UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

KONTINUÁCIA STACIONÁRNYCH A PERIODICKÝCH RIEŠENÍ SYSTÉMOV OBYČAJNÝCH DIFERENCIÁLNYCH ROVNÍC V PROGRAMOVACOM JAZYKU PYTHON

DIPLOMOVÁ PRÁCA

2017 Bc. Alexandra RAVINGEROVÁ

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

KONTINUÁCIA STACIONÁRNYCH A PERIODICKÝCH RIEŠENÍ SYSTÉMOV OBYČAJNÝCH DIFERENCIÁLNYCH ROVNÍC V PROGRAMOVACOM JAZYKU PYTHON

DIPLOMOVÁ PRÁCA

Študijný program:	Ekonomicko-finančná matematika a modelovanie
Študijný odbor:	9.1.9. Aplikovaná matematika (1114)
Školiace pracovisko:	Katedra aplikovanej matematiky a štatistiky
Vedúci práce:	doc. Mgr. Pavol Bokes, PhD.

Bratislava 2017 **Bc. Alexandra RAVINGEROVÁ**

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Alexandra Ravingerová
Študijný program: ekonomicko-finančná matematika a modelovanie (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: aplikovaná matematika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Kontinuácia stacionárnych a periodických riešení systémov obyčajných diferenciálnych rovníc v programovacom jazyku Python.
Continuation of stationary and periodic solutions to systems of ordinary differential equations in the programming language Python.

Cieľ: Cieľom práce bude študovať závislosť stacionárnych a izolovaných periodických riešení systémov obyčajných diferenciálnych rovníc na bifurkačnom parametri pomocou softvéru AUTO-07p v rámci programovacieho jazyka Python.

Vedúci: doc. Mgr. Pavol Bokes, PhD.
Katedra: FMFI.KAMŠ - Katedra aplikovanej matematiky a štatistiky
Vedúci katedry: prof. RNDr. Daniel Ševčovič, CSc.

Dátum zadania: 16.11.2016

Dátum schválenia: 16.11.2016 prof. RNDr. Daniel Ševčovič, CSc.
garant študijného programu

.....
študent

.....
vedúci práce

Poďakovanie Touto cestou sa chcem poďakovať vedúcemu práce doc. Mgr. Pavlovi Bokesovi, PhD. za príležitosť vypracovať túto prácu pod jeho vedením, cenné rady a motiváciu. Veľká vďaka patrí mojej rodine a priateľom za to, že vo mňa veria a špeciálne Matejovi za nesmiernu trpezlivosť a podporu počas celého štúdia.

Abstrakt v štátnom jazyku

RAVINGEROVÁ, Alexandra: Kontinuácia stacionárnych a periodických riešení systémov obyčajných diferenciálnych rovníc v programovacom jazyku Python [Diplomová práca], Univerzita Komenského v Bratislave, Fakulta matematiky, fyziky a informatiky, Katedra aplikovanej matematiky a štatistiky; školiteľ: doc. Mgr. Pavol Bokes, PhD., Bratislava, 2017, 68 s.

Táto práca sa zaoberá kontinuáciou riešení dynamických systémov z teoretického aj aplikovaného pohľadu. Ponúkame pohľad do teórie dynamických systémov a do bifurkačnej teórie. Z teoretického hľadiska sa venujeme základným numerickým princípom kontinuácie. Predstavujeme Auto-07p, program na riešenie kontinuačných a bifurkačných problémov diferenciálnych rovníc. Na konkrétnych príkladoch prinášame podrobný návod na používanie Auto-07p v spolupráci s programovacím jazykom Python a interpretáciu získaných riešení. Na záver prinášame aplikáciu získaných poznatkov na modeli správania neurónovej skupiny.

Kľúčové slová: Python, Auto-07p, Bifurkácia, Kontinuácia, Dynamické systémy

Abstract

RAVINGEROVÁ, Alexandra: Continuation of stationary and periodic solutions to systems of ordinary differential equations in the programming language Python [Master's Thesis], Comenius University in Bratislava, Faculty of Mathematics, Physics and Informatics, Department of Applied Mathematics and Statistics; Supervisor: Doc. Mgr. Pavol Bokes, PhD., Bratislava, 2017, 68 p.

The thesis is focused on continuation of solutions of dynamical systems in terms of theory and practical use. It provides insight into theory of dynamical systems and bifurcation theory. It also presents fundamental principles of numerical continuation. The thesis introduces Auto-07p, a software for continuation and bifurcation problems in differential equations. Using specific examples the thesis offers a detailed manual for use and interpretation of obtained results of Auto-07p together with programming language Python. Finally, the thesis models behaviour of neurons under changing input current.

Keywords: Python, Auto-07p, Bifurcation, Continuation, Dynamical systems

Obsah

Úvod

Teória bifurkácií sa zaoberá myšlienkou, že riešenie systémov diferenciálnych úloh s parametrom sa pri zmene parametra mení a v niektorých prípadoch môže malá zmena parametra viesť k rozsiahlej zmene správania celého systému. Tento systém diferenciálnych rovníc môže predstavovať jednotlivé biochemické reakcie v bunkách [16] alebo popisovať princípy fungovania neurónových systémov v mozgu [6]. Je ľahké predstaviť si, aké široké uplatnenie maju dynamické systémy a bifurkácie v prírodných vedách i ekonómii. Jedným z hlavných podnetov pre vypracovanie tejto práce je práve vysoký potenciál rozširovanie znalostí o zvolenú tému.

Úlohy vyššej zložitosti je častokrát náročné všeobecne vyriešiť. Ideálnym spôsobom, ako si túto prácu uľahčiť je použitie správneho počítačového softvéru. Na riešenie bifurkačných úloh sa obvykle používa samostatný program Auto alebo špeciálne knižnice k programu MatLab. V našej práci využívame program Auto v spolupráci s Pythonom. Jednou z výhod tohto výberu je voľný prístup k týmto programom.

Cieľom práce je vytvoriť ucelený a užitočný úvod do problematiky bifurkácií. Venujeme sa teoretickým aj praktickým aspektom riešenia kontinuačných a bifurkačných úloh. V prvej kapitole popisujeme teóriu dynamických systémov a bifurkácií. Druhá kapitola je úvodom do numerickej kontinuácie riešení. Tretia kapitola je zameraná na samotný použitý softvér a praktické aspekty jeho používania. Vo štvrtej kapitole ponúkame návod na prácu s programom Auto. V poslednej kapitole získané poznatky aplikujeme na konkrétnu úlohu a interpretujeme výsledky.

1 Dynamické systémy

V tejto kapitole v krátkosti uvádzame teóriu dynamických systémov a ich bifurkácií. Venujeme sa vybraným druhom bifurkácií a ich vlastnostiam. Kapitola vychádza z [15] a [11].

1.1 Základné pojmy a vlastnosti dynamických systémov

Definícia 1.1 (Stavový priestor)**.** Množinu všetkých stavov, v ktorých sa môže systém nachádzať nazývame *stavovým priestorom* a obyčajne označujeme X.

Definícia 1.2 (Evolučný operátor)**.** Nech T je číselná množina a nech pre dané $t \in T$ je v stavovom priestore X dané zobrazenie

$$\varphi^t : X \to X,$$

ktoré transformuje počiatočný stav systému $x_0 \in X$ v čase 0 do stavu $x_t \in X$ v čase t tak, že platí

$$x_t = \varphi^t x_0 \tag{1}$$

Takéto zobrazenie nazývame *evolučným operátorom* dynamického systému.

Evolučné operátory majú dve prirodzené vlastnosti. Prvou z nich je

$$\varphi^0 = \mathrm{id}, \tag{2}$$

kde id označuje identické zobrazenie na X. Pre všetky $x \in X$ platí $\mathrm{id}\, x = x$, čo znamená, že *stav systému sa spontánne nemení*. Druhou vlastnosťou evolučných operátorov je

$$\varphi^{t+s} = \varphi^t \circ \varphi^s. \tag{3}$$

To znamená, že

$$\varphi^{t+s} x = \varphi^t \left(\varphi^s x\right) \tag{4}$$

pre všetky $x \in X$ a $t, s \in T$ také, že obe strany rovnice (4) sú definované. Vlastnosť (3) v podstate hovorí, že výsledok evolúcie systému počas $(t + s)$ časových jednotiek,

začínajúcej v bode $x \in X$ je rovnaký, ako keby sme najprv nechali systém meniť sa zo stavu x iba počas s časových jednotiek a následne pokračovali zo stavu $\varphi^s x$ po dobu t časových jednotiek. Inými slovami, systém sa riadi podmienkami, ktoré nezávisia od času - je to *autonómny systém*.

Definícia 1.3 (Dynamický systém)**.** *Dynamický systém* je trojica $\{T, X, \varphi^t\}$, kde T je časová množina, X je stavový priestor a $\varphi^t : X \to X$ je jednoparametrická trieda evolučných operátorov parametrizovaná cez $t \in T$ a spĺňajúca podmienky (2) a (3).

Ak $T = \mathbb{R}$ a $\varphi^t x$ je hladká funkcia (x, t), hovoríme, že systém je *spojitý*.

Definícia 1.4 (Trajektória)**.** Trajektória so začiatkom v x_0 je usporiadaná podmnožina stavového priestoru X, ktorá spĺňa

$$O_r(x_0) = \left\{x \in X : x = \varphi^t x_0, \forall t \in T \text{ také, že } \varphi^t x_0 \text{ je definované}\right\}$$

Najjednoduchší druh trajektórií sú pevné body.

Definícia 1.5 (Pevný bod)**.** Bod $x^0 \in X$ nazývame *pevným bodom*, ak pre všetky $t \in T$ platí, že $\varphi^t x^0 = x^0$. Takýto bod nazývame tiež ekvilibrium, rovnovážny alebo stacionárny bod.

Definícia 1.6 (Cyklus)**.** *Cyklus* je neekvilibriová periodická trajektória L_0 spĺňajúca

$$\forall t \in T \; \exists \; T_0 > 0 : \forall x_0 \in L_0 \quad \varphi^{t+T_0} x_0 = \varphi^t x_0.$$

Najmenšie T_0, ktoré spĺňa túto vlastnosť sa nazýva *perióda* cyklu L_0.

Definícia 1.7 (Invariantná množina)**.** *Invariantná množina* dynamického systému $\{T, X, \varphi^t\}$ je podmnožina $S \subset X$ taká, že ak $x_0 \in S$, tak $\varphi^t x_0 \in S$ pre všetky $t \in T$.

Definícia 1.8 (Stabilita)**.** Invariantná množina S_0 sa nazýva *stabilnou*, ak

(i) pre ľubovoľné dostatočne malé okolie $U \supset S_0$ existuje okolie $V \supset S_0$ také, že $\varphi^t x \in U \; \forall x \in V$ a $\forall t > 0$

(ii) existuje okolie $U_0 \supset S_0$ také, že pre $t \to +\infty$ platí, že $\varphi^t x \to S_0 \; \forall x \in U_0$

Ak je S_0 pevný bod alebo cyklus, definícia je štandardnou definíciou stability pevných bodov resp. cyklov.

Vlastnosť (i) sa nazýva *Lyapunovská stabilita*. Ak je množina S_0 Lyapunovsky stabilná, okolné trajektórie nikdy neopustia jej okolie. Vlastnosť (ii) sa nazýva *asymptotická stabilita*. Existujú invariantné množiny, ktoré spĺňajú iba jednu z uvedených vlastností.

Definícia 1.9 (Fázový portrét)**.** Rozdelenie stavového priestoru na trajektórie nazývame *fázovým portrétom* dynamického systému.

Kým pre spojité jednorozmerné dynamické systémy je zobrazenie $t \to \varphi^t x_0$ monotónne, dvoj- a viacrozmerné systémy majú bohatšiu klasifikáciu kvalitatívneho správania. Východiskovým bodom pre preskúmanie kvalitatívneho správania je klasifikácia typov pevných bodov.

Uvažujme dvojrozmerný systém

$$\dot{x} = f(x), \quad x = (x_1, x_2)^T \in \mathbb{R}^2,$$

kde f je hladká spojitá funkcia. Nech $x = 0$ je pevný bod, $f(0) = 0$ a nech

$$A = \left.\frac{df(x)}{dx}\right|_{x=0}$$

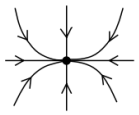
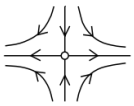
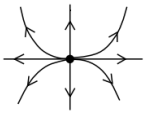
je jeho Jacobiho matica. Matica A má dve vlastné čísla λ_1, λ_2 - korene charakteristickej rovnice

$$\lambda^2 - \sigma\lambda + \Delta = 0,$$

kde σ je stopa a Δ determinant matice A. Vzhľadom na rôzne λ_1,λ_2 môžeme definovať niekoľko druhov fázových portrétov a ich stabilitu. Tieto sú zhrnuté v tabuľke 1. Grafy zobrazujúce fázové portréty boli prevzaté z [11].

Definícia 1.10 (Homeomorfizmus)**.** *Homeomorfizmom* nazývame bijektívne spojité zobrazenie h, pre ktoré platí, že aj jeho inverzia h^{-1} je spojité zobrazenie.

Definícia 1.11 (Topologická ekvivalencia)**.** Dynamický systém $\{T, \mathbb{R}^n, \varphi^t\}$ nazývame *topologicky ekvivalentný* k dynamickému systému $\{T, \mathbb{R}^n, \psi^t\}$, ak existuje homeomorfizmus $h : \mathbb{R}^n \to \mathbb{R}^n$, ktorý zobrazuje trajektórie prvého systému na trajektórie druhého systému pri zachovaní smeru plynutia času.

Vlastné hodnoty	Fázový portrét		Stabilita
$Im\lambda_1, Im\lambda_2 = 0 \wedge Re\lambda_1, Re\lambda_2 < 0$		uzol	stabilný
$Im\lambda_1, Im\lambda_2 \neq 0 \wedge Re\lambda_1, Re\lambda_2 < 0$		ohnisko	stabilné
$Im\lambda_1, Im\lambda_2 = 0 \wedge Re\lambda_1 < 0 < Re\lambda_2$		sedlo	nestabilné
$Im\lambda_1, Im\lambda_2 = 0 \wedge Re\lambda_1, Re\lambda_2 > 0$		uzol	nestabilný
$Im\lambda_1, Im\lambda_2 \neq 0 \wedge Re\lambda_1, Re\lambda_2 > 0$		ohnisko	nestabilné

Tabuľka 1: Prehľad druhov pevných bodov

1.2 Bifurkácie

Pod pojmom bifurkácia rozumieme v netechnickej reči rozdelenie čohosi na dve časti. Môže to byť miesto, v ktorom sa rozvetvuje rieka [10]. V medicíne je to miesto, kde sa napríklad priedušnica rozdeľuje na dve priedušky [8]. V matematickom kontexte však má bifurkácia oveľa konkrétnejšiu definíciu.

Uvažujme spojitý dynamický systém závislý od parametrov

$$\dot{x} = f(x, \alpha), \tag{5}$$

kde $x \in \mathbb{R}^n$ sú fázové premenné a $\alpha \in \mathbb{R}^m$ parametre. Uvažujme fázový portrét systému (5). Ak zmeníme hodnotu parametra α, zmení sa aj tento fázový portrét. Existujú dve možnosti - buď je nový fázový portrét topologicky ekvivalentný s pôvodným, alebo došlo k zmene topológie.

Definícia 1.12 (Bifurkácia)**.** Ak pre dynamický systém (5) existuje parameter $\alpha = \alpha_0$ taký, že pre α_1 ľubovoľne blízke k α_0 nie sú k nim prislúchajúce fázové portréty topologicky ekvivalentné, hovoríme, že nastala *bifurkácia*.

Hodnotu α_0 parametra α, v ktorom nastala zmena topológie systému nazývame *bifurkačnou hodnotou*. Na zobrazenie bifrukácií využívame tzv. *bifurkačný diagram*.

Definícia 1.13 (Bifurkačný diagram)**.** *Bifurkačný diagram* dynamického systému je stratifikácia (rozvrstvenie) priestoru parametrov spolu so zodpovedajúcimi fázovými portrétmi každej vrstvy.

Ak má dynamický systém jedno- alebo dvojrozmerný fázový priestor a závisí iba od jedného parametra, jeho bifurkačný diagram môže byť zobrazený ako priamy produkt fázového a parametrického priestoru $\mathbb{R} \times \mathbb{R}$ resp. $\mathbb{R}^2 \times \mathbb{R}$, s fázovými portrétmi zobrazenými ako jedno- respektíve dvojrozmerné vrstvy $\alpha = \text{konšt.}$.

V najjednoduchších prípadoch je priestor parametrov zložený z konečného počtu oblastí v $\mathbb{R}^m$. Vnútri každej z týchto oblastí je fázový portrét úlohy topologicky ekvivalentný. Oblasti sú oddelené *bifurkačnými hranicami*, čo sú hladké variety v $\mathbb{R}^m$ (napríklad krivky alebo plochy). Tieto hranice sa môžu pretínať alebo splývať a ich prieniky rozdeľujú bifurkačné hranice na podoblasti. Bifurkačná hranica je definovaná fázovým objektom (ekvilibirium, cyklus a podobne) a *bifurkačnou podmienkou*, ktorá určuje typ bifurkácie (napríklad Hopfova bifurkácia, transkritická bifurkácia). Bifurkačná podmienka je obvykle viazaná na vlastnosti vlastných čísel Jacobiho matice systému.

Definícia 1.14 (Kodimenzia)**.** Rozdiel medzi dimenziou priestoru parametrov a dimenziou korešpondujúcej bifurkačnej hranice nazývame *kodimenziou* bifurkácie v systéme (5).

Kodimenziu môžeme ekvivalentne definovať ako počet nezávislých podmienok, ktoré vymedzujú bifurkáciu. Kodimenzia jedného typu bifurkácie je rovnaká pre všetky generické systémy závislé od dostatočného počtu parametrov.

Minimálny počet voľných parametrov potrebných na vytvorenie bifurkácie kodimenzie k je práve k. Opačne, v generickom systéme s k parametrami stačí sledovať bifurkácie kodimenzie maximálne k. Ak sa v m-rozmernom systéme nachádza bifurkácia kodimenzie k ($m > k$), tak vieme nájsť k-rozmernú bifurkačnú hranicu, v ktorej sa táto bifurkácia nachádza. Ak urobíme bijektívne zobrazenie na iný podpriestor dimenzie k, tak pôvodný bifurkačný diagram a jeho obraz budú topologicky ekvivalentné.

Pre lokálne (a niektoré globálne) bifurkácie stacionárnych bodov preto existujú univerzálne bifurkačné diagramy, získané z topologicky normálnych tvarov.

Niekedy je možné zostrojiť jednoduchý systém polynomiálny v ξ_i

$$\dot{\xi} = g\left(\xi, \beta; \sigma\right), \quad \xi \in \mathbb{R}^k, \sigma \in \mathbb{R}^l, \tag{6}$$

ktorý v $\beta = 0$ dosahuje ekvilibrium $\xi = 0$ spĺňajúce k bifurkačných podmienok určujúcich pre toto ekvilibrium bifurkáciu kodimenzie k. Vektor σ je vektor koeficientov σ_i, $i = 1, 2, ..., l$ polynómov z (6). Vo všetkých prípadoch, ktorými sa budeme zaoberať, bude v priestore parametrov konečne veľa oblastí zodpovedajúcich topologicky neekvivalentným fázovým priestorom. V najjednoduchšom prípade budú všetky σ_i dosahovať konečne veľa rôznych celočíselných hodnôt.

Spolu so systémom (6) uvažujme systém

$$\dot{x} = f\left(x, \alpha\right), \quad x \in \mathbb{R}^n, \alpha \in \mathbb{R}^k, \tag{7}$$

ktorý má pre $\alpha = 0$ pevný bod $x = 0$.

Definícia 1.15 (Genericita)**.** Systém, ktorý spĺňa konečne veľa *generických podmienok* sa nazýva *generický systém*. Generické podmienky majú tvar nerovností

$$N_i\left[f\right] \neq 0, \quad i = 1, 2, ..., s,$$

kde každé N_i je nejaká algebraická funkcia niektorej parciálnej derivácie od $f\left(x, \alpha\right)$ vzhľadom na x a α v bode $(x, \alpha) = (0, 0)$.

Definícia 1.16 (Topologicky normálny tvar)**.** Dynamický systém (6) sa nazýva *topologicky normálnym tvarom* bifurkácie, ak pre ľubovoľný generický systém (7) s pevným bodom $x = 0$ spĺňajúcim v $\alpha = 0$ rovnaké bifurkačné podmienky je systém (6) blízko počiatku pre nejaké hodnoty koeficientov σ_i lokálne topologicky ekvivalentý k (7).

Ak je zostrojený topologicky normálny tvar bifurkácie, jeho bifurkačný diagram má univerzálny význam, pretože sa objavuje v bifurkačných diagramoch generických systémov.

Definícia 1.17 (Indukovaný systém)**.** Hovoríme, že systém

$$\dot{y} = g\left(y, \beta\right), \quad y \in \mathbb{R}^n, \beta \in \mathbb{R}^m,$$

je *indukovaný* systémom

$$\dot{x} = f(x, \alpha), \quad x \in \mathbb{R}^n, \alpha \in \mathbb{R}^m,$$

ak $g(y, \beta) = f(y, p(\beta))$, kde $p : \mathbb{R}^m \to \mathbb{R}^m$ je spojité zobrazenie.

Zobrazenie p pri tom nemusí byť homeomorfizmom, takže nemusí byť invertovateľné.

Definícia 1.18 (Verzálna deformácia)**.** Systém (6) je *verzálnou deformáciou* korešpondujúcej lokálnej bifurkácie, ak ľubovoľný systém (7) s pevným bodom v $x = 0$, spĺňajúci rovnaké podmienky v $\alpha = 0$, je blízko počiatku lokálne topologicky ekvivalentný k indukovanému systému (6) pre nejaké hodnoty koeficientov σ_i.

Dá sa dokázať, že vo veľa prípadoch sú topologicky normálne formy, ktoré odvádzame, v skutočnosti verzálnymi deformáciami pre korešpondujúce bifurkácie. Takéto bifurkácie sa nazývajú *generické*. Ďalej sa v tejto kapitole budeme venovať iba úlohám, ktoré obsahujú generické bifurkácie.

1.2.1 Bifurkácia typu sedlo-uzol

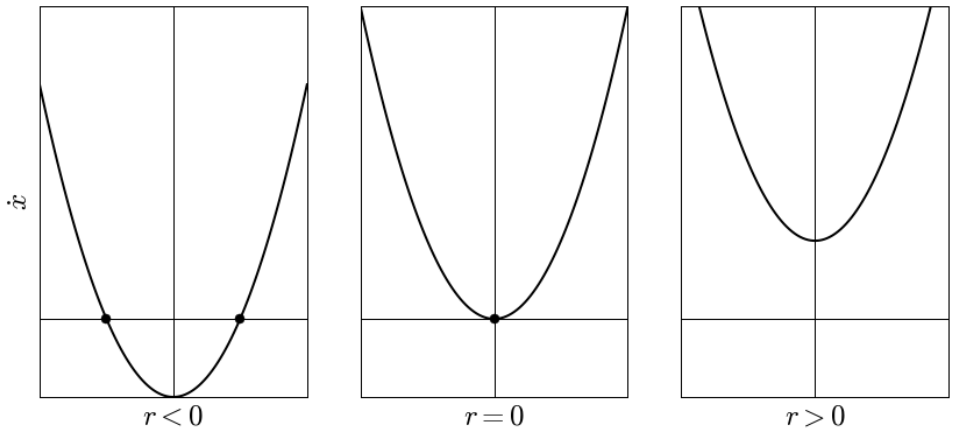
Jedným z najjednoduchších typov bifurkácií je bifurkácia typu sedlo-uzol, v niektorej literatúre označovaná aj ako tzv. *fold bifurcation* alebo *out-of-blue-sky bifurcation*. Je to preto, že z fázového priestoru, v ktorom sa nevyskytoval žiaden stacionárny bod, sa malou zmenou bifurkačného parametra jeden pevný bod objaví, aby sa ihneď rozdelil na dva, jeden stabilný a jeden nestabilný. Ak sa na dynamiku pozrieme z opačnej strany, môžeme ju interpretovať ako dva pevné body s rôznou stabilitou, ktoré sa navzájom priťahujú, aby nakoniec anihilovali a zmizli.

Bifurkáciu typu sedlo-uzol môžeme dobre ukázať na konkrétnom prípade, pretože ide o generickú formu bifurkácie [11].

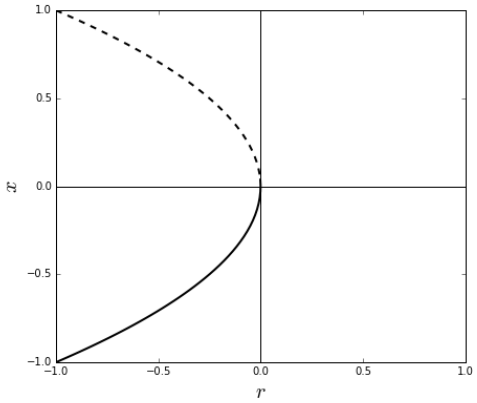
Majme obyčajnú diferenciálnu rovnicu s parametrom

$$\dot{x} = r + x^2. \tag{8}$$

Pre rôzne hodnoty parametra r existuje rozličné množstvo pevných bodov, ako vidno aj na Obrázku 1.



Obr. 1: Riešenia rovnice (8) pre rôzne hodnoty parametra r



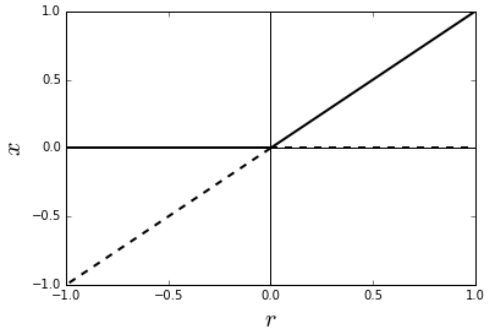
Obr. 2: Bifurkácia typu sedlo-uzol - Bifurkačný diagram úlohy (8)

1.2.2 Transkritická bifurkácia

Transkritická bifurkácia nastáva v situácii, že nelineárny systém má dva pevné body. Jeden z nich je stabilný, druhý nestabilný a blížia sa k sebe, v bifurkačnom bode sa stretnú a pokračujú ďalej vo svojich vytýčených trajektóriách, ale s vymenenou stabilitou/nestabilitou.

Bifurkácia je jednoducho ilustrovateľná napríklad na úlohe

$$\dot{x} = rx - x^2. \tag{9}$$

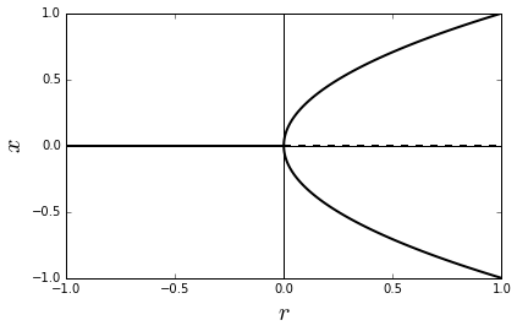


Obr. 3: Transkritická bifurkácia - Bifurkačný diagram úlohy (9).

1.2.3 Vidlicová bifurkácia

Vidlicová bifurkácia (z angl. *pitchfork*) nastáva, keď do bifurkačného bodu existuje iba jediný pevný bod, ktorý sa v bode bifurkácie rozvetví na 3 časti. Podľa toho, či bol pevný bod stabilný alebo nestabilný rozoznávame superkritickú, respektíve subkritickú vidlicovú bifurkáciu. Ak je bifurkácia superkritická, zo stabilného pevného bodu sa rozvetvia dve stabilné vetvy a tretí pevný bod v pôvodnom smere sa zmení na nestabilný. Ak je pôvodný pevný bod nestabilný, (ne)stabilitu ostatných vetiev je možné analogicky obrátiť. Príkladom superkritickej vidlicovej bifurkácie môže byť nasledovná diferenciálna rovnica s parametrom r:

$$\dot{x} = rx - x^3. \tag{10}$$



Obr. 4: Vidlicová bifurkácia - Bifurkačný diagram úlohy (10).

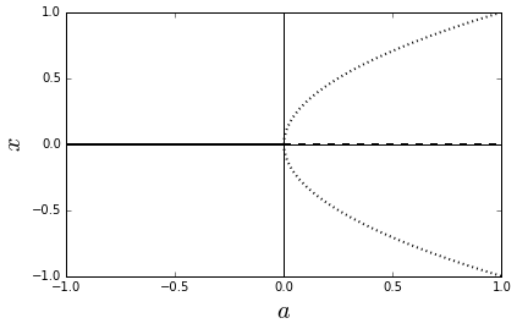
1.2.4 Hopfova bifurkácia

Hopfova bifurkácia je jednou z najdôležitejších nových typov bifurkácií, ktoré vznikajú v priestore s dimenziou $n \geq 2$. Nastáva, ak vo fázovom priestore zo špirály so stabilným

alebo nestabilným ohniskom vznikne v bode bifurkácie limitný cyklus. V prípade, že vzniknutý limitný cyklus je nestabilný a pevný bod pre rovnaké hodnoty bifuračného parametra stabilný, nazývame bifurkáciu superkritickou. Ak je stabilita vymenená, bifurkácia je subkritická. Uvažujme dvojrozmerný systém s parametrom a

$$\begin{aligned} \dot{x} &= -y + x\left(a - x^2 - y^2\right) \\ \dot{y} &= x + y\left(a - x^2 - y^2\right). \end{aligned} \tag{11}$$

Bifurkačný diagram úlohy (11), konkrétne jeho priemet do roviny $y = 0$ sa nachádza na obrázku 5. Plnou čiarou je označený stabilný pevný bod, prerušovanou čiarou nestabilný pevný bod a bodkovaná čiara označuje maximálnu a minimálnu hodnotu x dosahovanú v limitnom cykle zodpovedajúcom k danej hodnote parametra a.



Obr. 5: Hopfova bifurkácia - Bifurkačný diagram úlohy (11).

Existujú ďalšie typy bifurkácií, avšak na účely tejto práce nám postačujú bifurkácie popísané v tejto podkapitole. Na hlbšie pochopenie hoklinických, heteroklinických a iných bifurkácií a ich správania odporúčame [15] a [11].

2 Úvod do metód numerickej kontinuácie

Zámerom tejto kapitoly je priblíženie algoritmov, na základe ktorých funguje program Auto-07p, ktorý budeme používať v nasledujúcich kapitolách. Neuvádzame preto zoznam všetkých druhov problémov, ktoré sa riešia kontinuačnými metódami, ani do podrobností neporovnávame jednotlivé druhy metód. Uvádzame teoretické základy, vďaka ktorým je možné riešiť úlohy numerickej kontinuácie a princípy fungovania vybraných metód, ktoré sa na riešenie týchto úloh používajú. Vychádzali sme z knihy [2] a učebných textov [4] a [5].

2.1 Základné princípy numerickej kontinuácie

Uvažujme systém nelineárnych rovníc

$$F(x) = 0, \tag{12}$$

kde $F : \mathbb{R}^n \to \mathbb{R}^n$ je, pre jednoduchú ilustráciu princípov, hladké zobrazenie. To znamená, že jeho derivácia ľubovoľného rádu je spojitá. Ak existuje regulárne riešenie $\bar{x}$ úlohy (12), teda $\det\left(F'\left(\bar{x}\right)\right) \neq 0$ a poznáme jeho dobrú aproximáciu x_0, môžeme $\bar{x}$ vypočítať použitím *Newtonovej metódy*

$$x_{i+1} = x_i - \left[F'\left(x_i\right)\right]^{-1} F\left(x_i\right), \quad i = 0, 1, 2..., \tag{13}$$

alebo pomocou iného algoritmu Newtonovho typu

$$x_{i+1} = x_i - A_i^{-1} F\left(x_i\right), \quad i = 0, 1, 2..., \tag{14}$$

kde A_i je nejaká vhodná aproximácia Jakobiánu $F'\left(x_i\right)$. Výhodou tohto prístupu je rýchla konvergencia k riešeniu, nevýhodou je, že musia platiť vyššie uvedené predpoklady.

Ďalej uvažujme nelineárny systém s parametrom

$$H(x, \lambda) = 0, \quad x \in \mathbb{R}^n, \lambda \in \mathbb{R}, \tag{15}$$

kde o $H : \mathbb{R}^{n+1} \to \mathbb{R}^n$ máme nasledujúce predpoklady

Predpoklad 2.1. $H : \mathbb{R}^{n+1} \to \mathbb{R}^n$ je hladké zobrazenie.

Predpoklad 2.2. Existuje bod $u_0 \in \mathbb{R}^{n+1}$ taký, že

(i) $H(u_0) = 0$,

(ii) Jacobiho matica $H'(u_0)$ má maximálnu hodnosť - $\operatorname{rank}(H'(u_0)) = n$.

Za týchto predpokladov môžeme zvoliť index $i \in \{1, 2, ..., n+1\}$ taký, že podmatica Jacobiho matice $H'(u_0)$, získaná z $H'(u_0)$ odstránením i-teho stĺpca, nie je singulárna. Z Vety o implicitnej funkcii potom vieme, že množinu bodov, ktoré spĺňajú $H(x, \lambda) = 0$ môžeme lokálne parametrizovať vzhľadom na i-tu súradnicu [2]. Reparametrizáciou tejto množiny potom dostávame krivku c, ktorá spĺňa nasledovnú lemu.

Lema 2.3. *Za predpokladov 2.1 a 2.2 existuje hladká krivka $\alpha \in J \mapsto c(\alpha) \in \mathbb{R}^{n+1}$ pre nejaký otvorený interval J, $0 \in J$, taký, že pre všetky $\alpha \in J$ platí*

(i) $c(0) = u_0$,

(ii) $H(c'(\alpha)) = 0$,

(iii) $rank(H'(c'(\alpha))) = n$,

(iv) $c'(\alpha) \neq 0$.

Definícia 2.4 (Kolmý vektor)**.** Nech A je matica rozmeru $n \times (n+1)$ s hodnosťou $\operatorname{rank}(A) = N$. Potom vektor $t(A) \in \mathbb{R}^{(n+1)}$ spĺňajúci podmienky

(i) $At = 0$

(ii) $\|t\|$=1

(iii) $\det \begin{pmatrix} A \\ t^T \end{pmatrix} > 0$,

sa nazýva *kolmý vektor indukovaný* A.

Definícia 2.5 (Počiatočná úloha)**.** Počiatočná úloha je úloha v tvare

$$\dot{u} = t(H'(u)) \tag{16}$$

$$u(0) = u_0. \tag{17}$$

Pravá strana rovnice (16) je definovaná iba pre také body u, že Jacobiho matica $H'(u)$ má plnú hodnosť. Z tejto podmienky vychádza nasledovná definícia.

Definícia 2.6 (Regulárny bod a regulárna hodnota)**.** Nech $f : \mathbb{R}^p \to \mathbb{R}^q$ je hladké zobrazenie. Bod $x \in \mathbb{R}^p$ sa nazýva *regulárnym bodom* f, ak Jakobiho matica $f'(x)$ má plnú hodnosť $\min\{p, q\}$. Hodnota $y \in \mathbb{R}^q$ sa nazýva *regulárnou hodnotou* f, ak všetky $x \in \mathbb{R}^p$ pre ktoré platí $f(x) = y$ sú regulárne body f.

2.2 Metódy typu prediktor-korektor

Myšlienka metód typu prediktor-korektor je numericky sledovať krivku c z úvodu podkapitoly 2.1 tak, že pozdĺž nej generujeme postupnosť bodov u_i, $i = 1, 2, ...$, ktoré spĺňajú vybrané tolerančné kritérium, napríklad $\|H(u_i)\| \leq \varepsilon$ pre zvolené $\varepsilon > 0$. Pod označením $\|a\|$ rozumieme Euklidovskú normu v n-rozmernom priestore $\|a\| = \sqrt{\sum_{i=1}^{n} a_i^2}$, $a \in \mathbb{R}^n$. Predpokladáme, že je zadaný regulárny štartovací bod $u_0 \in \mathbb{R}^{n+1}$ taký, že $H(u_0) = 0$.

Je intuitívne a v [2] dokázané, že pre dostatočne malé $\varepsilon > 0$ existuje jediná hodnota parametra s_i taká, že platí $s_i = \underset{s \in J}{\operatorname{argmin}} \|c(s) - u_i\|$, teda že bod $c(s_i)$ je zo všetkých bodov na krivke c najbližšie k u_i.

Na ilustráciu postupu, ktorým sa generujú body u_i pozdĺž krivky c predpokladajme, že bod $u_i \in \mathbb{R}^{n+1}$ spĺňa $\|H(u_i)\| \leq \varepsilon$. Ak u_i je regulárny bod H, potom, ako sme ukázali v časti 2.1 existuje jedinečná krivka $c_i : J \to \mathbb{R}^{n+1}$ definovaná na maximálnom možnom intervale existencie J, ktorá spĺňa počiatočnú úlohu

$$\begin{aligned} \dot{u} &= t(H'(u)) \\ u(0) &= u_i. \end{aligned} \tag{18}$$

Na dosiahnutie nového bodu u_{i+1} pozdĺž krivky c najprv urobíme krok nazývaný *prediktor*. Ako prediktor sú typicky používané jednoduché explicitné metódy, často je používaný tzv. Eulerov prediktor

$$v_i = u_i + ht(H'(u_i)), \tag{19}$$

kde $h > 0$ je veľkosť kroku. Nasledujúci korekčný krok, nazývaný tiež *korektor* má vďaka skutočnosti, že krivka riešení c spĺňa rovnicu $H(u) = 0$, veľkú účinnosť. Aj pre slabý odhad prediktora v_{i+1} bude mať iteratívny korekčný proces rýchlu konvergenciu ku krivke riešení c. Na ilustráciu situácie nech w_{i+1} je bod na krivke c najbližšie k v_{i+1}.

Tento bod je riešením optimalizačného problému

$$\|w_{i+1} - v_{i+1}\| = \min_{H(w)=0} \|w - v_{i+1}\| . \tag{20}$$

Ak u_i je dostatočne blízke krivke c a krok h je dostatočne malý, potom bude bod predikcie v_{i+1} dostatočne blízko krivky c a minimalizačný problém (20) bude mať jednoznačné riešenie w_{i+1}. Očividný spôsob získania dobrého odhadu w_{i+1} je použitie metódy Newtonovho typu. Predpokladajme, že pomocou jednej až dvoch iterácií takejto metódy získame bod u_{i+1} aproximujúci w_{i+1} s toleranciou $\|H(u_{i+1})\| \leq \varepsilon$. Bod u_{i+1} potom považujeme za ďalší bod pozdĺž krivky c a celý postup opakujeme.

Na zostrojenie efektívnej a robustnej metódy typu prediktor-korektor, ktorou je možné dobre aproximovať zložité krivky, je potrebné vziať do úvahy niekoľko dôležitých bodov, medzi inými aj

- prispôsobovanie veľkosti kroku h,
- implementácia korekčného algoritmu,
- zahrnutie prediktora vyššieho rádu,
- zaobchádzanie so špeciálnymi bodmi, napríklad lokálnymi extrémami alebo bifurkačnými bodmi a ich aproximácia.

Týmito problémami sa ďalej zaoberá [2]. Je potrebné poznamenať, že kontinuačné metódy typu prediktor-korektor sa značne odlišujú od rovnomenných metód na numerickú integráciu počiatočných úloh. Zatiaľ čo prediktory oboch druhov metód sú si podobné, korektor v kontinuačných metódach je založený na kontraktívnosti množiny riešení $\{u \in \mathbb{R}^{n+1} : H(u) = 0\}$, preto tu môžeme využívať iteračné metódy ako napríklad Newtonovu metódu. Túto vlastnosť však vo všeobecnosti krivky riešení počiatočných úloh nespĺňajú, ich korektory limitne konvergujú k bodu, ktorého presnosť závisí na veľkosti kroku h.

2.3 Po častiach lineárne metódy

Zatiaľ čo metódy typu prediktor-korektor približne sledovali presné riešenie krivky c z podklapitoly 2.1, po častiach lineárne metódy (z angl. *piecewise-linear methods*) presne

sledujú po častiach lineárnu krivku c_τ, ktorá je aproximáciou c. Na presné popísanie tejto krivky potrebujeme definovať niekoľko pojmov.

Definícia 2.7 (Simplex)**.** Nech $v_1, v_2, ..., v_{j+1} \in \mathbb{R}^{n+1}$, $j \leq n+1$ sú afínne nezávislé body, čiže také, že vektory $v_k - v_1, k = 2, 3, ..., j+1$ sú lineárne nezávislé. Konvexný obal $[v_1, v_2, ..., v_{j+1}]$ množiny $\{v_1, v_2, ..., v_{j+1}\}$ sa nazýva *j-simplex* v $\mathbb{R}^{n+1}$. Konvexný obal $[w_1, w_2, ..., w_{k+1}]$ ľubovoľnej podmnožiny $\{w_1, w_2, ..., w_{k+1}\} \subset \{v_1, v_2, ..., v_{j+1}\}$ je tiež simplex a nazýva sa *stenou simplexu* $[v_1, v_2, ..., v_{j+1}]$.

Definícia 2.8 (Triangulácia)**.** *Triangulácia* τ v $\mathbb{R}^{n+1}$ je rozdelenie priestoru $\mathbb{R}^{n+1}$ na $(n+1)$-simplexy tak, že platí

(i) prienik ľubovoľnej dvojice simplexov v τ je buď stena týchto simplexov alebo prázdna množina

(ii) ľubovoľná ohraničená množina v $\mathbb{R}^{n+1}$ má prienik iba s konečným počtom simplexov v τ

Definícia 2.9 (Po častiach lineárna aproximácia)**.** Pre ľubovoľné zobrazenie $H : \mathbb{R}^{n+1} \to \mathbb{R}^n$ *po častiach lineárna aproximácia* H_τ k H príbuzná k triangulácii τ v $\mathbb{R}^{n+1}$ je zobrazenie jednoznačne definované vlastnosťami

(i) $H_\tau(v) = H(v)$ pre všetky vrcholy τ

(ii) pre ľubovoľný $(n+1)$-simplex $\sigma = [v_1, v_2, ..., v_{j+1}] \in \tau$ je zúženie $H_\tau|_\sigma$ afínne zobrazenie.

Dôsledkom toho je, že ak $u = \sum_{i=1}^{n+2} \alpha_i v_i$ je bod v σ, tak pre koeficienty α_i platí $\sum_{i=1}^{n+2} \alpha_i = 1$ a $\alpha_i > 0$ pre $i = 1, 2, ..., n+2$. Zároveň v dôsledku afinity H_τ vieme, že

$$H_\tau(u) = H\left(\sum_{i+1}^{n+2} \alpha_i v_i\right) = \sum_{i+1}^{n+2} \alpha_i H(v_i).$$

Množina $\{u \in \mathbb{R}^{n+1} : H(u) = 0\}$ obsahuje lomenú čiaru $c_\tau : \mathbb{R} \to \mathbb{R}^{n+1}$ aproximujúcu c. Sledovanie tejto lomenej čiary sa deje pomocou krokov podobných tým v metódach lineárneho programovania, ako je napríklad simplexová metóda.

Zároveň tieto metódy možno kombinovať s metódami typu prediktor-korektor. Napríklad môžeme vrcholy častiach lineárnej krivky c_τ aproximujúcej c považovať za prediktor a ako korektor použiť opäť iteračnú metódu Newtonovského typu.

Je vhodné poznamenať, že v prípade po častiach spojitých kontinuačných metód nie je potrebný predpoklad na hladkosť zobrazenia H.

Špecifickými druhmi problémov, s ktorými sa stretávame pri riešení kontinuačných úloh je detekcia rôznych druhov bifurkačných bodov, hľadanie periodických riešení, detekcia bifurkačných vetiev a prepínanie medzi jednotlivými vetvami. Z dôvodu rozsiahlosti tejto problematiky sa im v práci nevenujeme, avšak v prípade záujmu čitateľa sú, rovnako ako podrobnosti o vyššie popísaných metódach, rozobraté v [2], [4], [5] a čiastočne v [11].

3 Python a Auto-07p

V tejto kapitole sa zaoberáme softvérovými riešeniami použitými na riešenie kontinuačných úloh. V krátkosti popíšeme jednotlivé programy, vysvetlíme postup ich inštalácie a na záver priblížime význam jednotlivých častí Auto-07p.

3.1 Python

Python [13] je objektovo orientovaný interaktívny programovací jazyk s dynamickou sémantikou. Existuje od roku 1991 a dnes podporuje širokú škálu knižníc, balíčkov a modulov a je vhodný na skriptovanie aj spájanie kódu z rôznych programovacích jazykov do jedného celku. Až na veľmi špecifické prípady je bezplatný pre všetky hlavné operačné systémy - Windows, Unix/Linux i MacOS. V tejto práci nám Python, konkrétne verzia 2.7.10 pre Windows, umožňuje flexibilnejšie používanie výstupov z Auto-07p. V podkapitole 3.3 sa budeme venovať postupu inštalácie.

3.2 Auto-07p

Auto [1] je softvér na riešenie kontinuačných a bifurkačných problémov obyčajných diferenciálnych rovníc, pôvodne vytvorený v roku 1976 E. Doedelom. Za dobu jeho existencie sa na ňom podieľala rada ďalších vedcov a momentálne ôsma, najaktuálnejšia, je verzia Auto-07p z roku 2007.

Auto-07p dokáže urobiť čiastočnú bifurkačnú analýzu úloh algebraických systémov v tvare

$$f(u, \lambda) = 0, \quad f, u \in \mathbb{R}^n \tag{21}$$

a systémov obyčajných diferenciálnych rovníc v tvare

$$u'(t) = f(u(t), \lambda), \quad f, u \in \mathbb{R}^n \tag{22}$$

s počiatočnými podmienkami, okrajovými podmienkami, a integrálnymi väzbami. Pod označením λ rouzmieme jeden alebo viac parametrov. Auto-07p tiež dokáže riešiť niektoré kontinuačné a evolučné úlohy parabolických parciálnych diferenciálnych rovníc v tvare

$$u_t = Du_{xx} + f(u, \lambda), \quad f, u \in \mathbb{R}^n, \tag{23}$$

kde D označuje konštantnú diagonálnu maticu. Súčasťou Auto-07p je tiež softvér HOMCONT, určený na bifurkáciu homoklinických trajektórií.

3.3 Inštalácia

Uvedený softvér bol nainštalovaný a používaný na osobnom počítači s operačným systémom Windows, verzia 8.1. Pretože úkony potrebné na spustenie Auto-07p nezodpovedali návodu uvedenému v dokumentácii [3], uvádzame celý postup inštalácie. Pri inštalácii nám čiastočne pomohol návod v [7].

1. **Stiahnutie a inštalácia Pythonu:** My sme použili balík WinPython, 64-bitovú verziu získaný z https://sourceforge.net/projects/winpython/files/WinPython_2.7/2.7.10.3/. Výhodou WinPythonu je, že v jednej inštalácii obsahuje niekoľko prostredí na používanie jazyka Python a najpoužívanejšie balíčky ako NumPy, SciPy a iné, ktoré by sme ináč museli sťahovať samostatne. Dôležité je stiahnuť verziu, ktorá nie je označená *zero*, v tom prípade spomínané balíčky neobsahuje.

2. **Stiahnutie a inštalácia MinGW:** Ďalším krokom je inštalácia MinGW, open-source prostredia na vývoj aplikácií v operačnom systéme Windows z https://sourceforge.net/projects/mingw/. Je potrebné nainštalovať MinGW so **všetkými** komponentami, ktoré sú pri inštalácii ponúknuté - konkrétne sú to C Compiler, C++ Compiler, Fortran Compiler, ObjC Compiler, MSYS Basic System a MinGW Developer ToolKit.

3. **Spustenie MinGW:** Vyskúšame, či funguje MinGW. V priečinku `C:/MinGW/msys/1.0` by sa mal nachádzať súbor `msys.bat`. Spustíme ho a otvorí sa nám takzvaný *shell*, uživateľské prostredie, ktoré simuluje prostredie Linuxu. Mal by obsahovať zápis vo forme `meno_používateľa@meno_počítača` a reagovať na Linux príkazy, ako napríklad **`pwd`** na zistenie aktuálneho adresára alebo **`ls`** na vypísanie súborov, ktoré sa v aktuálnom adresári nachádzajú. Plná adresa aktuálneho adresára `msys` je `C:/MinGW/msys/1.0/home/meno_používateľa`. Prostredie `msys` zatiaľ nezatvárame.

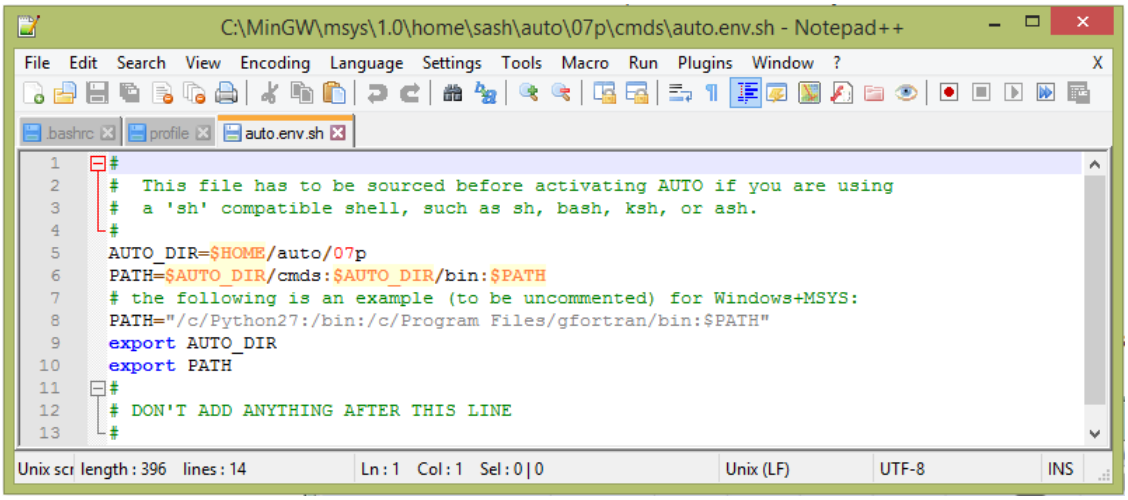
4. **Stiahnutie Auto-07p:** Stiahneme aktuálnu verziu Auto-07p z https://sourceforge.net/projects/auto-07p/. V našom prípade to bola verzia Auto-07p 0.9.1.

5. **Inštalácia Auto-07p:** Stiahnutý súbor `auto07p-0.9.1.tar.gz` presunieme do aktuálneho adresára z kroku č. 3 - `C:/MinGW/msys/1.0/home/meno_používateľa`. V prostredí `msys` zopakujeme príkaz **`ls`**, ktorý nám potvrdí, že súbor `auto07p-0.9.1.tar.gz` sa nachádza v priečinku. Postupne zavoláme nasledujúce príkazy

   - **`qunzip auto07p-0.9.1.tar.gz`**
   - **`tar xvfo auto07p-0.9.1.tar`** - Výsledkom je priečinok `auto` obsahujúci ďalší priečinok `auto/07p`
   - **`cd auto/07p`** - Presunieme sa do tohto priečinku
   - **`./configure`** - Pripravíme Auto na kompiláciu. Pravdepodobne sa vypíše upozornenie, že nebolo možné nainštalovať vybrané grafické knižnice. Keďže však máme v pláne používať Auto spolu s Pythonom, nie je to pre nás problém.
   - **`make`** - Skompilujeme Auto-07p
   - **`make clean`** - Odstránime nepotrebné príkazy

6. **Úpravy Auto-07p:** Kým spustíme Auto-07p je potrebné urobiť nasledujúce úpravy:

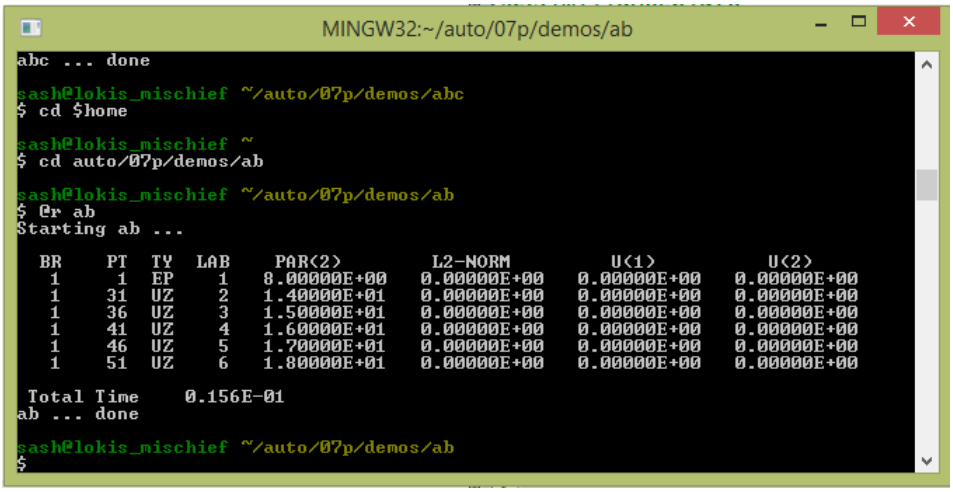
   - **`cd $HOME`** prípadne **`cd ../..`** - Vrátime sa do domovského priečinka
   - **`touch .bashrc`** - Vytvoríme skrytý súbor s názvom `.bashrc`.
   - Otvoríme tento súbor vo vybranom textovom editore napíšeme do neho reťazec **`source /home/meno_užívateľa/auto/07p/cmds/auto.env.sh`**. Miesto `meno_používateľa` je potrebné použiť konkrétne používateľské meno. Súbor uložíme.
   - V textovom editore otvoríme súbor s názvom `profile` nachádzajúci sa v priečinku `C:/MinGW/msys/1.0/etc`, na jeho koniec pripíšeme textový reťazec `. ~/.bashrc` a súbor uložíme.
   - Otvoríme v textovom editore súbor `auto.env.sh`, ktorý sa nachádza v priečinku **`C:/MinGW/msys/1.0/meno_užívateľa/auto/07p/cmds`**. Odkomentujeme ôsmy riadok odstránením znaku # na jeho začiatku tak, aby vyzeral ako súbor na obrázku 6.



Obr. 6: Upravený súbor `auto.env.sh`

- **`exit`** - zatvoríme shell a necháme načítať vykonané zmeny.

7. Overíme úspešnosť inštalácie: Opäť otvoríme prostredie **`msys`**, presunieme sa do ukážkových úloh Auto-07p, napríklad na model `ab`, príkazom **`cd auto/07p/demos/ab`** a zavoláme **`@r ab`**. Výsledkom by mal byť výstup podobný tomu na obrázku 7.



Obr. 7: Výstup demo programu v Auto-07p

3.4 Používané súbory a ich význam

Výstup kontinuačných úloh je zapísaný v troch štandardných výstupných súboroch a vypísaný na obrazovku v okne, v ktorom je spustené Auto-07p. Ak neurčíme inak, na obrazovku sú vypísané iba špeciálne body riešenia. Druhy týchto špeciálnych bodov sa

nachádzajú v tabuľke 2. Na obrazovku sa vypisujú pod uvedenými dvojpísmenovými kódmi, vo výstupných súboroch `fort.7` a `fort.8` figurujú pod kódmi číselnými.

Skratka	**Kód**	**Typ riešenia**
BP	(1)	Bod rozvetvenia (algebraické systémy)
LP	(2)	Bifurkačný bod typu sedlo-uzol (algebraické systémy)
HB	(3)	Hopfova bifurkácia
	(4)	Bod výstupu špecifikovaný užívateľom
UZ	(-4)	Výstup na užívateľom určenej hodnote parametra
LP	(5)	Bifurkačný bod typu sedlo-uzol (diferenciálne rovnice)
BP	(6)	Bod rozvetvenia (diferenciálne rovnice)
PD	(7)	Zdvojenie periódy
TR	(8)	Bifurkácia na tore
EP	(9)	Koncový bod; normálne ukončenie
MX	(-9)	Abnormálne ukončenie; nekonvergentné riešenie

Tabuľka 2: Typy špeciálnych bodov riešenia úlohy

Výstupný súbor `fort.7` obsahuje všetky body bifurkačného diagramu. Jeho formát je rovnaký ako formát výstupu na obrazovku, avšak obsahuje všetky vyrátané body riešenia.

Výstupný súbor `fort.8` obsahuje všetky údaje o vybraných označených riešeniach. Dôvodom je možnosť odštartovať ďalšiu kontinuáciu aj z týchto špecifikovaných bodov. Býva rozsiahlejší než `fort.7`.

Výstupný súbor `fort.9` obsahuje diagnostické hlásenia, históriu konvergencie a vlastné hodnoty. Obsah tohto súboru je odporúčané pravidelne kontrolovať.

Na špecifikáciu úlohy, ktorú chceme vyriešiť slúžia dva druhy vstupných súborov. Súbor formátu `.f90` (napríklad `xxx.f90`), obsahujúci rovnice a k nemu prislúchajúci súbor konštánt `c.xxx`. Uvádzame účely, na ktoré sa konštanty používajú a funkciu vybraných druhov konštánt. Konštanty majú niekoľko rôznych účelov:

- definujú rozmery riešenej úlohy
  - `NDIM` - rozmer systému rovníc

  - `NPAR` - maximálny počet parametrov

- definujú požadovanú prednosť numerického riešenia
  - `EPSL` - relatívne kritérium konvergencie pre parametre
  - `ITMX` - maximálny počet iterácií povolených na nájdenie špeciálneho bodu
- definujú veľkosť kontinuačného kroku
  - `DS` - dĺžka prvého kroku
  - `DSMAX` - maximálna veľkosť kroku (`DSMAX`> 0)
  - `DSMIN` - minimálna veľkosť kroku (`DSMIN`> 0)
- určujú hranice kontinuačného diagramu
  - `STOP` - konštanta určuje špeciálne body (ich druh a poradie), na ktorých sa má kontinuácia zastaviť
  - `RL0` - horná hranica hlavného kontinuačného parametra
  - `RL1` - dolná hranica hlavného kontinuačného parametra
- riadia formu výstupu
  - `unames` - vektor priraďujúci mená jednotlivým stavovým premenným
  - `IPLT` - určuje druh vektorovej normy
  - `IID` - určuje množstvo diagnostického výstupu, ktoré sa zapíše do súboru `fort.9`
- výpočtové konštanty, ktoré určujú aký druh úlohy chceme rátať a ktoré špeciálne body chceme hľadať
  - `ILP` - binárna konštanta, ktorá určuje, či detekujeme bifurkačné body typu sedlo-uzol
  - `IPS` - definuje o aký druh problému sa jedná

Úplný prehľad konštánt a ich hodnôt je zahrnutý v [3] a vybraným konštantám sa venujeme v nasledujúcej kapitole.

4 Príklady použitia Auto-07p

Na jednoduchých príkladoch ilustrujeme, ako sa používa Auto-07p v spolupráci s Pythonom a ako je možné interpretovať výstupy z týchto programov.

4.1 Ekológia - Jednorozmerná bifurkácia s jedným parametrom

Uvažujme populáciu živočícha alebo rastliny, ktorej správanie popisuje obyčajná diferenciálna rovnica

$$\dot{x} = rx\left(1 - \frac{x}{K}\right) - \alpha. \tag{24}$$

Úloha je prevzatá z [11], kde bola použitá ako príklad na prerátanie. Závislá premenná x označuje stav populácie v čase t, konštanta r je prirodzená úroveň jej rastu a K je zaťažiteľnosť populácie. Parameter α je riadiacim parametrom bifurkácie a označuje úroveň lovu, respektíve zberu. Môžeme predpokladať, že keď α presiahne istú hranicu, stabilný stav populácie sa kvalitatívne zmení. Vzhľadom na jednoduchosť úlohy by bolo možné ju vyriešiť bez použitia softvéru, avšak my na ňom ukážeme základy používania Auto-07p v spolupráci s Pythonom.

4.1.1 Zadanie úlohy do vstupných súborov

Prvým krokom k vyriešeniu úlohy je jej zadanie do vstupných súborov Auto07p. Súbory upravujeme v ľubovoľnom textovom editore. Rovnice úlohy (24) by mali byť zapísané v súbore `pr1.f90` v jazyku Fortran, respektíve v analogických súboroch vo Fortrane 77 (`pr1.f`) alebo v jazyku C (`pr1.c`). Konštanty použité k vyriešeniu úlohy by mali byť zapísané v súbore `c.pr1`. Najjednoduchšie je upraviť súbory z niektorej z ukážkových úloh Auto 07p v `auto/07p/demos`.

Upravíme preto súbory skopírované z `auto/07p/demos/ab`. Súbor `pr1.f90` obsahuje niekoľko subrutín. Najprv sa budeme venovať subrutine `FUNC`, ktorá má obsahovať diferenciálnu rovnicu (24). Vo Fortrane je potrebné dodeklarovať všetky používané premenné, v našom prípade to bude vyzerať nasledovne:

```
DOUBLE PRECISION x, alpha, r, K
```

Následne je potrebné určiť ich typ - konštanta, bifurkačný parameter alebo stavová premenná. Stavová premenná sa označuje ako `U(i)` a bifurkačný parameter ako `PAR(j)`, kde $i, j = 1, 2, 3, \ldots$ Konštantám stačí priradiť želanú hodnotu.

```
x = U(1)
alpha = PAR(1)
r = 3
K = 1
```

Hodnoty r a K sme zvolili bez ujmy na všeobecnosti. Pri všeobecnejšom riešení úlohy by sme aj ich mohli definovať ako bifurkačné parametre, tomu sa však budeme venovať v ďalších častiach tejto kapitoly.

Poslednou časťou subrutiny `FUNC` je zápis diferenciálnych rovníc. K premennej `U(i)` prislúcha pravá strana diferenciálnej rovnice `F(i)`.

```
F(1) = r*x*(1 - x/K) - alpha
```

Do subrutiny `STPNT` doplníme počiatočný stav parametrov a stavových premenných v stacionárnom riešení, teda tak, aby boli derivácie podľa všetkých stavových premenných 0. Keďže môžeme meniť aj počiatočnú hodnotu bifurkačných parametrov, takýchto riešení je nekonečne veľa a konkrétny výber možno urobiť tak, aby mal čo najjednoduchší algebraický tvar. V našom príklade sa zdá prirodzené začať s populáciou, do ktorej nezasahujú žiadne vonkajšie vplyvy a teda $\alpha = 0$, čo implikuje, že horný stacionárny bod rovnice (24) je daný $x = 1$. Obsah subrutiny `STPNT` doplníme nasledovne:

```
PAR(1) = 0
U(1)   = 1
```

Ďalej upravíme konštanty v súbore `c.pr1`. Tento súbor používa Auto-07p na to, aby vedel, čo konkrétne má s informáciami v `pr1.f90` robiť.

Konkrétny význam parametrov je v krátkosti uvedený v kapitole 3 a do podrobností v [4]. Väčšina parametrov môže ostať taká, aká bola pri úlohe v `auto/07p/demos/ab` z ktorej sme vychádzali. Je potrebné skontrolovať, či sedia počty a názvy stacionárnych premenných,

```
NDIM =   1  # number of variables
unames = {1:x}
```

aj názvy a počty bifurkačných parametrov. V prípade, že je parametrov viac, je potrebné pomocou `ICP` určiť, ktorý z nich má byť použitý na kontinuáciu. Úloha (24) má iba jeden bifurkačný parameter a preto existuje iba jediná možnosť.

```
NPAR =   1  # (maximal) number of parameters
parnames = {1:a}
ICP =  [1]  # bifurcation parameters. Note: 11 is reserved for the period
   ↪  of periodic solutions.
```

Skontrolujeme aj ohraničenie hodnôt, ktoré môže bifurkačný parameter nadobúdať.

```
RL0 =   -1  # lower bound on the principal continuation parameter
RL1 =    1  # upper bound on the principal continuation parameter
```

Určíme, či hľadáme stacionárne riešenia obyčajných diferenciálnych rovníc, alebo periodické riešenia Hopfovej bifurkácie a na záver povolíme detekciu tzv. foldov, teda bifurkácií typu sedlo-uzol.

```
IPS =   1   # 1: stationary solutions of ODEs with detection of Hopf; 2:
   ↪ computation of periodic solutions from a Hopf bifurcation or a
   ↪ periodic orbit from a previous run
ILP =   1   # 0/1: off/on detection of folds
```

Oba súbory uložíme a pokračujeme ku kontinuácii cez Auto-07p prostredníctvom Pythonu.

4.1.2 Spustenie programu

Kombináciu Auto-07p a Pythonu je možné realizovať z ľubovoľnej platformy, z ktorej je možné spúšťať Python kód. Môže to byť Windows príkazový riadok, prípadne nejaký druh integrovaného vývojového prostredia (napr. IDLE, IPython, Spyder, Jupyter, ...) alebo úplne iná alternatíva. My sme pracovali v prostredí Spyder, preto nasledujúce postupy a riešenia nemusia byť platné univerzálne. Spyder bol súčasťou balíčka WinPython, ktorú sme nainštalovali v kapitole 3.

Prvým krokom po spustení zvoleného prostredia je naimportovať doňho Auto-07p. Teoreticky by malo byť možné ho naimportovať po jednoduchom príkaze `import auto`, avšak nám v tomto prípade vyhlasoval chybu `ImportError: No module named auto`. Bolo potrebné nastaviť cestu k priečinkom s Auto-07p a prvý kód do súboru `plot_pr1.py` je nasledovný:

```
import sys
auto_directory="C:/MinGW/msys/1.0/home/meno_používatela/auto/07p"
sys.path.append(auto_directory+'/python')
import auto
```

Na mieste `meno_používateľa` sa vyskytne konkrétne meno používateľa použitého počítača. V tomto momente by malo byť možné spustiť kontinuáciu úlohy `pr1`, predtým ale treba nastaviť aktuálny adresár na ten, v ktorom sa nachádzajú vstupné súbory. Použijeme na to príkaz `cd` v tvare

```
cd C:/Users/meno_používatela/.../meno_priečinka
```

V tomto momente môžeme spustiť kontinuáciu príkazom

```
pr1 = auto.load('pr1')
b1 = auto.r('pr1')
```

V prípade, že výstupom tohto príkazu `b1 = auto.r('pr1')` je iba `gfortran -O -c pr1.f90 -o pr1.o` a nie je možné volať žiadne ďalšie príkazy, je pravdepodobné, že zvolená aplikácia nevie vytvoriť alebo prepísať objekt `pr1.o` a prepojiť ho spolu so

vstupnými súbormi na spustiteľný súbor `pr1.exe`. Vtedy sa v aktuálnom adresári nachádzajú iba vstupné súbory a súbor s názvom `pr1.o`. Postup, ktorý sa nám v takejto situácii osvedčil, je nasledovný.

- Vymažeme súbor s príponou `.o` z priečinka so vstupnými súbormi
- Otvoríme konzolu `msys`, ktorá sa nachádza v adresári MinGW, konkrétne v `MinGW/msys/1.0`.
- Zmeníme aktuálny adresár rovnako, ako v prostredí Pythonu
- Zavoláme `@r pr1`

Posledný príkaz spustí kontinuáciu, vypíšu sa konkrétne body riešenia a do priečinku so vstupnými súbormi pribudnú okrem `pr1.o` aj výstupné súbory `fort.7`, `fort.8` a `fort.9` a aplikácia `pr1`. Teraz sa môžeme vrátiť naspäť do prostredia Pythonu.

V Spyderi zopakujeme postup naimportovania Auto-07p a spustenia kontinuácie riešenia našej úlohy. Výstup by však mal byť rovnaký ako v konzole `msys`.

4.1.3 Riešenie a interpretácia

V tomto momente bola na obrazovku vypísaná časť výstupu z Auto-07p.

```
In [9]: b1 = auto.r(pr1)
Starting pr1 ...

  BR    PT  TY  LAB       a          L2-NORM          x
   1     1  EP    1   0.00000E+00   1.00000E+00   1.00000E+00
   1   115  EP    2  -1.02386E+00   2.68951E-01  -2.68951E-01

 Total Time    0.312E-01
pr1 ... done
Note: The following floating-point exceptions are signalling: IEEE_UNDERFLOW_FLAG
```

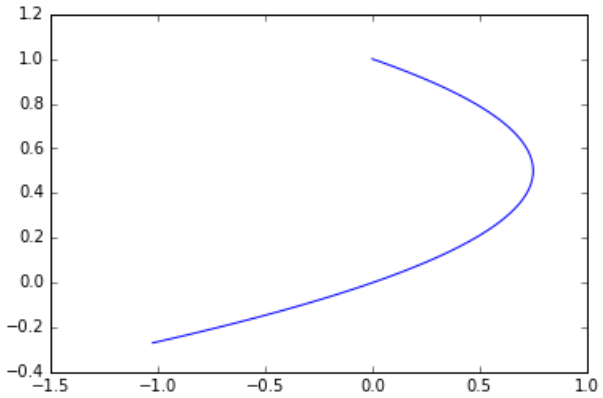
Obr. 8: Výstup z Auto-07p v prostredí Spyder

Vidíme počiatočný a konečný bod bifurkácie. Stĺpec `BR` označuje, o akú vetvu riešenia ide - v našom prípade existuje iba jediná vetva. Stĺpec `PT` označuje poradové číslo zobrazeného bodu dosiahnutého kontinuáciou. `TY` označuje typ bodu, ktorý vidíme a môže mať niekoľko rôznych hodnôt, podrobnejšie popísaných v [4] a v 3.4. Konkrétne `EP` je skratkou pre ending point, čiže koncový bod.

Všetky body kontinuácie sú zapísané vo `fort.7`. Ukážme si, ako sa mení stacionárny stav x v závislosti od parametra α. Importujeme do Pythonu metódu `pyplot` z knižnice `matplotlib` a zobrazíme dvojice bodov x a α v grafe.

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots(1, figsize=(6,4))
ax.plot(b1[0]['a'],b1[0]['x'])
```

Výstup tohto kódu je na obrázku 9. Na x-ovej osi sa nachádza parameter α a kolmo naň sú vynesené stacionárne riešenia diferenciálnej rovnice (24).



Obr. 9: Prvý graf bifurkácie nakreslený v Pythone

Pretože pre príliš veľké α neexistuje žiaden stacionárny stav, pri $\alpha = 0,75$, $x = 0,5$ vzniká jeden bifurkačný bod a pre $\alpha < 0,75$ existujú vždy 2 stacionárne stavy, môžeme povedať, že ide o bifurkáciu typu sedlo-uzol.

Túto domnienku potvrdzuje aj skutočnosť, že ak vykonáme kontinuáciu rovnakej úlohy, iba s aktivovanou konštantou `ILP=1`, ktorá zodpovedá hľadaniu bifurkačných bodov typu sedlo-uzol, objaví sa vo výstupe aj bod typu `LP`, ktorý presne tieto druhy bifrukácií označuje. Môžeme požadovať, aby bol obrázok symetrický. V tom prípade máme 2 možnosti. Buď ohraničíme α tak, aby bolo nezáporné, alebo urobíme kontinuáciu riešení aj opačným smerom až po $\alpha = -1$. Tieto možnosti zavoláme ako

```
b2 = auto.r('pr1', RL0=0)
b3 = auto.r('pr1', DS='-')
```

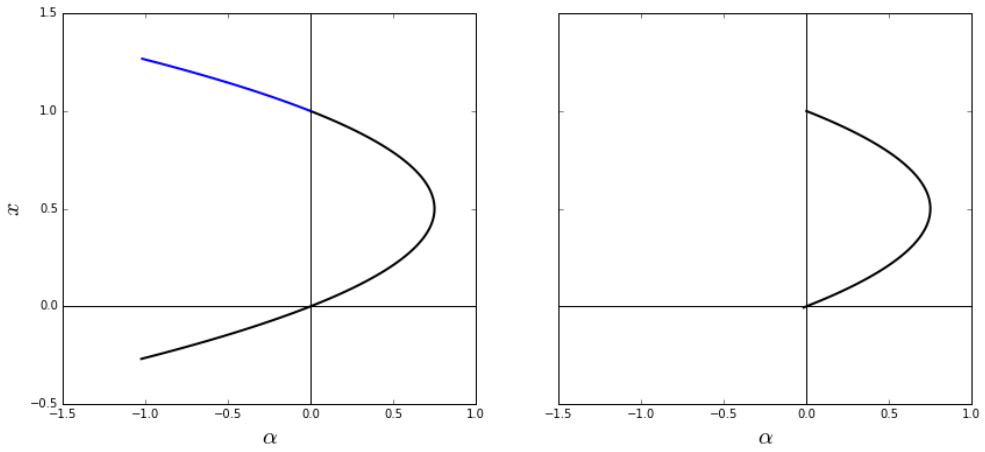
```
In [10]: b1 = auto.r(pr1, ILP=1)
Starting pr1 ...

  BR    PT  TY  LAB       a          L2-NORM          x
   1     1  EP    1   0.00000E+00   1.00000E+00   1.00000E+00
   1    44  LP    2   7.50000E-01   5.00000E-01   5.00000E-01
   1   115  EP    3  -1.00873E+00   2.65665E-01  -2.65665E-01

 Total Time    0.156E-01
pr1 ... done
Note: The following floating-point exceptions are signalling: IEEE_UNDERFLOW_FLAG
```

Obr. 10: Rozšírený výstup z Auto-07p v prostredí Spyder

kde `b2` zodpovedá zmene ohraničenia α a `b3` kontinuácii opačným smerom. `RL0` a `DS` sú parametre zo súboru `c.pr1`. Prvý spomenutý už bol vysvetlený, `DS` zodpovedá kroku numerickej kontinuácie. Pre porovnanie sú obe možnosti vykreslené na obrázku 11. Modrou je vyznačená časť, ktorú sme doplnili pri obrátení kroku kontinuácie. Na vykreslenie a úpravu formy grafov sme použili nasledovné príkazy.

```
fig2, (ax1, ax2) = plt.subplots(1,2, sharey=True, figsize=(14,6))
ax1.plot(b1[0]['a'],b1[0]['x'],'-k',lw=2)
ax1.plot(b3[0]['a'],b3[0]['x'],'-b',lw=2)
ax1.set_xlim(-1.5,1)
ax1.set_ylim(-0.5,1.5)
ax1.axhline(y=0, color='k')
ax1.axvline(x=0, color='k')
ax1.set_xlabel("$\\alpha$", fontsize=20)
ax1.set_ylabel("$x$", fontsize=20)

ax2.plot(b2[0]['a'],b2[0]['x'],'-k',lw=2)
ax2.set_xlim(-1.5,1)
ax2.set_ylim(-0.5,1.5)
ax2.axhline(y=0, color='k')
ax2.axvline(x=0, color='k')
ax2.set_xlabel("$\\alpha$", fontsize=20)
fig2.savefig('pr1-1.png')
```

Keďže ide o bifrukáciu typu sedlo-uzol, jedna vetva stacionárnych stavov má obsa-



Obr. 11: Dva spôsoby úpravy grafu

hovať stabilné a druhá nestabilné riešenia. Stabilita stacionárnych stavov je zaznačená vo výstupnom súbore `fort.7` ako znamienko pri poradovom čísle jednotlivých bodov v stĺpci `PT`. Ak je pred číslom −, ide o stabilné, ak +, tak nestabilné riešenie.

Aby sme tieto informácie zobrazili aj na grafe, potrebujeme sa k týmto znamienkam dopracovať. Spojíme preto kontinuácie `b1` a `b3` do jednej a uložíme ich.

```
bb = b1 + b3
#bb = auto.merge(bb)
#bb = auto.relabel(bb)
auto.save(bb, 'bb')
```

V aktuálnom adresári nám vzniknú ďalšie výstupné súbory - súbor `s.bb` s riešením, `d.bb` s diagnostikou úlohy a `b.bb` s bifurkačným diagramom. Obsah súboru `b.bb` načítame do premennej typu list a oddelíme názvy jednotlivých stĺpcov.

```
content = None
with open('b.bb', 'r') as f:
    content = f.readlines()

content_csv = [[el for el in content[17].split(' ') if len(el) > 0 and el
    ↪  != '\n']]
content_csv[0][0] = 'branch'
```

```
column_names = content_csv[0]
```

Pomocou for cyklu prejdeme cez celý súbor a zapíšeme hodnoty na zodpovedajúce miesta.

```
for line in content:
    dummy = line.split(' ')
    dummy = [el for el in dummy if len(el) > 0 and el != '\n']
    if dummy[0] == '0':
        continue

    for el_i, el in enumerate(dummy):
        if el_i < 4:
            dummy[el_i] = int(el)
        else:
            dummy[el_i] = float(el)

    if len(dummy) > 1:
        content_csv.append(dummy)
```

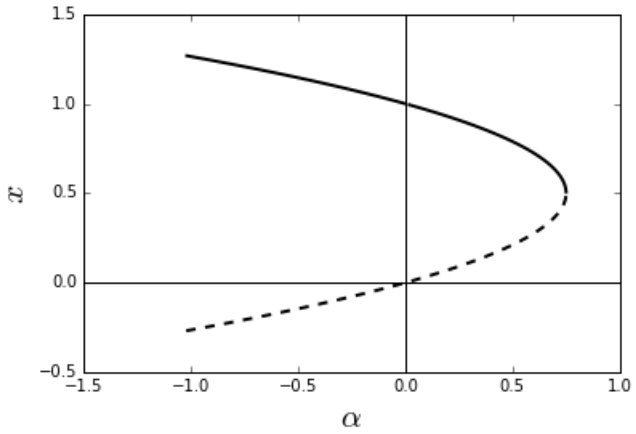
Naimportujeme knižnicu `pandas`, zoznam jednotlivých bodov prekonvertujeme na premennú typu `data frame` a pre lepšiu manipuláciu ju preindexijeme.

```
import pandas as pd

df = pd.DataFrame(content_csv, columns=column_names)
df = df[1:]
df.index=df.index-1
```

Premennú `df` už ľahko použijeme na vykreslenie bifurkačného diagramu aj so zobrazením stability riešení. Plná čiara na obrázku 12 označuje stabilné a prerušovaná nestabilné stacionárne body.

```
fig, ax = plt.subplots(1, figsize=(6,4))
ax.plot(df[(df.PT>0)].a,df[(df.PT>0)].x,'--k',lw=2)
```

```
ax.plot(df[(df.PT<0) & (df.a<=0)].a,df[(df.a<=0)&(df.PT<0)].x,'-k',lw=2)
ax.plot(df[(df.PT<0) & (df.a>0)].a,df[(df.a>0)&(df.PT<0)].x,'-k',lw=2)
ax.set_xlim(-1.5,1)
ax.set_ylim(-0.5,1.5)
ax.axhline(y=0, color='k')
ax.axvline(x=0, color='k')
ax.set_xlabel("$\\alpha$", fontsize=20)
ax.set_ylabel("$x$", fontsize=20)
fig.savefig('pr1-2.png')
```



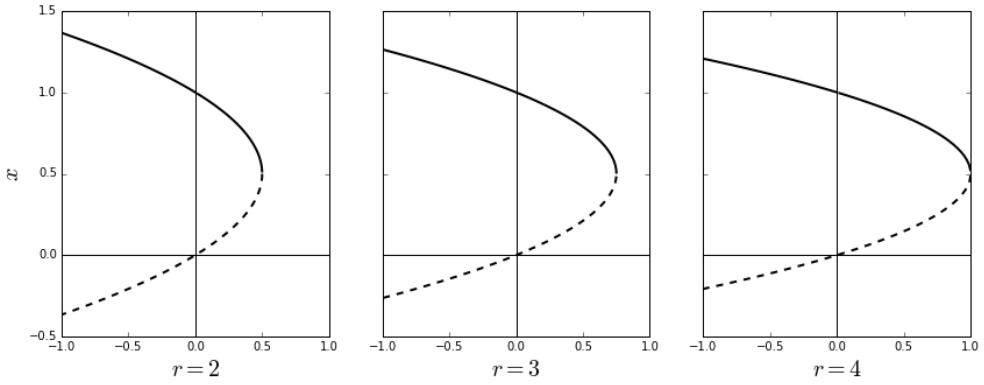
Obr. 12: Bifurkačný diagram úlohy (24) s vyznačenou stabilitou

Výsledný graf môžeme interpretovať tak, že ak je úroveň lovu α vyššia než 0.75, neexistuje pre danú populáciu žiadny stacionárny stav. Dá sa ľahko ukázať, že $\dot{x}$ je vždy záporné a tak podľa modelu $x \to -\infty$. Avšak podľa sedliackeho rozumu nemôže byť veľkosť populácie záporná, takže ak $x\,(t)$ dosiahne hodnotu 0, tak nebude ďalej klesať a ostane nulová.

Na druhú stranu miernejšia úroveň lovu umožňuje dosiahnutie dvoch rôznych stacionárnych bodov s rôznou stabilitou. Stabilný stacionárny stav populácie je vždy vyšší než nestabilný. Čím nižšia je úroveň lovu, tým je väčšia populácia v stabilnom stave. Tento trend pokračuje ďalej aj pre $\alpha < 0$, čiže v prípade, keď je populácia zväčšovaná z vonkajších zdrojov - sú vysádzané ďalšie rastliny alebo vypúšťané ďalšie zvieratá.

4.2 Ekológia - Bifurkácia s viacerými parametrami

Model uvedený v predchádzajúcej kapitole má ďalšie dva parametre, r a K, ktorých hodnotu sme si pevne stanovili. Ako by však vyzerala úloha ak by bola prirodzená úroveň rastu populácie r iná? Bifurkačné diagramy úlohy (24) s troma rôznymi hodnotami r sú na obrázku 13.



Obr. 13: Bifurkačný diagram úlohy (24) pre rôzne hodnoty r.

Vidíme, že čím vyššia je úroveň rastu populácie r, tým väčšia môže byť úroveň zberu/lovu α a zároveň existovať pevný bod pre populáciu x. Zároveň je veľkosť populácie v stabilnom stave bez vonkajších vplyvov ($\alpha = 0$) pre rôzne úrovne r rovnaká.

4.2.1 Zadanie úlohy do vstupných súborov

Zmeňme úlohu (24) tak, aby obsahovala dva bifurkačné parametre α a r. Skopírujeme súbory `pr1.f90` a `c.pr1` a uložíme ich ako `pr2.f90` a `c.pr2`.

V textovom editore zmeníme obsah `pr2.f90`. Vymažeme zo subrutiny `FUNC` riadok obsahujúci `r=3` a nahradíme ho `r=PAR(2)`. Do subrutiny `STPNT` doplníme podmienku `PAR(2)=3`.

V súbore `c.pr2` zmeníme hodnoty zodpovedajúce počtu a názvom bifurkačných parametrov. Prvé číslo v poradí vektora `ICP` označuje hlavný bifurkačný parameter, podľa ktorého je vykonávaná kontinuácia. Stačilo by napísať iba `ICP=1`, avšak ak použijeme tvar `ICP=[1,2]` tak sa vo výstupe objavia oba bifurkačné parametre, čo zvyšuje prehľadnosť.

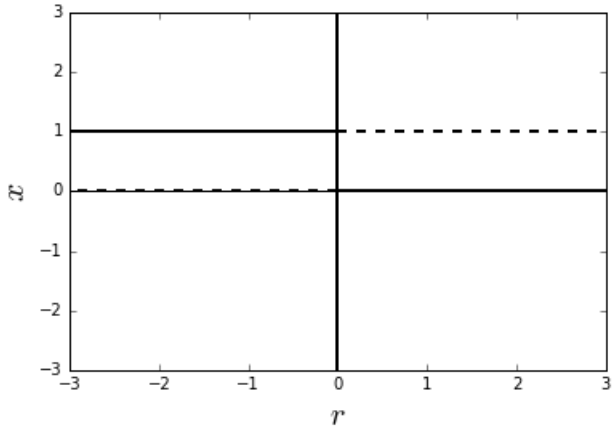
```
NPAR=   2   # (maximal) number of parameters
parnames= {1:a, 2:r}
ICP =  [1,2]  # bifurcation parameters. Note: 11 is reserved for the
   ↪ period of periodic solutions.
```

4.2.2 Riešenie a interpretácia

Najprv sa pozrieme, ako vyzerá bifurkácia podľa r pre $\alpha = 0$. Podľa tvaru úlohy (24) by malo ísť o triviálny problém, keďže bez ohľadu na veľkosť r sú pevné body úlohy rovnaké. Overíme to však aj v praxi.

```
b3=auto.run('pr2', ICP=[2,1], DS='-', RL0=-3)
```

Na grafe 14 vidíme, že skutočne pre akékoľvek nenulové r existujú iba 2 pevné body, $x = 1$ a $x = 0$. Pre $r = 0$ existuje nekonečne veľa riešení. Je to prirodzené, pretože v populácii s nulovým koeficientom prírastku a nulovou úrovňou lovu v našom modeli nedochádza k žiadnym zmenám. Ide však o značne degenerovaný prípad, preto sa po-



Obr. 14: Bifurkačný diagram úlohy (24) podľa r pre $\alpha = 0$.

zrieme na bifurkácie pre nenulové α. Prvým krokom je sa k takej hodnote dopracovať bez ďalšieho rátania. Použijeme bifurkáciu podľa α, ktorú poznáme z prechádzajúcej úlohy.

```
qq=auto.run('pr2', DS='-')
```

```
auto.save(qq,'qq')
qq2=auto.run('pr2')
auto.save(qq2,'qq2')
```

Bifurkácie nás dovedú do bodu, kde $\alpha = -1$. Následne odtiaľ oboma smermi vedieme bifurkáciu podľa parametra r.

```
w1=auto.run(qq,ICP=[2,1])
w2=auto.run(qq,ICP=[2,1], DS='-')
w3=auto.run(qq2,ICP=[2,1])
w4=auto.run(qq2,ICP=[2,1], DS='-')
```

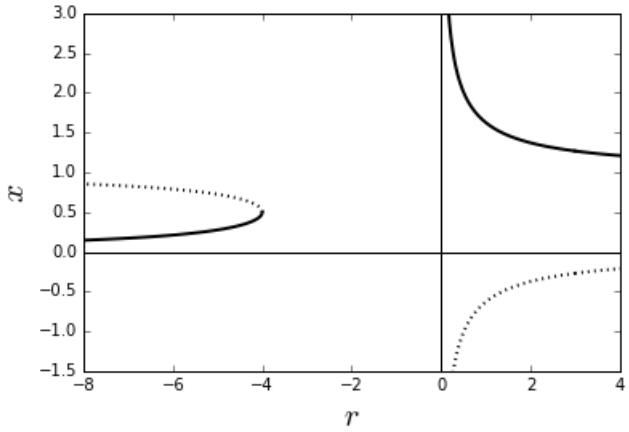
Ešte je potrebné preskúmať kontinuáciu pre záporné hodnoty parametra r. Prislúchajúce vstupné súbory, nazvané `pr22.f90` a `c.pr22`, vytvoríme skopírovaním súborov s úlohou `pr2`. Počiatočné hodnoty parametrov zmeníme na `PAR(1)=-1`, `PAR(2)=-4` a `U(1)=0.5`, inak ponecháme súbory nezmenené.

```
w5=auto.run('pr22', ICP=[2,1])
w6=auto.run('pr22', DS='-', ICP=[2,1])
```

Určíme stabilitu podľa hodnôt zaznačených vo výstupných súboroch a riešenie vykreslíme na grafe (15).

```
fig, (ax) = plt.subplots(1,1, figsize=(6,4))
ax.plot(w1['r'],w1['x'],'-k',lw=2)
ax.plot(w2['r'],w2['x'],'-k',lw=2)
ax.plot(w3['r'],w3['x'],':k',lw=2)
ax.plot(w4['r'],w4['x'],':k',lw=2)
ax.plot(w5['r'],w5['x'],':k',lw=2)
ax.plot(w6['r'],w6['x'],'-k',lw=2)
ax.axhline(y=0, color='k')
ax.axvline(x=0, color='k')
ax.set_xlabel("$r$", fontsize=20)
ax.set_ylabel("$x$", fontsize=20)
ax.set_xlim(-8,5)
```

```
ax.set_ylim(-1.5,3)
ax.axhline(y=0, color='k')
ax.axvline(x=0, color='k')
ax.set_xlabel("$r$", fontsize=20)
ax.set_ylabel("$x$", fontsize=20)
fig.savefig('pr2-1-2.png')
```



Obr. 15: Bifurkačný diagram úlohy (24) podľa r pre $\alpha = -1$.

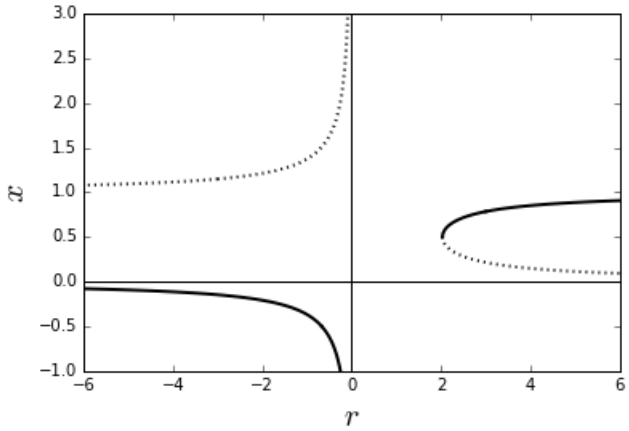
Bifurkačný diagram pre kladnú hodnotu α získame tak, že prvú bifurkáciu zastavíme skôr, než sa prehupne do záporných čisel. Keďže α je maximálne $0,75$, nastavíme horné ohraničenie bifurkačného parametra na menšiu hodnotu, napríklad na $0,5$. Dosiahneme to tak, že do volania `auto.run` pridáme parameter `RL1=0.5`.

```
qq3=auto.run('pr2', RL1=0.5)
auto.save(qq3, 'qq3')
qq4=auto.run('pr2', RL1=0.5, DS='-')
auto.save(qq3, 'qq3')
```

Ďalšie kroky sú analogické ako v prípade $\alpha < 0$. Výsledný bifurkačný diagram je zobrazený na obrázku 16.

Pri $r = 4\alpha$ dochádza k bifurkácii typu sedlo-uzol. Ak je úroveň rastu populácie v absolútnej hodnote štyrikrát väčšia než úroveň lovu, existujú dva pevné body, jeden stabilný a jeden nestabilný, limitne sa blížiace k hodnotám 1 a 0.

V prípade, že majú nenulové α a r opačné znamienka, existujú opäť dve riešenia s



Obr. 16: Bifurkačný diagram úlohy (24) podľa r pre $\alpha = 0.5$.

opačnou stabilitou. Pri rýchlosti rastu populácie limitne sa blížiacom k nule ($r \to 0$) blížia sa stacionárne stavy populácie do nekonečna, respektíve do mínus nekonečna. Opačne, ak sa rýchlosť rastu populácie blíži do nekonečna, resp. mínus nekonečna, stacionárne stavy populácie sa blížia k 0 alebo 1.

Pre nenulové α neexistuje pevný bod pre $r \in \langle 0, 4\alpha)$ ak $\alpha > 0$, respektíve pre $r \in (4\alpha, 0\rangle$ ak $\alpha < 0$. To znamená, že ak nie je úroveň rastu populácie štyrikrát väčšia než úroveň jej výlovu, neexistuje pre túto populáciu žiaden stacionárny stav a veľkosť populácie diverguje.

4.3 Chémia - Viacrozmerná bifurkácia s limitným cyklom

Uvažujme nasledovný teoretický model pre konkrétny typ autokatalytických chemických reakcií nazývaný Brusselator [12], [15].

$$\begin{aligned} A &\to X \\ B + X &\to Y + D \\ 2X + Y &\to 3X \\ X &\to E \end{aligned} \tag{25}$$

Názov modelu vznikol zložením slov oscilátor a Brusel, podľa Université libre de Bruxelles, kde bol navrhnutý. X, Y, A, B, D a E sú množstvá jednotlivých chemických látok. Za prepokladu, že A a B sú konštantné, D a E priebežne odstraňujeme zo systému a rýchlosť priebehu všetkých reakcií je konštantná, môžeme úlohu zbezrozmerniť

na systém dvoch diferenciálnych rovníc s dvoma parametrami ((27)).

$$\begin{aligned} \dot{u} &= 1-(b+1)u+au^2v \\ \dot{v} &= bu-au^2v \\ & a>0, b>0 \end{aligned} \tag{26}$$

My sa budeme venovať prípadu s parametrom $a = 1$. Úlohu sme prevzali z [12], kde bola zadaná ako úloha na precvičenie. My na nej ilustrujeme používanie Auto-07p na vyhľadávanie limitných cyklov.

4.3.1 Zadanie úlohy do vstupných súborov

Vstupné súbory opäť získame skopírovaním z niektorej z ukážkových úloh v `auto/07p/demos` alebo zo súborov použitých pri vypracovaní predchádzajúcich dvoch príkladov. Nazveme ich `pr3.f90` a `c.pr3`. Najprv zadáme do subrutiny `FUNC` v `pr3.f90` dynamický systém (27).

```
uu = U(1)
vv = U(2)
b  = PAR(1)
a  = 1

F(1) = 1 - (b+1)*uu + a*uu*uu*vv
F(2) = b*uu - a*uu*uu*vv
```

Do subrutiny `STPNT` doplníme začiatočný bod bifurkácie. Vybrali sme si hodnoty $u = 1$, $v = 1$ a $b = 1$.

```
PAR(1) = 1
U(1) = 1
U(2) = 1
```

Súbor `c.pr3` pozmeníme tak, aby bol v súlade so zadaním úlohy. Úloha obsahuje 2 premenné a 1 parameter, ktorý by nemal byť menší ako 0.

```
NDIM=   2   # number of variables
unames= {1:u, 2:v}
NPAR=   1   # (maximal) number of parameters
parnames= {1:b}
ICP =  [1]  # bifurcation parameters. Note: 11 is reserved for the period
   ↪  of periodic solutions.
RL0 =   0   # lower bound on the principal continuation parameter
RL1 =   4   # upper bound on the principal continuation parameter
```

Najprv budeme hľadať Hopfovu bifurkáciu, teda bod, kedy sa zmení stabilný pevný bod na nestabilný a okolo neho vznikne limitný cyklus.

```
IPS =   1   # 1: stationary solutions of ODEs with detection of Hopf; 2:
   ↪ computation of periodic solutions from a Hopf bifurcation or a
   ↪ periodic orbit from a previous run
ISP =   1   # 0: do not detect special solutions; 1: detect branch points
   ↪  and Hopf bifurcation; 2: detect all special solutions
```

4.3.2 Riešenie a interpretácia

Po zadaní úlohy do vstupných súborov prejdeme k samotnému riešeniu úlohy. Postup je analogický ako v časti 4.1. Vo zvolenom prostredí Pythonu zmeníme aktuálny adresár na ten, ktorý obsahuje vstupné súbory. Použijeme úvodnú sekvenciu kódu na pridanie Auto-07p a ďalších potrebných knižníc a spustíme kontinuáciu.

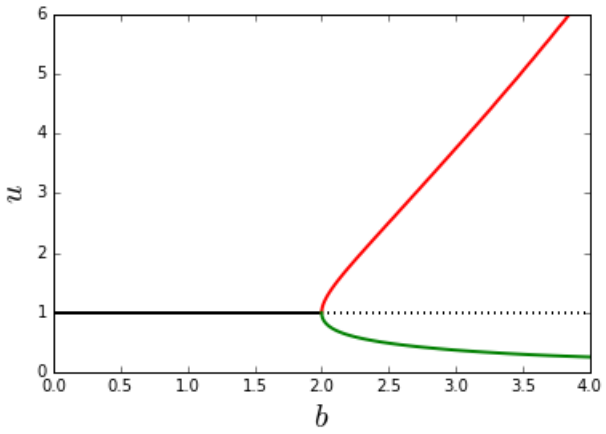
```
b1=auto.run('pr3')
b2=auto.run('pr3', DS='-')
```

V `b1` nastala Hopfova bifurkácia, označená vo výstupe ako `HB` v stĺpci `TY`. Z tohto bodu chceme hľadať periodické riešenia a ich periódu. Spustíme preto ďalšiu kontinuáciu, začínajúcu z bodu prvej Hopfovej bifurkácie v `b1`, ktorá hľadá periodické riešenia. Zároveň chceme hľadať nielen maximálne, ale aj minimálne hodnoty premenných, ktoré sa v konkrétnom limitnom cykle dosahujú.

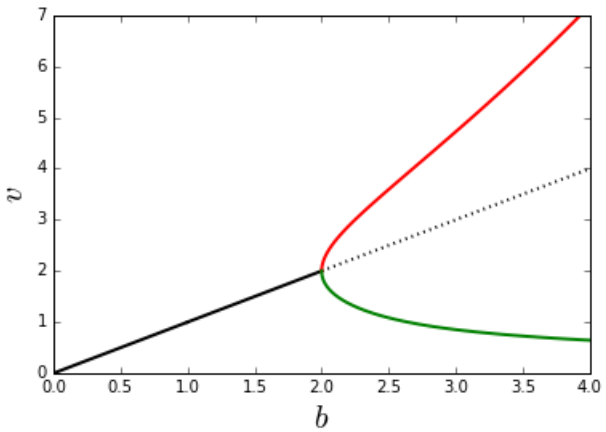
```
per=auto.run(b1('HB1'), ICP=[1,11],IPS=2, IPLT=-1)
per2=auto.run(b1('HB1'), ICP=[1,11],IPS=2, IPLT=-2)
```

Príkaz voláme dvakrát s rozdielnymi hodnotami `IPLT=-i`, pretože minimálna hodnota periodického riešenia je vyrátaná vždy iba pre i-tu premennú.

Vytvoríme priemety nájdených riešení do rovín rovnobežných s u a v. Uvádzame časť zdrojového kódu na vykreslenie grafu 17, postup pri 18 je analogický



Obr. 17: Bifurkačný diagram úlohy (27)



Obr. 18: Bifurkačný diagram úlohy (27)

```
fig, ax = plt.subplots(1, figsize=(6,4))
ax.plot(b1[0][:18]['b'],b1[0][:18]['u'],'-k',lw=2)
ax.plot(b1[0][17:]['b'],b1[0][17:]['u'],':k',lw=2)
```

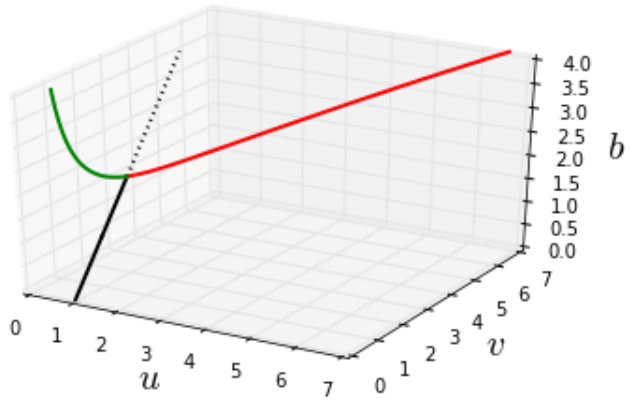
```
ax.plot(b2['b'],b2['u'],'-k',lw=2)

ax.plot(per[0]['b'],per[0]['MAX u'],'-r',lw=2)
ax.plot(per[0]['b'],per[0]['MIN u'],'-g',lw=2)

ax.set_xlim(0,4)
ax.set_ylim(0,6)
ax.axhline(y=0, color='k')
ax.axvline(x=0, color='k')
ax.set_xlabel("$b$", fontsize=20)
ax.set_ylabel("$u$", fontsize=20)
fig.savefig('pr3-1.png')
```

Na záver zobrazíme bifurkačný diagram v troch rozmeroch na grafe (19).

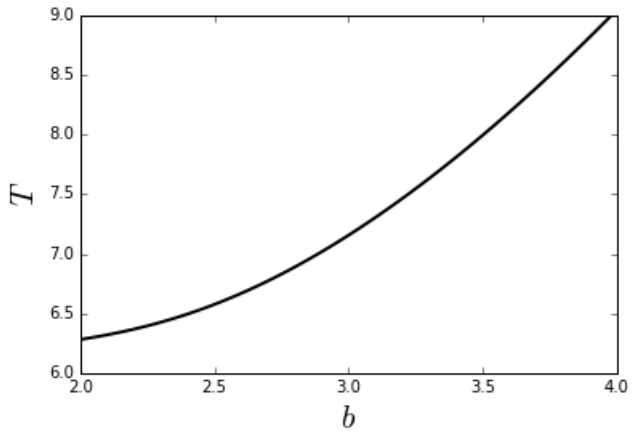
```
fig = plt.figure()
ax = fig.gca(projection='3d')
ax.plot(b1[0][:18]['u'],b1[0][:18]['v'],b1[0][:18]['b'],'-k',lw=2)
ax.plot(b1[0][17:]['u'],b1[0][17:]['v'],b1[0][17:]['b'],':k',lw=2)
ax.plot(b2['u'],b2['v'],b2['b'],'-k',lw=2)
ax.plot(per[0]['MAX u'],per2[0]['MAX v'],per[0]['b'],'-r',lw=2)
ax.plot(per[0]['MIN u'],per2[0]['MIN v'],per[0]['b'],'-g',lw=2)

ax.set_xlim(0,7)
ax.set_ylim(0,7)
ax.set_zlim(0,4)
ax.set_xlabel("$u$", fontsize=20)
ax.set_ylabel("$v$", fontsize=20)
ax.set_zlabel("$b$", fontsize=20)
plt.subplots_adjust(bottom=0.15,right=0.95)
fig.savefig('pr3-3.png')
```

Vidíme, že Hopfova bifurkácia, ktorá v úlohe nastala je superkritická, teda limitný cyklus k sebe riešenia priťahuje a pevný bod v jeho vnútri ich odpudzuje. To znamená, že chemická reakcia pre $b \geq 2$ sa v tomto modeli neustáli v jednom konkrétnom stave, ale neustále sa cyklicky mení.

Podľa výstupných súborov vieme povedať, že v blízkosti bifurkačného bodu $b = 2$



Obr. 19: Bifurkačný diagram úlohy (27).



Obr. 20: Závislosť periódy limitného cyklu od parametra b

je perióda limitného cyklu veľkosti 2π a následne sa zväčšuje, ako vidno na grafe 20. Rýchlosť oscilačnej premeny jednotlivých látok sa teda s rastúcim parametrom b stále spomaľuje.

5 Aplikácia - Model jednej neurónovej skupiny

V tejto kapitole aplikujeme teoretické poznatky z predchádzajúcich častí práce na zjednodušený model spánkového cyklu. Cieľom a jedným z vlastných prínosov práce je namodelovať a následne preskúmať riešenia, ku ktorým sa vďaka použitému softvéru dopracujeme. Ako napovedá už názov kapitoly, model bude obsahovať limitné cykly. Našim zámerom bude identifikovať bifurkačné body, zobraziť ich a interpretovať. Vychádzať pri tom budeme z modelu navrhnutého v [14] a z poznatkov zhrnutých v [6] a [9].

Namodelujeme správanie jednej skupiny neurónov s rovnakou funkciou. Následne ukážeme, ako sa pri zmene vstupného signálu do populácie neurónov mení ich aktivita.

5.1 Model jednej neurónovej skupiny

V ľudskom mozgu sa nachádza viacero druhov nervových buniek, neurónov. Všetky nervové bunky sa však skladajú z tela (soma), vstupných a výstupných výbežkov. Vstupné výbežky, dendrity, vytvárajú dendrický strom alebo ide o tzv. bazálne dendrity. Každý neurón ma iba jeden výstupný výbežok nazývaný axón, obalený izolujúcou myelínovou vrstvou, na konci rozvetvený na tisíce výbežkov, ktoré sú zakončené terminálmi.

Spojenie dvoch neurónov, ktoré umožňuje prenos vzruchu z presynaptického terminálu jedného neurónu, cez synaptickú štrbinu, do postsynaptickej membrány druhého neurónu, sa nazýva synapsia. Signály, ktoré neuróny prijímajú a vysielajú môžu byť buď chemického alebo elektrického charakteru. Elektrické signály, nazývané tiež akčné potenciály, môžu mať kladný alebo záporný náboj. Podľa toho sa prenášajú buď excitačnými (kladné), alebo inhibičnými (záporné) synapsiami. Signály, ktoré nesú informáciu o svetle nie sú kvalitatívne odlišné od signálov o tlakových vlnách alebo o pachu. Hlavným rozdielom medzi nimi je cesta, po ktorej sa pohybujú - skupiny neurónov, ktoré sú do ich šírenia mozgom zapojené.

Uvažujme skupinu neurónov s rovnakou funkciou. Nech parameter I_{tot} označuje celkový vstupný prúd, ktorý prijímajú dendrity populácie neurónov. Stavová premenná x popisuje membránový potenciál populácie, čo môžeme interpretovať ako celkovú aktivitu tejto skupiny buniek [14]. Premenná y je takzvaná *premenná obnovy*, v ktorej sú

zhrnuté pomalšie procesy spôsobujúce deaktiváciu spojení medzi neurónmi [6]. Dynamiku celého systému popisuje sústava 2 diferenciálnych rovníc s parametrom,

$$\begin{aligned} \dot{x} &= f(x,y) + I_{TOT} \\ \dot{y} &= \varepsilon \frac{g(x,y)}{\tau(x)}. \end{aligned} \tag{27}$$

Funkcie f a g sú nelineárne a majú tvar

$$f(x,y) = 3x - x^3 + 2 - y \tag{28}$$

$$g(x,y) = \varepsilon \frac{\gamma H_\infty(x) - y}{\tau(x)}, \tag{29}$$

kde $H_\infty(x)$ je hladkou aproximáciou Heavisideovej schodovitej funkcie,

$$H_\infty(x) = \frac{1}{2}\left(tanh\left(100x\right) + 1\right), \tag{30}$$

čiže platí, že $H_\infty(x) \approx 0$ pre $x < 0$ a $H_\infty(x) \approx 1$ pre $x > 0$.

Funkcia $\tau(x)$ je tiež aproximáciou schodovitej funkcie s bodom nespojitosti v $x = 0$,

$$\tau(x) = \tau_1 + (\tau_2 - \tau_1) H_\infty(x), \tag{31}$$

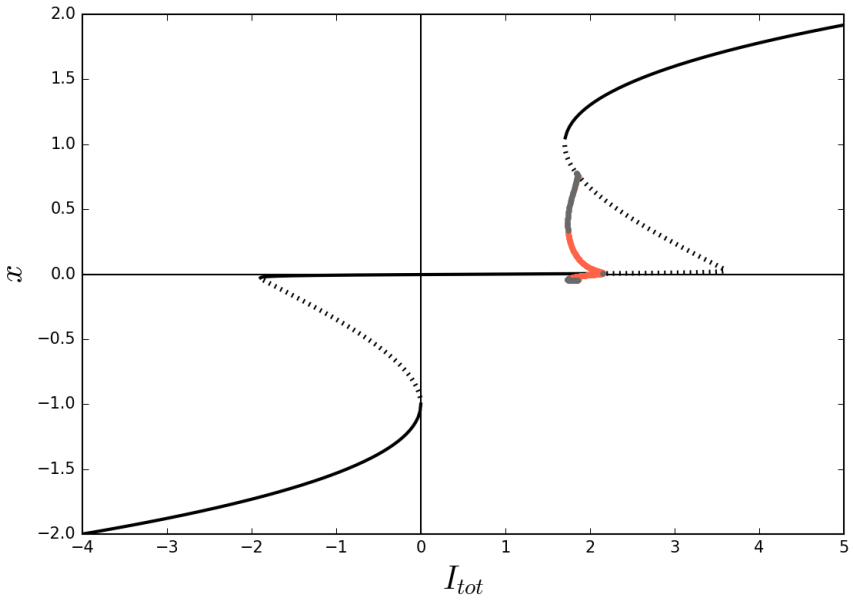
a platí, že $\tau(x) \approx \tau_1$ pre $x < 0$ a $\tau(x) \approx \tau_2$ pre $x > 0$. Hodnoty konštánt ε a γ boli prebraté z [14], kde boli stanovené na $\varepsilon = 3$ a $\gamma = 5,7$. Funkciu H_∞ sme oproti tomuto článku o konštantu posunuli, pretože nespĺňala podmienku na aproximáciu Heavisideovej funkcie. Zvyšné vzťahy ostali rovnaké.

Ak $x > 0$, neuróny považujeme za aktívne, ak $x < 0$, tak za neaktívne.

Pomocou Auto-07p a Pythonu nájdeme stacionárne aj periodické riešenia a ich bifurkácie, vyšetríme ich stabilitu a zostrojíme bifurkačné diagramy. Zdrojový kód je priložený v Prílohe A.

Zistili sme, že model obsahuje dva bifurkačné body typu sedlo-uzol, jednu Hopfovu bifurkáciu a dva ďalšie bifurkačné body. Pri kontinuácii periodických riešení sme zistili, že sa mení stabilita periodických riešení. Závery ponúknuté v článku [14] takéto správanie nepredpokladali.

Z obrázkov vidíme, že pri zápornej hodnote parametra I_{tot}, teda ak má akčný potenciál prijímaný neurónami inhibičný charakter, je populácia neurónov v našom modeli neaktívna.



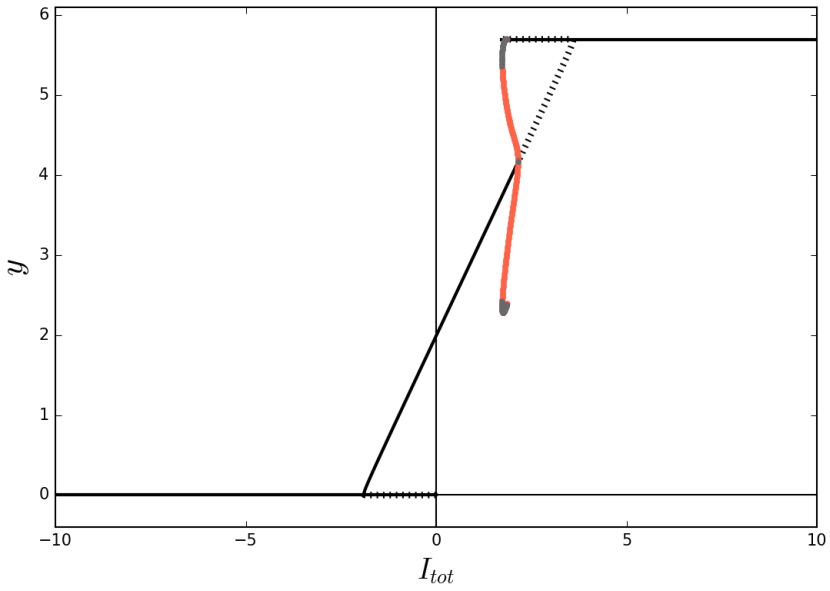
Obr. 21: Bifurkačný diagram modelu (28), priemet do roviny $y =$ konšt.

Nech sa nachádzame v stabilnom stacionárnom stave $x << 0$. Ak postupne zvyšujeme akčný potenciál I_{tot}, tak v momente, keď I_{tot} prvý krát presiahne nulu, stacionárny stav preskočí na stabilný stacionárny stav $x \approx 0$. Ak by sme teraz začali spätne I_{tot} znižovať, tak pri prekročení hranice $I_{tot} = 0$ sa žiaden takýto skok neudeje a systém ostane v stave $x \approx 0$. Znamená to, že aktivita neurónov nezávisí iba od aktuálnej hodnoty akčného potenciálu, ale aj od toho, akým spôsobom táto hodnota nastala. Vlastnosti, že výsledný stav systému je závislý od predchádzajúcich stavov, sa nazýva hysterézia.

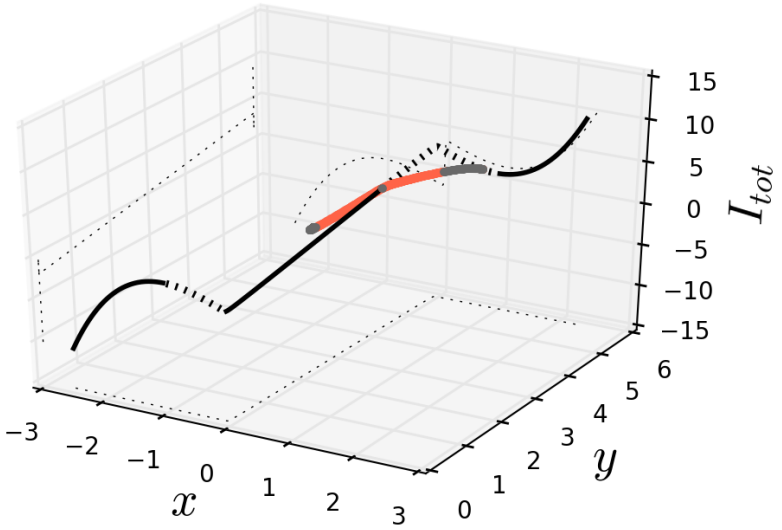
Analogický skok nastáva aj v prípade, že neurónová skupina je stabilne aktívna a akčný potenciál sa postupne znižuje.

Okrem hysterézie môžeme v systéme pozorovať aj Hopfovu bifurkáciu. Nastáva pri vymedzených kladných hodnotách akčného potenciálu. Hodnoty x v tomto limitnom cykle oscilujú na intervale, ktorý obsahuje 0, čo znamená, že pri nezmenenej veľkosti akčného potenciálu sa neuróny opakovane aktivujú a deaktivujú. Striedajú sa tak momenty vysielania výstupného signálu alebo sú nečinnosti.

Ak pri takomto limitnom cykle mierne poklesne hodnota akčného potenciálu, cyklus sa stane nestabilným a ľubovoľne malá odchýlka od jeho trajektórie bude znamenať postupný koniec opakovaného vysielania signálu.



Obr. 22: Bifurkačný diagram modelu (28), priemet do roviny $x =$ konšt.



Obr. 23: Trojrozmerný bifurkačný diagram modelu (28)

Záver

Práca ponúka ucelený pohľad na tému kontinuácie riešení systémov obyčajných diferenciálnych rovníc ako po teoretickej, tak i po praktickej stránke. Oboznamuje čitateľa s teóriou a zároveň prináša konkrétne spôsoby používané na riešenie špecifických druhov úloh a ďalej ich aplikuje na model v oblasti neurológie.

V úvodných dvoch kapitolách sme sa zamerali na teoretické aspekty dynamických systémov. V prvej kapitole sme postupne predstavili teóriu dynamických systémov a ich bifurkácií. Vysvetlili sme pojem bifurkácia a uviedli niekoľko typov bifurkácií, s ktorými sme v ďalších častiach práce skúmali.

Druhá kapitola sa venovala teórii numerickej kontinuácie. Ponúkli sme základný vhľad do metód používaných na numerické hľadanie stacionárnych riešení pri meniacom parametri dynamického systému. Venovali sme sa metódam typu prediktor-korektor a po častiach lineárnym metódam.

Tretia kapitola slúžila na oboznámenie sa so softvérom určeným na riešenie dynamických systémov. Podrobne sme popísali postup potrebný na inštaláciu programu Auto-07p v operačnom systéme Windows, vrátane krokov, ktoré neboli špecifikované v dokumentácii [3] k samotnému programu. Jedným z prínosov práce je tak uľahčenie prístupu k týmto softvérovým riešeniam ďalším záujemcom o riešenie kontinuačných úloh dynamických systémov.

S rovnakým zámerom sme vo štvrtej kapitole vyriešili tri rozličné úlohy a podrobne čitateľa previedli celým procesom ich riešenia. Začali sme vložením zodpovedajúcich rovníc do vstupných súborov a pokračovali špecifikáciou želaného výsledku pomocou konštánt, samotným spustením úlohy, až po následné zobrazenie a interpretáciu získaných výsledkov.

S nadobudnutými znalosťami sme v poslednej kapitole modelovali vybranú úlohu o správaní sa neurónových skupín a získané výsledky interpretovali. Pri modelovaní správania sa jednej neurónovej skupiny s vstupným napätím ako bifurkačným parametrom sme dospeli k výsledkom mierne odlišným od prepokladaných zistení špecifikovaných v článku [14], v ktorom bol model navrhnutý. Zistili sme, že limitný cyklus modelu, ktorý vznikol pri Hopfovej bifurkácii mení stabilitu. Dôvodom môže byť numerická nepresnosť Auto-07p, nezrovnalosť v predpokladoch o predpise funkcie H_∞, na ktorú sme

v tomto článku narazili, prípadne podcenenie bohatosti správania funkcií, ktorými bol model popísaný. Na zistenie konkrétnych príčin a vyvodenie hlbších záverov by bolo potrebné ďalšie skúmanie.

Prínosom práce bolo prehľadné a podrobné zoznámenie sa s teóriou dynamických systémov a softvérovými nástrojmi používanými na riešenie jej úloh. Je prínosná pre čitateľov, ktorí sa s danou témou stretávajú po prvý krát a chceli by pohodlne získať ucelené vedomosti i zručnosti na riešenie konkrétnych problémov a používanie Auto-07p v spojení s Pythonom.

Práca ponúka priestor na predstavenie ďalších typov bifurkácií a s nimi súvisiacich príkladov používania programu Auto-07p. Takéto rozšírenie môže pomôcť uchopiť širšie spektrum úloh z prírodných vied a ekonómie. Taktiež je možné hlbšie sa venovať modelu správania neurónových skupín a urobiť bifurkačnú analýzu pre zjednodušený model spánkového cyklu, ktorý na úlohu riešenú v práci prirodzene naväzuje.

Zoznam použitej literatúry

[1] *Auto*, dostupné na internete (14.5.2017)
http://indy.cs.concordia.ca/auto/

[2] ALLGOWER, E. L., GEORG, K.: *Introduction to Numerical Continuation Methods*, Society for Industrial and Applied Mathematics, Philadelphia, 2003

[3] DOEDEL, E. J., OLDEMAN, B. E.: *AUTO-07P: Continuation and Bifurcation Software for Ordinary Differential Equations*, dostupné na internete (14.5.2017):
http://www.dam.brown.edu/people/sandsted/auto/auto07p.pdf

[4] DOEDEL, E. J.: *Lecture Notes on Numerical Analysis of Nonlinear Equations*, dostupné na internete (14.5.2017)
http://indy.cs.concordia.ca/auto/notes.pdf

[5] DOEDEL, E. J.: *An Introduction to Numerical Continuation Methods with Applications*, dostupné na internete (14.5.2017)
http://users.encs.concordia.ca/~doedel/courses/comp-6361/slides.pdf

[6] GERSTNER, W., KISTLER, W. M.: *Spiking Neuron Models*, Cambridge University Press, Cambridge, 2002

[7] GROSSO, M.: *HOW TO INSTALL AUTO2007 ON WINDOWS*, dostupné na internete (14.5.2017)
http://people.unica.it/massimilianogrosso/auto-on-windows/

[8] HORSFIELD, K., et al.: *Models of the human bronchial tree*, Journal of Applied Physiology Vol. 31, No. 2, 207 - 212, 1971

[9] KANDEL, E. - SCHWARTZ, J. - JESSEL, T.: *Principles of Neural Science*, McGraw-Hill, New York, 2000

[10] KOŽEŠNÍK, J., ŠTĚPÁNEK, M. et al. eds.: *Ilustrovaný encyklopedický slovník, I. diel A - I*, Academia, Praha, 1980

[11] KUZNETSOV, Y. A.: *Elements of Applied Bifurcation Theory, Second Edition*, New York: Springer, 1998

[12] MURRAY, J. D.: *Mathematical Biology: I. An Introduction, Third Edition*, Springer, New York, 2002

[13] PYTHON SOFTWARE FOUNDATION: *Python*, dostupné na internete (14.5.2017)
https://www.python.org/

[14] REMPE, M. J., BEST, J. - TERMAN, D. J.: *A mathematical model of the sleep/wake cycle*, Journal of Mathematical Biology Vol. 60, No. 5, 615 - 644, 2016

[15] STROGATZ, Steven H.: *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering*, Addison-Wesley Pub., Reading, MA, 1994

[16] TYSON, J. J. - CHEN, K. C. - NOVAK, B.: *Sniffers, buzzers, toggles and blinkers: dynamics of regulatory and signaling pathways in the cell*, Current Opinion in Cell Biology Vol. 15 No.2, 221 - 231, 2003

Príloha A: Zdrojový kód

Súbor s funkciami `neur.f90`

```
!---------------------------------------------------------------------
!---------------------------------------------------------------------
      SUBROUTINE FUNC(NDIM,U,ICP,PAR,IJAC,F,DFDU,DFDP)
! Evaluates the algebraic equations or ODE right hand side

! Input arguments :
!      NDIM   :   Dimension of the ODE system
!      U      :   State variables
!      ICP    :   Array indicating the free parameter(s)
!      PAR    :   Equation parameters

! Values to be returned :
!      F      :   ODE right hand side values

! Normally unused Jacobian arguments : IJAC, DFDU, DFDP (see manual)

      IMPLICIT NONE
      INTEGER NDIM, IJAC, ICP(*)
      DOUBLE PRECISION U(NDIM), PAR(*), F(NDIM), DFDU(*), DFDP(*)
      DOUBLE PRECISION x,y, Itot, eps, tau1, tau2, gam, ff, g, tau

    tau(x,tau1,tau2) = tau1 + (tau2 - tau1)*(0.5*TANH(x*100)+0.5)
    ff(x,y) = 3*x - x*x*x + 2 - y
    g(x,y,eps,gam,tau1, tau2) = eps*(gam*(0.5*TANH(x*100)+0.5)-y)/(tau1 +
       ↪  (tau2 - tau1)*(0.5*TANH(x*100)+0.5))

    eps   = 3
    gam   = 5.7
    tau1  = 1
    tau2  = 2

    X   = U(1)
    Y   = U(2)
```

```
    Itot= PAR(1)

    F(1) = ff(X,Y) + Itot
    F(2) = eps*g(X,Y, eps, gam, tau1, tau2)/tau(X, tau1, tau2)

      END SUBROUTINE FUNC
!-----------------------------------------------------------------------
!-----------------------------------------------------------------------
      SUBROUTINE STPNT(NDIM,U,PAR,T)

! Input arguments :
!      NDIM   :   Dimension of the ODE system

! Values to be returned :
!      U      :   A starting solution vector
!      PAR    :   The corresponding equation-parameter values
!      T      :   Not used here

      IMPLICIT NONE
      INTEGER NDIM
      DOUBLE PRECISION U(NDIM), PAR(*), T

! Initialize the equation parameters
   PAR(1) = - 10.125

! Initialize the solution
    U(1) = - 2.5
    U(2) = 0

      END SUBROUTINE STPNT
!-----------------------------------------------------------------------
!-----------------------------------------------------------------------
! The following subroutines are not used here,
! but they must be supplied as dummy routines

      SUBROUTINE BCND
      END SUBROUTINE BCND
```

```
      SUBROUTINE ICND
      END SUBROUTINE ICND

      SUBROUTINE FOPT
      END SUBROUTINE FOPT

      SUBROUTINE PVLS
      END SUBROUTINE PVLS
!---------------------------------------------------------------------
!---------------------------------------------------------------------
```

Súbor konštánt `c.neur`

```
NDIM =   2  # number of variables
unames = {1:x, 2:y}
NPAR =   1  # (maximal) number of parameters
parnames = {1:r}
ICP =  [1]  # bifurcation parameters. Note: 11 is reserved for the period
   ↪  of periodic solutions.
RL0 =   -10.125 # lower bound on the principal continuation parameter
RL1 =   100     # upper bound on the principal continuation parameter
IPS =   1   # 1: stationary solutions of ODEs with detection of Hopf; 2:
   ↪ computation of periodic solutions from a Hopf bifurcation or a
   ↪ periodic orbit from a previous run
ISP =   1   # 0: do not detect special solutions 1: detect branch points
   ↪ and Hopf bifurcation 2: detect all special solutions
ILP =   0   # 0/1: off/on detection of folds
ISW =   1   # 1: normal value; -1: switch the branch
JAC =   0   # 0: derivatives obtained by differencing; 1: derivatives
   ↪ provided by the user
NMX =   500     # maximal number of continuation steps
# CONTINUATION STEP SIZE
DS  =   0.005
DSMIN = 0.001
DSMAX = 0.05
IADS =   1
```

```
THL =  {11: 0.0} # gives zero weight to the 11th parameter (the period)
THU =  {}
# ITERATION ALGORITHMS
EPSL = 1e-07
EPSU = 1e-07
EPSS = 1e-05
ITMX = 8    # number of iterations for the location of special solutions
ITNW = 5    # number of Newton--Chord iteration
NWTN = 3    # the iteration at which Newton is switched to Chord
# PARAMETERS OF LITTLE USE
IRS =   0   # 0: new run; >0: label of a solution (superceded by Python
    ↪ CLUI)
NPR =   500     # write data every NPR-th step (superceded by Python CLUI
    ↪ )
NBC = 0         # number of boundary conditions
NINT= 0     # number of integral conditions
MXBF=  10   # maximal number of bifurcations (for algebraic problems only
    ↪ )
IID =   2   # 2: regular diagnostic output
IPLT = 0    # 0: sets L2 norm as principal solution measure
# DISCRETISATION CONSTANTS
NTST =  20
NCOL =   4
IAD  =   3
```

Python kód

```
cd C:/Users/sash/Documents/python/sleep2
import sys
auto_directory="C:/MinGW/msys/1.0/home/sash/auto/07p"
sys.path.append(auto_directory+'/python')
import auto
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import pandas as pd
```

```
b1 = auto.r('neur', NMX=1000,RL1=10, ISP=2, ILP=1)
auto.save('b1')

content = None
with open('b.b1', 'r') as f:
    content = f.readlines()

content_csv = [[el for el in content[17].split(' ') if len(el) > 0 and el
    ↪  != '\n']]
content_csv[0][0] = 'branch'
column_names = content_csv[0]

for line in content:
    dummy = line.split(' ')
    dummy = [el for el in dummy if len(el) > 0 and el != '\n']
    if dummy[0] == '0':
        continue

    for el_i, el in enumerate(dummy):
        if el_i < 4:
            dummy[el_i] = int(el)
        else:
            dummy[el_i] = float(el)

    if len(dummy) > 1:
        content_csv.append(dummy)

df = pd.DataFrame(content_csv, columns=column_names)
df = df[1:]
df.index=df.index-1

# PERIODICKE RIESENIA pri Hopfovej bifurkacii
per = auto.run(b1('HB1'),ICP=[1,11],IPS=2, IPLT=-1)
auto.save('per')

content = None
with open('b.per', 'r') as f:
    content = f.readlines()
```

```
content_csv = [[el for el in content[18].split('  ') if len(el) > 0 and
    ↪ el != ' \n']]
content_csv[0][0] = 'branch'
column_names = content_csv[0]

for line in content:
    dummy = line.split(' ')
    dummy = [el for el in dummy if len(el) > 0 and el != '\n']
    if dummy[0] == '0':
        continue

    for el_i, el in enumerate(dummy):
        if el_i < 4:
            dummy[el_i] = int(el)
        else:
            dummy[el_i] = float(el)

    if len(dummy) > 1:
        content_csv.append(dummy)

df2 = pd.DataFrame(content_csv, columns=column_names)
df2 = df2[1:]
df2.index=df2.index-1
df2.columns = [c.replace(' ', '') for c in df2.columns]

# PERIODICKE RIESENIA pri Hopfovej bifurkacii 2
per2 = auto.run(b1('HB'),ICP=[1,11],IPS=2, IPLT=-2)
auto.save('per2')

content = None
with open('b.per2', 'r') as f:
    content = f.readlines()

content_csv = [[el for el in content[18].split('  ') if len(el) > 0 and
    ↪ el != ' \n']]
content_csv[0][0] = 'branch'
column_names = content_csv[0]
```

```
for line in content:
    dummy = line.split(' ')
    dummy = [el for el in dummy if len(el) > 0 and el != '\n']
    if dummy[0] == '0':
        continue

    for el_i, el in enumerate(dummy):
        if el_i < 4:
            dummy[el_i] = int(el)
        else:
            dummy[el_i] = float(el)

    if len(dummy) > 1:
        content_csv.append(dummy)

df3 = pd.DataFrame(content_csv, columns=column_names)
df3 = df3[1:]
df3.index=df3.index-1
df3.columns = [c.replace(' ', '') for c in df3.columns]

# 2D grafy
fig, ax = plt.subplots(1, figsize=(8,6))
ax.axhline(y=0, color='k')
ax.axvline(x=0, color='k')
ax.plot(df[(df['PT']>0) & (df['r']<=0)].r,df[(df['PT']>0) & (df['r']<=0)
    ↪ ].x,':k',lw=3)
ax.plot(df[(df['PT']>0) & (df['r']>0)].r,df[(df['PT']>0) & (df['r']>0)].x
    ↪ ,':k',lw=3)
ax.plot(df[(df['PT']<0) & (df['x']<=-0.99)].r,df[(df['PT']<0)& (df['x'
    ↪ ]<=-0.99)].x,'-k',lw=2)
ax.plot(df[(df['PT']<0) & (df['x']>-0.99) & (df['x']<1)].r,df[(df['PT'
    ↪ ]<0) & (df['x']>-0.99) & (df['x']<1)].x,'-k',lw=2)
ax.plot(df[(df['PT']<0) & (df['x']>=0.99)].r,df[(df['PT']<0) & (df['x'
    ↪ ]>=0.99)].x,'-k',lw=2)
ax.plot(df2[df2['PT']>0].r,df2[df2['PT']>0].MAXx,'.',color='tomato')
ax.plot(df2[(df2['PT']<0)].r,df2[(df2['PT']<0)].MAXx,'.', color='dimgray'
    ↪ )
```

```
ax.plot(df2[df2['PT']>0].r,df2[df2['PT']>0].MINx,'.',color='tomato')
ax.plot(df2[(df2['PT']<0)].r,df2[(df2['PT']<0)].MINx,'.', color='dimgray'
    ↪ )
ax.set_xlim(-4,5)
ax.set_ylim(-2,2)
ax.set_xlabel("$I_{tot}$", fontsize=20)
ax.set_ylabel("$x$", fontsize=20)
plt.subplots_adjust(bottom=0.15,right=0.95)
fig.savefig('2-2-0.png', dpi=150)

fig, ax = plt.subplots(1, figsize=(8,6))
ax.axhline(y=0, color='k')
ax.axvline(x=0, color='k')
ax.plot(df[(df['PT']<0)&(df['y']<5)&(df['x']<-0.99)].r,df[(df['PT']<0)&(
    ↪ df['y']<5)&(df['x']<-0.99)].y,'-k',lw=2)
ax.plot(df[(df['PT']<0)&(df['y']<5)&(df['x']>-0.99)].r,df[(df['PT']<0)&(
    ↪ df['y']<5)&(df['x']>-0.99)].y,'-k',lw=2)
ax.plot(df[(df['PT']<0)&(df['y']>5)].r,df[(df['PT']<0)&(df['y']>5)].y,'-k
    ↪ ',lw=2)
ax.plot(df[(df['PT']>0)&(df['y']<3)].r,df[(df['PT']>0)&(df['y']<3)].y,':k
    ↪ ',lw=4)
ax.plot(df[(df['PT']>0)&(df['y']>3)].r,df[(df['PT']>0)&(df['y']>3)].y,':k
    ↪ ',lw=4)
ax.plot(df3[(df3['PT']>0)].r,df3[(df3['PT']>0)].MAXy,'.',color='tomato')
ax.plot(df3[(df3['PT']<0)].r,df3[(df3['PT']<0)].MAXy,'.',color='dimgray')
ax.plot(df3[(df3['PT']>0)].r,df3[(df3['PT']>0)].MINy,'.',color='tomato')
ax.plot(df3[(df3['PT']<0)].r,df3[(df3['PT']<0)].MINy,'.',color='dimgray')
ax.set_xlim(-10,10)
ax.set_ylim(-0.4,6.1)
ax.set_xlabel("$I_{tot}$", fontsize=18)
ax.set_ylabel("$y$", fontsize=20)
plt.subplots_adjust(bottom=0.15,right=0.95)
fig.savefig('2-2-1.png', dpi=150)

# 3D graf
fig = plt.figure()
ax = fig.gca(projection='3d')
```

```
ax.plot(df.x,df.y,-15,':k',lw=0.5)
ax.plot(df.x,[6]*len(df.x),df.r,':k',lw=0.5)
ax.plot([-3]*len(df.x),df.y,df.r,':k',lw=0.5)

ax.plot(df2[(df2['PT']>0)].MINx.values,df3[(df3['PT']>0)].MINy.values,df2
    ↪ [(df2['PT']>0)].r.values,'.r',color='tomato')
ax.plot(df2[(df2['PT']<0)].MINx.values,df3[(df3['PT']<0)].MINy.values,df2
    ↪ [(df2['PT']<0)].r.values,'.',color='dimgray')
ax.plot(df[(df['PT']<0)&(df['x']<-0.5)].x.values,df[(df['PT']<0)&(df['x'
    ↪ ]<-0.5)].y.values,df[(df['PT']<0)&(df['x']<-0.5)].r.values,'-k',lw
    ↪ =2)
ax.plot(df[(df['PT']<0)&(df['x']>-0.5)&(df['x']<0.5)].x.values,df[(df['PT
    ↪ ']<0)&(df['x']>-0.5)&(df['x']<0.5)].y.values,df[(df['PT']<0)&(df['x
    ↪ ']>-0.5)&(df['x']<0.5)].r.values,'-k',lw=2)
ax.plot(df[(df['PT']<0)&(df['x']>0.5)].x.values,df[(df['PT']<0)&(df['x'
    ↪ ]>0.5)].y.values,df[(df['PT']<0)&(df['x']>0.5)].r.values,'-k',lw=2)
ax.plot(df[(df['PT']>0)&(df['y']>3)].x.values,df[(df['PT']>0)&(df['y']>3)
    ↪ ].y.values,df[(df['PT']>0)&(df['y']>3)].r.values,':k',lw=3)
ax.plot(df[(df['PT']>0)&(df['y']<3)].x.values,df[(df['PT']>0)&(df['y']<3)
    ↪ ].y.values,df[(df['PT']>0)&(df['y']<3)].r.values,':k',lw=3)
ax.plot(df2[(df2['PT']>0)].MAXx.values,df3[(df3['PT']>0)].MAXy.values,df2
    ↪ [(df2['PT']>0)].r.values,'.', color='tomato')
ax.plot(df2[(df2['PT']<0)].MAXx.values,df3[(df3['PT']<0)].MAXy.values,df2
    ↪ [(df2['PT']<0)].r.values,'.',color='dimgray')
ax.set_xlabel("$x$", fontsize=20)
ax.set_ylabel("$y$", fontsize=20)
ax.set_zlabel("$I_{tot}$", fontsize=20)
fig1 = plt.gcf()
plt.show()
plt.draw()
fig1.savefig('2-2-2.png', dpi=200)
```