Slovakia
milan.guzan@tuke.sk

***Abstract*—This paper introduces a method for performing parallel computations of the boundary surface of Chua's circuit. The presented approach could be characterized as a SIMD method which is based on the utilization of all available CPU cores. Performance comparisons between single and parallelized calculations on different computer systems are included in a table at the end of the article. The best results were observed using the highest compiler optimisation options on 64 bit architecture and using Intel i7 CPUs.**

***Keywords*—cross-section; boundary surface; chaos; Chua's circuit; attractor; Intel TBB; parallelization; SIMD**

I. Introduction

Boundary surface (BS) is an important part of analysis in the theory of non-linear circuits. Its importance lies in separating attractors. In a 2D graphical view, distinct colors are used to differentiate individual regions of attraction (RA) on a cross-section of a BS. 10 to 20 years in the past, projecting cross-sections of BS was only possible into one of the planes at the same time [1]. The later use of Monge's projection allowed to present 3D space using 2 planes [2], which improved the descriptiveness of non-linear circuit dynamics. After the spread of 3D graphics, showing BS in a 3D projection became an interesing option as it could be utilized to view the BS from multiple points and angles in real time. Such a projection requires a serie of 2D cross-sections, however the calculation of a single cross-section can easily take 5 hours on a modern desktop computer using 1 CPU [3], which in a case of e.g. 300 sections can produce a serious problem in terms of time and energy consumption. This paper presents a way of speeding up BS computation of Chua's circuit by distributing calculations between multiple CPU cores of a computer.

II. Chua's circuit and BS calculation

Chua's circuit is characterized by the following system of equations:

$$\begin{aligned} C_1\frac{du_1}{dt} &= \frac{1}{R}(u_2-u_1)-g(u_1)\\ C_2\frac{du_2}{dt} &= \frac{1}{R}(u_1-u_2)+i\\ L\frac{di}{dt} &= -u_2-\rho i, \end{aligned} \tag{1}$$

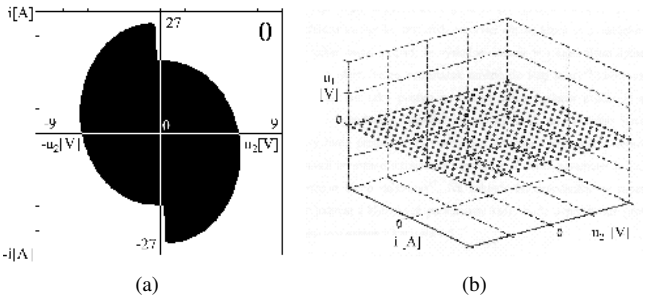
where:

$$\begin{aligned} g(u_1) = m_2u_1 &+ \frac{1}{2}(m_1-m_0)(|u_1-B_P|-|u_1+B_P|)+\\ &+\frac{1}{2}(m_2-m_1)(|u_1-B_0|-|u_1+B_0|). \end{aligned} \tag{2}$$

Depending on the initial conditions (IC), the state of the circuit, with parameters (3) can lead to a chaotic attractor (CHA) or a stable limit cycle (SLC).

$$\begin{aligned} C_1 &= 0.1\,\mathrm{F}, & C_2 &= 2\,\mathrm{F}, & R &= 1.428\,\Omega,\\ L &= 1/7\,\mathrm{H}, & B_P &= 1\,\mathrm{V}, & B_0 &= 900\,\mathrm{V}, & \rho &= 0\,\Omega,\\ m_0 &= -4\,\mathrm{S}, & m_1 &= -0.1\,\mathrm{S}, & m_2 &= 5\,\mathrm{S}. \end{aligned} \tag{3}$$

Cross-sections of the BS for (3) are illustrated by Fig. 1a. The imaginary line that separates white (RA for SLC) and black (RA for CHA) areas represents a single contour of BS. The cross-section was calculated by grid technique so that Fig. 1a was constructed using 440×440 points. The number in the top right corner denotes that the cross-section was led through plane $u_1 = 0$V [3]. Fig. 1 illustrates the grid for IC $u_1 = 0$V. The calculation has been done by solving the equation system (1) with the fourth order Runge-Kutta method, using a single-threaded program developed in programming language C. After a certain amount of time, every representative point (RP) got attracted eighter to SLC or CHA.



(a) (b)

Fig. 1. a) Cross-section of the BS for $u_1 = 0$V, b) Grid of ICs for $u_1 = 0$V.

Almost all conventional computers are nowadays equipped with multiple CPU cores, therefore it is interesting to consider parallelization as a speed-up technique for BS calculation. This way the operation could become less time, and energy consuming. From the Fig. 1b it can be clearly seen, that each trajectory could be calculated independently from the others. A similar method for parallel computing has already been introduced in [4] for calculation of Lyapunov exponents using MATLAB, which brought up to 4x improvement in terms of computation time (using 8 CPUs).

The process of the calculation of the BS shown in Fig. 1a started with the 0th column, where gradually 440 points (IC) have been computed. After the completion of the last column in the 0th row, the process started over using the 1st row until the 440th row. Every IC has been attracted to one of the two attractors. As it can be seen on the diagram on Fig. 2, this process can be expressed using two nested loops.

The trajectory type detection (CHA or SLC) for the given IC (i, u_1, u_2 or x, y, z) is managed by applying a set of testing conditions for every RP of the trajectory, until an RP that complies one of the conditions is found. Every call to the function *calculateTrajectory* is only dependent on the parameters of the circuit, set of testing conditions and ICs x, y and z. Since the parameters of the circuit, testing conditons and the value of z is constant during the whole calculation of a single cross-section and all values of x and y are known in the beginning of the calculation, all calls to this function during a BS calculation could be parallelized.

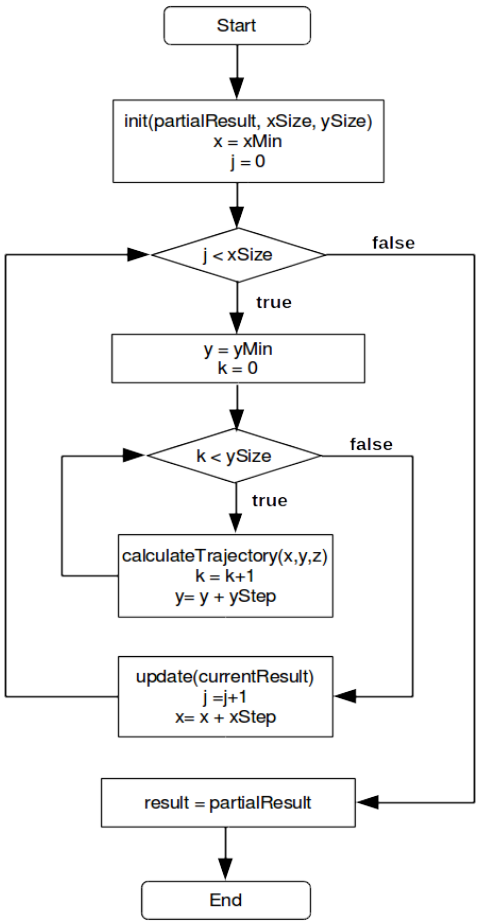


Fig. 2. Algorithm for Sequential Calculation of a BS.

III. Designing the Parallel Application

The approach presented in this section principally represents a type of a SIMD computation. The following technical requirements were stated on the application before starting the development: high computational performance, user friendly GUI and platform independence. Programming language C++ is one the fastest among modern programming languages [5], [6] that supports multiple platforms and a wide range of free libraries is available for it. Based on these facts, it was chosen for the development of the application. In addition to that, the Qt framework was chosen to power the GUI and Intel TBB for compute-intensive operations.

Intel TBB is a library for C++ that allows building portable parallel programs. It provides a higher level API for multithreading, than standard system calls, implementing task parallelism. With this approach the programmer defines the tasks, and the library's role is to optimally distribute them between the available CPU cores. The distribution is done by the Task Scheduler component, which utilizes *task stealing* – a technique that mitigates unequal workload between the threads [7]. Intel TBB also provides a number of generic parallel algorithms as C++ templates, e.g: parallel for loop (*parallel_for*), parallel reduction (*parallel_reduce*), pipelining (*parallel_pipeline*) or parallel sorting (*parallel_sort*) [8].

Our choice fell on *parallel_for*. This construction is intended to parallelize a serie of computations, while the particular computations are distinguished using a single integer value – the iterator. Therefore we can say, it is a limited version of the standard *for* loop, in which all iterations could be executed in parallel [8]. Based on the estimated characteristics of parallel tasks in our application and general recommendations about optimum grain size [8], [9], the outer loop (Fig. 2) was chosen to be replaced with *parallel_for*. This way one task represents the calculation of a single column of the cross-section, which implicates, that while calculating a cross-section of size M×N, where M equals to the number of columns and N equals to the number of rows, M tasks will be created.

Based on these requirements and design choices, an application called *Chuaviz* was created. Its user interface provides two switchable views. The first view (Fig. 3) allows to modify the parameters, perform the calculation of a single trajectory and visualize the result using 3 planes. The projected trajectory can be then moved by dragging and zoomed by mouse scrolling. The cuboids for CHA testing required for BS calculation are also visualized in this view. On Fig. 3 the cuboid can be observed as a red rectangle.

The second view, which can be seen on Fig. 4 is intended for the calculation and visualization of a single cross-section

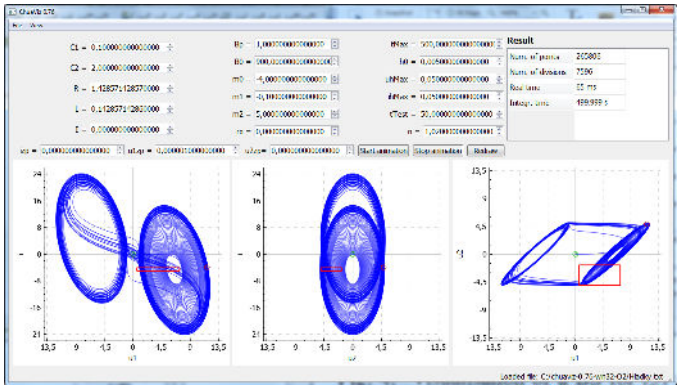


Fig. 3. Visualization of a single trajectory.

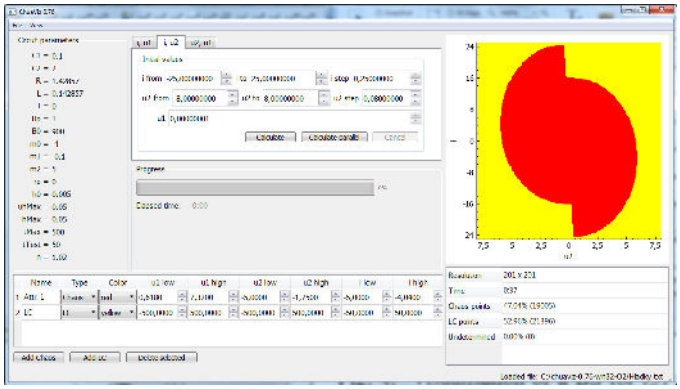


Fig. 4. Visualization of a BS for (3) and $u_1 = 0$V.

of the BS. Among others it allows to define new and modify existing cuboids for CHA and SLC testing.

IV. Performance testing

An important first step when optimizing the speed of the computation is to select a right value for parameter n – Table I. The integrational step is multiplied with this parameter, until $uhMax$ or $ihMax$ does not get exceeded. The role of these parameters is to define the maximal distance of the next point from the previous one, calculated with the Runge-Kutta method. They can be used to determine the accuracy of the calculation. Their value is also an important speed factor, therefore multiple variations were evaluated during the testing. In Table I, they are noted as (in the order of $uhMax$, $ihMax$): a) 0.05, 0.05, b) 0.05, 0.01, c) 0.01, 0.05, d) 0.01, 0.01. It can be clearly seen from Table I, that the fastest computations were for the case a) and $n = 1.02$ or $n = 1.002$. This should be further proven by calculation of cross-section of the BS. These performance tests have been completed on a single CPU core, with the application compiled using the O2 compiler optimization flag (the tests using Od and O1 were slower).

In the next test we computed a cross-section of the BS with 200×200 ICs. Our main subject of investigation was the placement of the cuboid for detecting CHA. The detection is based on a simple algorithm: once a point of the trajectory gets inside the borders of the cuboid, the current ICs are evaluated as leading to CHA and the calculation continues with the next ICs. Four variously placed cuboids were tested, the results are presented in Table II. As we can see, the 4th cuboid and the $n = 1.02$ or $n = 1.002$ values were the fastest again.

It should be noted that the time required to determine the type of every trajectory is recorded during the computation to identify time consuming trajectories. In the early times of the development, the real CPU time was measured for every trajectory, however later it has been changed to the integration time instead, which resulted in a significant speed increase – up to 2.3 times.

TABLE I
Computation Times of a Single Trajectory

	n	**2**	**1.2**	**1.02**	**1.002**
a)	Num. of points	266 561	265 660	261 619	285 382
	Num. of div.	266 561	69 879	7 474	825
	Time CPU [ms]	58.5	45.1	38.7	39.8
b)	Num. of points	1 317 582	1 276 267	1 275 883	1 299 908
	Num. of div.	1 317 584	335 706	36 455	3 751
	Time CPU [ms]	294.2	216.7	188.2	187.8
c)	Num. of points	1 365 083	1 327 870	1 327 870	1 327 300
	Num. of div.	1 365 085	349 279	37 915	3 830
	Time CPU [ms]	298	228.3	192.9	193.6
d)	Num. of points	505 903	485 011	479 195	474 347
	Num. of div.	505 903	127 576	13 692	1 369
	Time CPU [ms]	121.2	88.5	79.6	79.3

TABLE II
Computation Times of 1 Cross-section of BS (200 × 200 Points) for Various Cuboids for Detecting CHA

n	**2**	**1.2**	**1.02**	**1.002**
Time 1 [min]	1:14	0:49	0:42	0:44
Time 2 [min]	0:58	0:39	0:33	0:33
Time 3 [min]	1:06	0:44	0:36	0:38
Time 4 [min]	0:55	0:37	0:31	0:32

TABLE III
Comparison of Computation Times on Multi-core Computers

Computers	32 / 64 bit operting system			
	Od	O1	O2	1 CPU, O2
	t [min]	t [min]	t [min]	t [min]
PC1	2:17 / 1:45	0:57 / 0:28	0:30 / 0:29	1:23 / 1:15
PC2	2:51 / 1:42	1:22 / 0:38	0:39 / 0:38	3:24 / 2:24
PC3	1:43 / 1:35	0:53 / 0:50	0:43 / 0:44	2:24 / 2:29
PC4	6:06 / -	3:24 / -	1:29 / -	3:55 / -
PC5	4:55 / -	2:52 / -	1:27 / -	2:54 / -
PC6	7:40 / -	5:30 / -	2:10 / -	5:19 / -
PC7	0:51 / 0:43	0:24 / 0:18	0:17 / 0:18	1:22 / 1:20
PC8	2:47 / 2:04	1:13 / 0:58	0:59 / 0:57	1:54 / 1:52
PC9a	- / 0:35	- / 0:14	- / 0:14	- / 1:06
PC9b	0:49 / 0:38	0:24 / 0:16	0:15 / 0:15	1:18 / 1:17

TABLE IV
THE LIST OF TESTED COMPUTERS

Name	CPU model	OS	cores / threads
PC1	NB Intel i5-5200U, 2.2 GHz	Win 8.1 64 bit	2 / 4
PC2	Intel Q9550, 2.83 GHz	Win 7 64 bit	4 / 4
PC3	NB Intel i7-2670QM, 2.2 GHz	Win 7 64 bit	4 / 8
PC4	Intel E5200 2.5 GHz	Win 7 32 bit	2 / 2
PC5	AMD Athlon II 215, 2.7 GHz	Win 7 32 bit	2 / 2
PC6	Intel E6600, 2.4 GHz	Win 7 32 bit	2 / 2
PC7	Intel i7-3770, 3.4 GHz	Win 7 64 bit	4 / 8
PC8	Intel G3250T, 2.8GHz	Win 10 64 bit	2 / 2
PC9a	Intel i7-4790K, 4 GHz	Linux Mint 18	4 / 8
PC9b	Intel i7-4790K, 4 GHz	W10 64 bit (virt.)	4 / 8

The next Table III presents an overview of times required to complete the computation of a single cross-section on various desktop and notebook (NB) computers. Their parameters are listed in Table IV.

The measurments were made on idle computers, while 32 and 64 bit versions of the application were tested with three compiler optimization levels: disabled optimization (Od), binary size minimalization (O1) and optimization for maximum speed (O2). The measurements were repeated at least 5 times for each setting while in all cases the value of n was set to 1.02 and the fastest cuboid (see Time 4 in Table II) was used for CHA detection. During the measurements the process priority of the application was set to the operating system's default and the calculations were started from the GUI.

Regarding Table III, the 64 bit versions of the application outperformed the 32 bit versions in almost all cases, however the difference was significant only on low optimization levels. The fastest CPUs were the Intel i7s in PC7 and PC9, the Intel i7 in PC3 performed worse, since it is a mobile CPU. It is also interesting to see the difference between PC1 (5th generation) and PC3 (2th generation Core CPU).

The 4th column of Table III shows the calculation times of a single threaded computation using the O2 compiler flag. Comparing it with the 3rd column, the speed benefits of parallelization can be clearly seen. The calculation speed of BS has been increased by a factor of up to 5.23 on some quad-core CPUs (e.g. PC2).

According to these tests it is worth to invest in a newest generation high-end CPU. PC5 was also tested with 32 bit Windows 10 – the time to complete the calculations was approximately the same. It is also an interesting observation from Table III, that despite PC2 is older than PC1 and PC3, their performance in BS calculation is comparable. Another interesting observation is, that the virtual machine on PC9b does not slow down the computation significantly compared to native execution on Linux.

V. CONCLUSION

The application presented in this paper is intended for performing computations related to Chua's circuit. However, the introduced method of parallelization is not limited to the calculation of BS of Chua's circuit, but it could be also used for BS calculation of other non-linear dynamical systems, Lyapunov exponents or other computations consisting of M×N independent tasks [10], [11]. The authors will continue working towards implementation of the application for other dynamic systems.

The paper also highlights the importance of sensible design choices, like choosing the appropiate data to record, determining the maximal allowed distance between the result points $(uhMax, ihMax)$, compiler optimization options or selecting the right cuboid that identifies the trajectory type correctly, and at the same time it is fast.

Using the developed application it is now possible to utilize any conventional multi-core computer more efficiently towards reaching a notably faster computing speed for BS calculation of Chua's circuit compared to single threaded calculations. Even more performance gains could be possibly reached by using SIMD instructions, optimizing for concrete CPU models or with other source code optimizations techniques, which may be the subject of future research.

The compiled binaries, as well as the source code of the application presented in this paper, are available upon email request to co-author M. Guzan.

VI. ACKNOWLEDGMENT

Research described in this paper was supported through project number GA15-22712S.

REFERENCES

[1] V. Špány, P. Galajda, and M. Guzan, "The state space mystery in multiple-valued logic circuit with load plane – part I," *Acta Electrotechnica et Informatica*, vol. 1, pp. 17–22, 2001, ISSN 1335-8243.

[2] P. Galajda, V. Špány, and M. Guzan, "The state space mystery with negative load in multiple-valued logic," *Radioengineering*, vol. 8, no. 2, pp. 2–7, 1999, ISSN 1210-2512.

[3] B. Sobota, "3D modelling of Chua's circuit boundary surface," *Acta Electrotechnica et Informatica*, vol. 11, no. 1, pp. 44–47, 2011, ISSN 1335-8243.

[4] T. Götthans, J. Petržela, and Z. Hruboš, "Advanced parallel processing of lyapunov exponents verified by practical circuit," in *Proceedings of 21st Int. Conf. Radioelektronika 2011*, April 2011, pp. 1–4.

[5] L. Prechelt, "An empirical comparison of C, C++, Java, Perl, Python, Rexx, and Tcl for a search/string-processing program," Fakultät für Informatik, Univ. Karlsruhe, Germany, Tech. Rep. 2000-5, March 2000.

[6] M. Fourment and M. R. Gillings, "A comparison of common programming languages used in bioinformatics," *BMC Bioinformatics*, vol. 9, no. 1, p. 82, 2008. [Online]. Available: http://dx.doi.org/10.1186/1471-2105-9-82

[7] G. Contreras and M. Martonosi, "Characterizing and improving the performance of Intel Threading Building Blocks," in *2008 IEEE Int. Symposium on Workload Characterization*, Sept 2008, pp. 57–66.

[8] Intel Software. Intel® Threading Building Blocks Developer Reference: Algorithms. [Online]. Available: https://software.intel.com/en-us/node/506140

[9] K. Farnham. (2007, Aug) TBB parallel_for grain size experiments. [Online]. Available: https://software.intel.com/en-us/blogs/2007/08/07/tbb-parallel_for-grain-size-experiments

[10] J. Petržela, "Optimal piecewise linear approximation of the quadratic chaotic dynamics," *Radioengineering*, vol. 21, no. 1, pp. 20–28, 2012, ISSN 1210-2512.

[11] T. Götthans and J. Petržela, "Experimental study of the sampled labyrinth chaos," *Radioengineering*, vol. 20, no. 4, pp. 420–427, 2011, ISSN 1210-2512.

Príloha F – Článok na recenzii IEEE konferencie TSP, Barcelona, júl, 2017

Ďalší článok s názvom *Cross-Sections of Boundary Surface for Variations of Parameters in Chua's Circuit*, na ktorej sa autor tejto práce podieľal spoluautorstvom, bol predložený na medzinárodú konferenciu *2017 40th International Conference on Telecommunications and Signal Processing* pod zášitou IEEE. Konferencia je organizovaná šestnástimi univerzitami celého sveta v Barcelone, od 5 do 7. júla 2017. Aktuálne je článok na recenzii a čaká sa na definitívne schválenie.

Článok uvádza nové typy hraničných plôch, ktoré boli vypočítané aplikáciou *Chuaviz* prezentovanou v tejto práci. Na ďalších stranách prílohy uvádzame článok v plnom znení.

Cross-Sections of Boundary Surface for Variations of Parameters in Chua's Circuit

Ing. Milan Guzan, Ph.D., Kristián Szerdahelyi*
Dept. of Theoretical and
Industrial Electrical Engineering
Technical University of Košice
Košice, Slovakia
Email: milan.guzan@tuke.sk,
kristian.szerdahelyi@student.tuke.sk*

Zsolt Rácz, doc. Ing. Branislav Sobota, Ph.D.*
Dept. of Computers and Informatics
Technical University of Košice
Košice, Slovakia
Email: zsolt.racz@student.tuke.sk,
branislav.sobota@tuke.sk*

***Abstract*—A research of calculation and displaying of Chua's circuit's boundary surfaces for various parameters using cross-sections is presented in this paper. Colored regions of attraction help us to create a notion about the morphology of the boundary surface, which separates attractors from each other. The cross-sections of boundary surface presented in this paper, however, interfere with our conventional notion of the boundary surface's morphology. New cross-section types of boundary surface are introduced that were not published until now. These, considering their complex morphology are hard – if not impossible – to physically reconstruct in 3D.**

***Keywords*—attractor; boundary surface; chaos; Chua's circuit; cross-section; region of attraction; Runge-Kutta.**

I. Introduction

Boundary surface (BS) in non-linear electric circuits is an object that separates individual attractors from each other. Without sufficient knowledge about BS it is hard to explain failures of memory cells or to control their state, similarly as with Chua's circuit, when BS gave us answer to the question: What separates chaotic attractors (CHA) from stable limit cycles? [1]. For this reason in [2] BS was published for the first time. Its morphology was described using a serie of cross-sections. In the case of Chua's circuit and the double scroll CHA, BS could be imagined as a tube with a (sharp) protrusion on its side. In the case when a single double-scroll CHA breaks down to two single-scroll CHAs, the BS takes a coil-like shape [3] – [5]. This way the CHA is encapsulated in the BS and all of the initial conditions (IC) selected inside the tube will get attracted to CHA. The attractor for the ICs, that are located outside of the tube is infinity or in the case of a physical system, a stable limit cycle (SLC). If our goal is e.g. signal encryption, the amplitude of the encrypted signal should not exceed the BS that separates the undeterminable chaotic signal from the periodic one which could be determined easily. How high the input signal could be – the exact answer is given by BS. Publication [7] presents a large gallery of CHAs of Chua's circuit. It was the stimulus of the activity in [6] as well as of this paper. Is the morphology of the BS really going to be so universal for every single-scroll CHAs as it is described above?

Research described in this paper was supported through project number GA15-22712S.

II. Chua's circuit and boundary surface

Chua's circuit is characterized by a system of equations (1) and function (2).

$$\begin{aligned} C_1\frac{du_1}{dt} &= \frac{1}{R}(u_2-u_1)-g(u_1)\\ C_2\frac{du_2}{dt} &= \frac{1}{R}(u_1-u_2)+i\\ L\frac{di}{dt} &= -u_2-\rho i \end{aligned} \tag{1}$$

$$g(u_1) = G_b u_1 + \frac{1}{2}(G_\mathrm{a}-G_\mathrm{b})(|u_1+1|-|u_1-1|) \tag{2}$$

The cross-sections of BS for parameters 0a and 0b (listed in I) and for $u_1 = 0$ are shown in Fig. 1a and Fig. 1b. The images illustrate regions of attraction (RA) for CHA (gray RA) and for infinity (white RA) [3]. By calculating multiple cross-sections of BS they can be used for reconstruction of BS in 3D [4] – Fig. 1c. Based on that we can see that CHA is encapsulated in BS. The area surrounding the BS on Fig. 1c is the RA of infinity. Fig. 1 helps us to roughly imagine the conception of BS. The inside and the outside of the BS is illustrated with a single color. The BS – as it was already mentioned in the introduction – reminds a tube-shape with a (sharp) protrusion on its side.

The case of a coil-like BS shape mentioned in the introduction, when the double-scroll CHA breaks down to two single-scroll CHAs (parameters PC2 in Tab. I), is illustrated by Fig. 1d. At the first sight it seemed that every double or single-scroll CHA is going to have RAs that are similar to the one shown on Fig. 1. This hypothesis was also confirmed by [6], where cross-sections of BS were calculated for parameters PC1 – PC4 (abbr. from chapter "The Physical Circuits" in reference [7]). E.g. the RAs for PC1 and PC4 are similar to 1b and the RAs for PC2 were similar (although they were smaller) to Fig. 1d for parameters PC3. As much as 195 CHAs

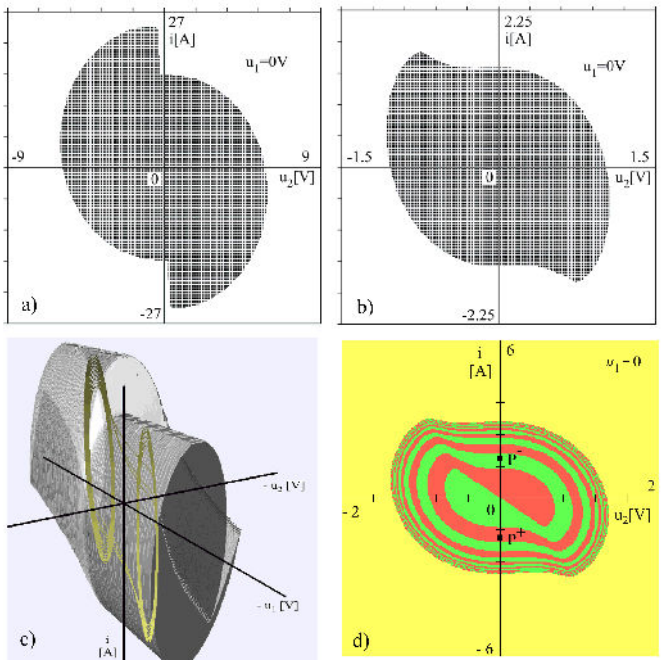


Fig. 1. Examples showing cross-sections of BS in 2D, projecting regions of attraction a), b), d) and a reconstruction of the BS from 300 cross-sections in 3D – c).

TABLE I
PARAMETERS FOR CALCULATION OF BS SHOWN IN THIS ARTICLE. THE PARAMETERS ARE EXPRESSED AS FOLLOWS: C1 AND C2 IN [F], R IN [Ω], L IN [H], GA IN [S], GB IN [S].

	C_1	C_2	R	L	ρ	G_a	G_b
0a	0.1	2	1.428571	0.142587	0	-4	-0.1
0b	0.11111	1	1.428571	0.142587	0	-0.8	-0.5
PC1	0.10443	1	0.989119	0.0625	0	-1.143	-0.714
PC2	0.10443	0.981	1	0.0625	0	-1.143	-0.714
PC3	0.10443	0.85	1	0.0625	0	-1.143	-0.714
PC4	0.10443	1	1	0.0625	0	-1.2	-0.714
PC5	-0.1333	8.2	1	0.31	-0.1	-0.98	-2.4
PC6	-0.1333	11	1	0.31	-0.1	-0.98	-2.4
PC13	-0.1333	10	0.786782	0.31	-0.1	-0.98	-2.4
PC15	-0.1333	10	0.778816	0.31	-0.1	-0.98	-2.4
PC17	0.06	15.35	-1	0.00667	0.000651	0.856	1.1
PC18	0.06	10	-1	0.0075	0.000651	0.856	1.1
PC20	1	-1.0837	33,33333	-1.49	2.228	-0.5	0.0064
PC31	1	-1.163	33,33333	-1.49	2.228	-0.5	0.0064
PC32	1.01	-1.0837	33,33333	-1.49	2.228	-0.5	0.0064
PC36	1	-1.0837	33,33333	-1.49	2.228	-0.52	0.0064

are presented in [7]. Are really the RAs going to be similar to the already known ones from Fig. 1a, Fig. 1b and Fig. 1d? We continued in calculatig cross-sections of BS further in plane i, u_2 at first using $u_1 = 0$, then for other values of u_1 – according to the morphology of the BS. However, due to the limited extent of this paper we are not showing cross-sections of BS for $u_1 \neq 0$ (except the case of PC32). The individual parameters are listed in Tab. I.

To avoid problems during the calculations, the units of parameters in Tab. I are given in base units instead of mS, μF, mH – as they are presented in [7].

Cross-sections of BS were calculated using the application presented in [8] with the option of parallel computation enabled. It was developed in C++, utilizes the Runge-Kutta method for calculation and offers a graphical user interface. The results are graphically presented in Fig. 2 using pairs of images – the phase portrait of the attractors in plane i, u_1 on the left and the cross-section on the right image. The parameters are denoted in the bottom left corner of the left image and all of the cross-sections of BS are calculated for $u_1 = 0$. Ten cases are illustrated in Fig. 2 this way. The labeling of axes in the first row of Fig. 2 (for PC5 and PC6) applies to all the other pairs of images as well, while the current i is shown in [A] and the voltages u_1 and u_2 in [V].

At the beginning of calculation, the cross-sections of BS for parameters PC5 and PC6 we got the expected results. The morphology of the BS was the same as we described in the introduction. For PC5 the BS was coil-shaped, since 2 single-scroll CHAs could be found in the circuit and for PC6 the RAs are single-colored – as it was expected. On these cross-sections of BS – opposed to Fig. 1 – we could not observe the sharp protrusion on their side. Similar types of cross-sections with slightly different dimensions were observed for parameters PC7 – PC 15 from [7]. However, using parameters PC16 – PC19 we have discovered some atypic cross-sections. The morphology of their RA was different from those that were published up to now. As we can see PC17 and PC18 in Fig. 2, it consists of a mix of a coherent single-colored RA and a coil-shaped formation representing the presence of two single-scroll CHAs. This kind of morphology has not been known until now. Parameters PC20 brought even more surprise. This cross-section of BS has been checked and calculated multiple times with adjustments of the accuracy using different integration step sizes, however, the character of the cross-section was preserved. Since the resolution of Fig. 2 is too small to see the surprising details, an enlarged version of the same cross-section is presented in Fig. 3. From the figure we can clearly see that:

- Four small yellow regions appear inside the red RA of CHA. The yellow color represents ICs that lead to infinity;
- The entire red region is interwoven with yellow dots.

The motion of the representative points was verified manually by calculating particular trajectories for some of the ICs. The trajectories were behaving according to the cross-section – the ICs laying in the yellow or red regions were attracted to infinity or to CHA. Therefore we can assume that Fig. 3 was computed correctly.

The RAs illustrated in Fig. 3, however, were not the only suprises. Going on with images PC31, PC32 and PC36 in Fig. 2 we can make other interesting observations. While examining a circuit with a single CHA we expected only

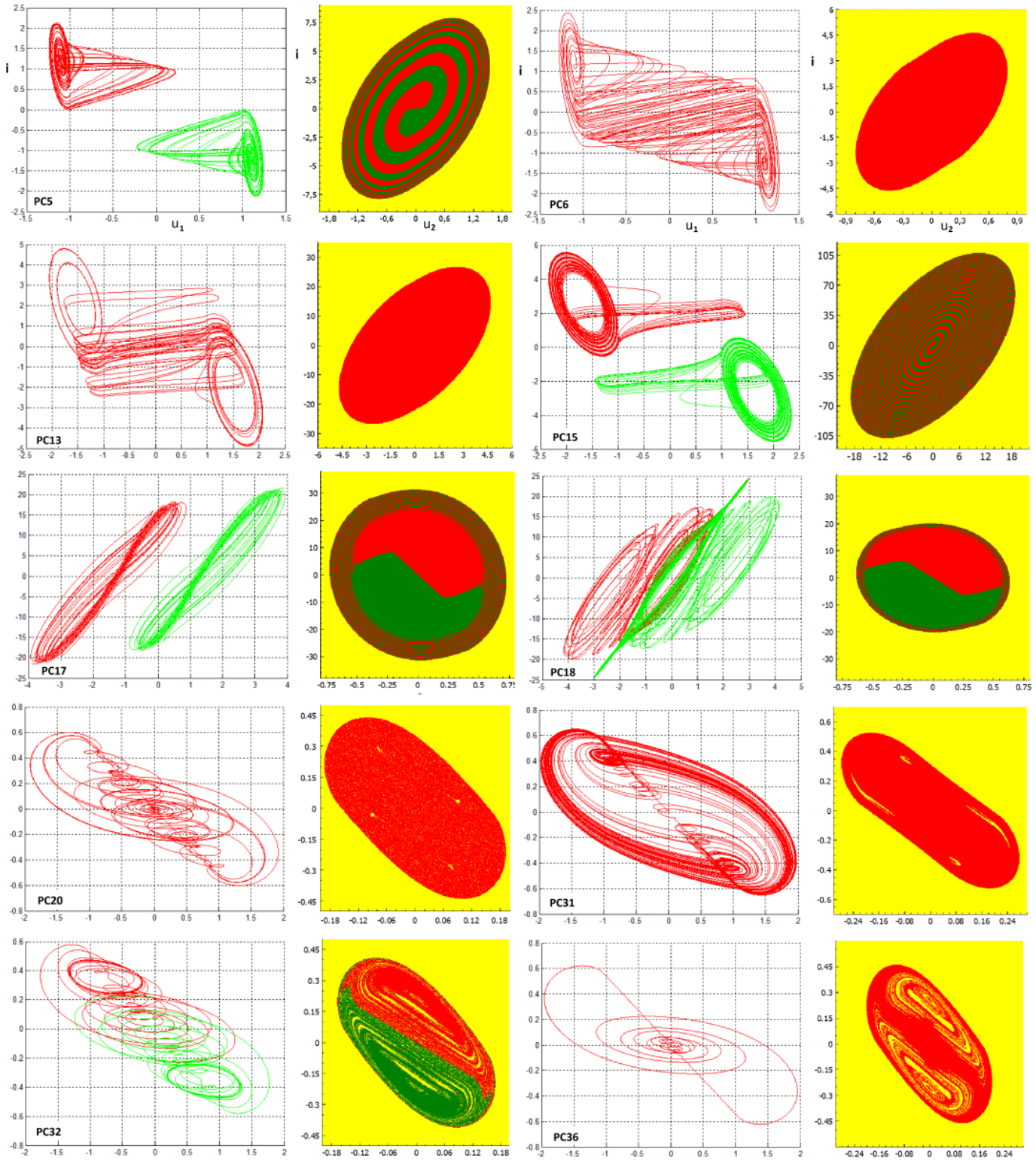


Fig. 2. Pairs of images showing chaotic attractors (left) and cross-sections of BS (right) for $u_1 = 0$. The names of the parameter sets for each of the pairs are shown in the bottom left corners of the trajectory. The numeric values of the parameters are revealed in Tab I.

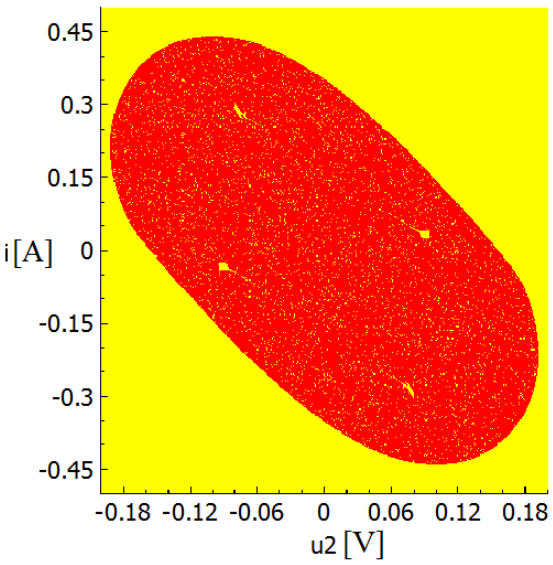


Fig. 3. The surprising morphology of RAs on the cross-section of BS for parameters PC20.

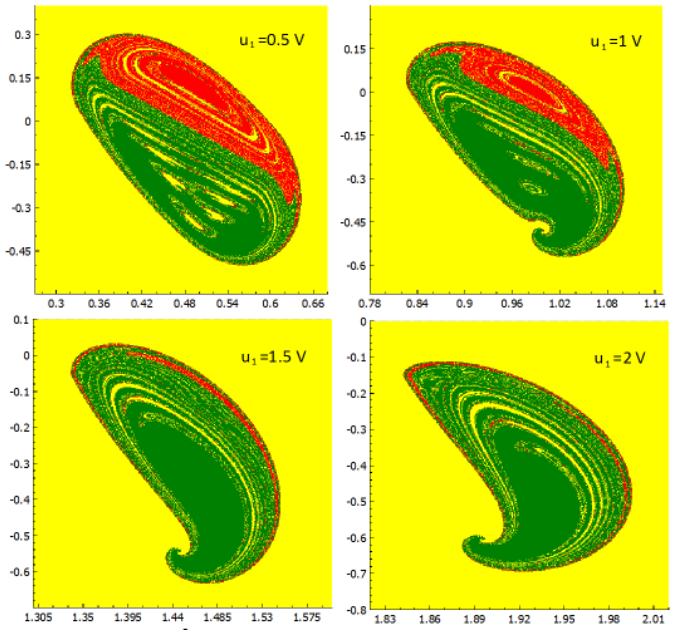


Fig. 4. Four cross-sections for parameters PC32 illustrating the morphology of BS in 3D.

two coherent single-colored RAs of yellow and red colors or three RAs of colors yellow, red and green in the case of two CHAs. Even though the mentioned colors are present on the images, however, they are intervowen with yellow areas again, representing the attractor of infinty. Boundary surfaces of this type have not been observed until now in any of the analysed autonomous systems; not in multiple-valued logic circuits nor in Chua's circut or other systems generating chaos [2] – [6], [9] – [14]. Probably this paper mentions these unconventially shaped cross-sections of BS for the first time ever.

To help the reader to imagine the morphology of BS in 3D, another four cross-sections of BS were calculated for parameters PC32 from $u_1 = 0.5\,\mathrm{V}$ to $2\,\mathrm{V}$ with a step of $0.5\,\mathrm{V}$ (the cross-section for $u_1 = 0$ can be found on Fig. 2). They are shown in Fig. 4. The labeling of axes is the same as in Fig. 2. Calculation took more than 200 minutes for every single cross-section on a computer equipped with Intel Core 2 Quad Q9550 2.83GHz CPU (running Windows 7 64 bit). A shift of the BS for $u_1 > 0$ in the bottom-left direction is clear from Fig. 4. Cross-sections for $u_1 < 0$ are centrally inverted compared to the images in Fig. 4 with green and red colors interchanged.

III. Conclusion

In the beginning of examining the morphology of Chua's circuit's BS we thought – based on the previous results – that the BS in 3D will be always shaped like a tube with a (sharp) protrusion on its side or have a coil-like shape. The first case of BS could be reconstructed using the calculated 2D cross-sections (Fig. 1c), however, the second (the coil-shape) cannot be effectively projected in 3D. This kind of BS morphology could be only visualized using a serie of 2D cross-sections. We also have found some other non-conventionally shaped BSs illustrated on Fig. 2 starting from PC17. These examples of cross-sections of BS are also not possible to reconstruct in 3D because the integrity of the red or green RA for CHA is damaged.

References

[1] T. Matsumoto, L. O. Chua, M. Komuro, "The double scroll," *IEEE Transactions on Circuits and Systems*, 1985, vol. CAS-32, pp. 797-818.

[2] V. Špány, L. Pivka, "Boundary surfaces in sequential circuits," *International Journal of Circuit Theory and Applications*, vol. 18, no. 4, pp. 349–360, 1990. DOI: 10.1002/cta.4490180404.

[3] V. Špány, P. Galajda, M. Guzan, L. Pivka, M. Olejár, "Chua's singularities: Great miracles in circuit theory," *International Journal of Bifurcation and Chaos (IJBC)*, vol. 20, no. 10, pp. 2993-3006, 2010.

[4] B. Sobota, "3D modelling of Chua's circuit boundary surface," *Acta Electrotechnica et Informatica vol. 11*, no. 1, pp. 44-47, 2011. ISSN: 1335-8243.

[5] M. Guzan, "Morphology of boundary surface of Chua's circuit using different values of R," *SEKEL 2009*, Brno: VUT, pp. 83-90, 2009.

[6] M. Guzan, "Variations of boundary surface in Chua's circuit," *Radioengineering*, vol. 24, no. 3 , pp. 814-823, 2015. ISSN 1210-2512.

[7] E. Bilotta, P. A. Pantano, "Gallery of Chua Attractors," *Series on Nonlinear Science Series A*, vol. 61, 2008, p. 607.

[8] Zs. Rácz, M. Guzan, B. Sobota, "Parallelizing boundary surface computation of Chua's circuit," *Radioelektronika 2017* (paper accepted).

[9] V. Špány, P. Galajda, M. Guzan, "The state space mystery in multiple-valued logic circuit with load plane - part I," *Acta Electrotechnica et Informatica*, vol. 1, no. 1, pp. 17-22, 2001. ISSN 1335-8243.

[10] M. Guzan, "Regions of attraction of sequential circuit," *SEKEL 2011*: International Conference Czech and Slovak Teachers of Electrical Engineering and Computer Science: Conference proceedings: Horní Lomná, 7.-9. Sept. 2011. - Ostrava: VŠB-TU, pp. 1-4, 2011. ISBN 978-80-248-2452 (in slovak).

[11] M. Guzan, "Boundary surface of a ternary memory in the absence of limit cycles," *Radioelektronika*, Proc. of 22-nd international conference: April 17-18, 2012, Brno, pp. 1-4, 2012. ISBN 978-80-214-4469-0

[12] J. Petržela, "Optimal piecewise- linear approximation of the quadratic chaotic dynamics," *Radioengineering*, vol. 21, no. 1, pp. 20-28, 2012.

[13] M. Guzan, "Boundary Surface of 5-Valued Memory," *Journal of Engineering*, vol. 2013, pp. 1-7, 2013. ISSN 2314-4904.

[14] J. Petržela, T. Gotthans, M. Guzan, "Dynamical tangles in third-order oscillator with single jump function," *The Scientific World Journal*, vol. 2014, article no. 239407, pp. 1-12, 2014. ISSN 2356-6140.

Príloha G – Obsah priloženého CD

- CD
  - chuaviz
    - dist
      - chuaviz-linux64 ..... 64 bitová verzia *Chuaviz* pre Linux
      - chuaviz-win32 ..... 32 bitová verzia *Chuaviz* pre Windows
      - chuaviz-win64 ..... 64 bitová verzia *Chuaviz* pre Windows
    - src ..... Zdrojové súbory aplikácie *Chuaviz*
  - chuaviz-de
    - dist
      - chuaviz-de-linux64 ..... 64 bitová verzia *Chuaviz DE* pre Linux
      - chuaviz-de-win32 ..... 32 bitová verzia *Chuaviz DE* pre Windows
      - chuaviz-de-win64 ..... 64 bitová verzia *Chuaviz DE* pre Windows
    - src ..... Zdrojové súbory aplikácie *Chuaviz DE*
  - parameters
    - chuaviz
      - cuboids ..... Parametre výpočtu pre merania z kapitoly 5.1.2
      - cuboids-multi ... Parametre výpočtu pre merania z kapitoly 5.1.3
      - pc ..... Parametre fyzikálnych obvodov
      - test-hmax ..... Parametre výpočtu pre merania z kapitoly 5.1.1
    - chuaviz-de
      - ce ..... Parametre bezrozmerných obvodov s kubickou funkciou $f(x)$
      - de ..... Parametre obvodov s 3-segmentovou funkciou $f(x)$
  - latex ..... Zdrojový text tejto práce v LaTeX
  - nastroje
    - qt-unified-online.exe .... Inštalátor *Qt* a *QtCreator* pre Windows
    - qt-unified-online.run ..... Inštalátor *Qt* a *QtCreator* pre Linux
  - DP-Racz.pdf ..... Elektronická verzia tejto práce