ZÁPADOČESKÁ UNIVERZITA V PLZNI

FAKULTA ELEKTROTECHNICKÁ

KATEDRA ELEKTROENERGETIKY A EKOLOGIE

DIPLOMOVÁ PRÁCE

Software pro zpracování dat z voltametrických měření

Bc. Oldřich Holý **2019**

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta elektrotechnická
Akademický rok: **2018/2019**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Oldřich HOLÝ**
Osobní číslo:	**E17N0031K**
Studijní program:	**N2644 Aplikovaná elektrotechnika**
Studijní obor:	**Aplikovaná elektrotechnika**
Název tématu:	**Software pro zpracování dat z voltametrických měření**
Zadávající katedra:	**Katedra elektroenergetiky a ekologie**

Zásady pro vypracování:

1. Sestavte přehled základních voltametrických elektrochemických metod.
2. Ve vybraném programovacím jazyce vytvořte software pro automatické vyhodnocení naměřeného voltametrického signálu.
3. Proveďte ověření správnosti vyhodnocení voltametrických signálů na reálných datech a dosažené výsledky zhodnoťte.

Rozsah grafických prací: **podle doporučení vedoucího**

Rozsah kvalifikační práce: **40 - 60 stran**

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **BAREK J., OPEKAR F., ŠTULÍK K. Elektroanalytická chemie Praha : Karolinum, 2005.**
2. **ZOSKI, Cynthia G., ed. Handbook of electrochemistry. 1st ed. Boston: Elsevier, 2007. xx, 892 s., [19] s. příl. ISBN 978-0-444-51958-0.**
3. **WANG, Joseph. Analytical electrochemistry [online]. 3rd ed. Hoboken: WILEY-VCH, 2006 [cit. 2018-04-05]. ISBN 0-471-67879-1.**

Vedoucí diplomové práce: **Ing. Robert Vik, Ph.D.**
Katedra technologií a měření

Datum zadání diplomové práce: **5. října 2018**

Termín odevzdání diplomové práce: **30. května 2019**

Prof. Ing. Zdeněk Peroutka, Ph.D.
děkan

L.S.

Doc. Ing. Karel Noháč, Ph.D.
vedoucí katedry

V Plzni dne 5. října 2018

Abstrakt

Práce se zabývá vývojem software pro automatické vyhodnocení dat z voltametrických měření. Téma vzniklo zejména za účelem vytvoření programu, který by nahradil dostupný software výrobců měřicích zařízení, který však z naprosté většiny neumožňuje dávkově zpracovávat a vyhodnocovat naměřená data a chybí přehledné zobrazení všech výsledků měření. To v případě rozsáhlých souborů naměřených dat značně komplikuje a zpomaluje jejich analýzu. Základním principem voltametrických metod je sledování závislosti proudu procházejícího pracovní elektrodou ponořenou v analyzovaném roztoku na potenciálu, přiloženého na tuto elektrodu z vnějšího zdroje. Projevem analytu v analyzovaném roztoku je proudový peak ve výstupní závislosti procházejícího proudu, voltamogramu. Jsou navrženy vhodné metody matematické analýzy k zjišťování polohy, základny a velikosti peaků. Před vlastním vývojem programu jsou definovány jeho základní požadavky a definice struktury formátu vstupních dat. K naprogramování aplikace je vybrán programovací jazyk Python, s využitím grafické knihovny Kivy a knihovny Matplotlib pro vizualizaci dat formou grafů. Je vytvořen hlavní řídicí skript, který tvoří logickou část aplikace. Grafické uživatelské rozhraní je v základu odděleno ve vlastním skriptu. Dále jsou vytvořeny skripty pro konfiguraci parametrů programu z konfiguračního souboru, zpracování dat a vlastní matematickou analýzu. Výstupem práce je funkční program umožňující dávkové zpracování dat s hromadným zobrazením výsledků analýzy a možností exportu dat pro další využití v grafické i numerické podobě. Pro rychlé nastavení parametrů struktury dat jsou výhodně vytvořeny konfigurační profily formátů dat ze softwarů výrobců měřicích zařízení využívaných na Katedře technologií a měření FEL ZČU. Grafická prezentace výsledků je přehledně provedena formou dvou pod sebou umístěných grafů. Horní slouží jako rychlý náhled opakovatelnosti jednotlivých měření souhrnným zobrazením všech zpracovaných souborů dat s možností uživatelské detekce chybných měření a odlehlých hodnot. Spodní graf umožňuje analýzu jednotlivých vzorků se zobrazením základen a výšek zjištěných peaků. Dále jsou zobrazeny souhrnné numerické výsledky všech analyzovaných voltametrických křivek.

Klíčová slova

Voltametrická měření, analýza dat, vývoj software, proudový peak, grafické uživatelské rozhraní, GUI, dávkové zpracování souborů

Abstract

Thesis deals with development of application for automatic voltammetry data analysis. The theme was established mainly because of application development, which can compensate available measuring equipment manufacturer's software. Available software is unavailable to do batch file processing and evaluate measured data at all. Also there is no proper complete data visualisation. It leads to complicated and slow analysis in case of large data sets. Main principle of voltammetry is dependence observation of current passing through the working electrode, which is submerged in analysed solution on potential applied from external source to the electrode. An analyte expression in analysed solution is the current peak in the voltamogram. For the peak location, its base and height the appropriate mathematical methods are brought in. Basic requirements and data structure of input data format are defined before the application development. Python, the programming language is chosen to program the application, with the use of the Kivy graphic library and Matplotlib graphing library for data visualisation. The main control script, which forms the logic part of application, is created. Basic graphic user interface is separated in its own script. Scripts for configuration, data processing and mathematical analysis are further created. The functional application as the output of this work is able to handle the batch file processing with the analysis results bulk view and with both graphic and numeric data export possibilities. The data structure configuration profiles of measuring equipment manufacturer's software, which is used on the Department of Technologies and Measurement by the University of West Bohemia, are created in advantage. Results graphic presentation is clearly displayed with two under each other placed graphs. The upper one is used as a quick summary preview of individual measures repeatability allowing user detection of bad measures or outliers. Lower graph allows one by one sample analysis, showing the base and the height of detected peaks. Summary numeric data results of all analysed voltammetric lines are displayed further.

Keywords

Voltammetric measures, data analysis, software development, current peak, graphic user interface, GUI, batch file processing

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně, s použitím odborné literatury a pramenů uvedených v seznamu, který je součástí této diplomové práce.

Dále prohlašuji, že veškerý software, použitý při řešení této diplomové práce, je legální.

.....

podpis

V Plzni dne 30.5.2019 Bc. Oldřich Holý

Poděkování

Rád bych poděkoval svému vedoucímu práce Ing. Robertu Vikovi, Ph.D. za odborné vedení mé diplomové práce, věcné a užitečné připomínky, velice přínosné konzultace, motivaci k dobrým výsledkům práce a vstřícné a ochotné jednání.

Dále děkuji Doc. Ing. Františku Vávrovi, CSc. za prohloubení mých znalostí v oblasti matematické a statistické analýzy a za trpělivost při objasňování této problematiky mé osobě.

Závěrem děkuji všem svým přátelům, kteří mě při práci podporovali a motivovali, zejména Ing. Jaroslavu Rybákovi, Bc. Janě Široké a Mgr. Evě Rybákové.

Obsah

Seznam symbolů a zkratek

ZKRATKA	ANGLICKY	ČESKÝ VÝZNAM
AC	alternating current	střídavý proud
ACV	alternating current voltammetry	metody střídavé voltametrie
ASV	anodic stripping voltammetry	anodická rozpouštěcí voltametrie
CSV	cathodic stripping voltammetry	katodická rozpouštěcí voltametrie
CSV	comma separated values	datový soubor oddělený čárkami
DAVM	Data Analysis Of Voltammetric Measures	Datová analýza voltametrických měření
DC	direct current	stejnosměrný proud
DPV	differential pulse voltammetry	diferenční pulsní voltametrie
FEL ZČU	Faculty of Electrical Engineering by University of West Bohemia	Fakulta elektrotechnická Západočeské univerzity
GUI	graphic user interface	grafické uživatelské rozhraní
Hz	hertz	jednotka frekvence
IUPAC	International Union for Pure and Applied Chemistry	Mezinárodní unie pro čistou a užitou chemii
NPV	normal pulse voltammetry	normální pulsní voltametrie
OS	operating system	operační systém
PNG	portable network graphics	grafický formát souboru
PSA	potentiometric stripping voltammetry	potenciometrická rozpouštěcí analýza
SWV	square wave voltammetry	SW voltametrie
C	[F]	Elektrická kapacita
I	[A]	elektrický proud
k	[-]	konstanta počtu okolních bodů (+/-)
R	[Ω]	Elektrický činný odpor
U, E, φ	[V]	elektrické napětí, potenciál

Úvod

Předkládaná práce se zabývá vývojem softwaru pro automatizaci vyhodnocení dat z elektrochemických měření. Téměř veškerý software dodávaný výrobci měřicích přístrojů pro elektrochemii je navržen tak, že umožňuje v jednu chvíli zpracovávat pouze jednu naměřenou voltametrickou křivku a až následně přejít ke zpracování další křivky. Ani jeden z programů dostupných na Katedře technologií a měření FEL ZČU nemá funkci hromadného zpracování naměřených souborů dat. Není zde dosud možnost zpracovat souborovou dávku s daty tak, aby software následně hromadně a přehledně zobrazil výsledky měření, tedy výšky a polohy jednotlivých vrcholů na jednotlivých křivkách. To v případě rozsáhlých souborů naměřených voltametrických dat značně komplikuje a zpomaluje jejich analýzu. Tyto rozsáhlé soubory dat vznikají např. při optimalizaci rozměrů a tvarů elektrod na senzorových elementech nebo v případě měření velkého počtu tištěných senzorů, kdy je účelem posouzení opakovatelnosti fyzikálně-chemických vlastností natištěných souborových elementů v závislosti na jejich poloze na tiskovém substrátu. V takovýchto případech je třeba opakovat za shodných podmínek velké množství měření. Jejich analýza je pak spíše rutinní prací, kterou lze právě rychlým náhledem na hromadně zpracovaná a vyhodnocená data výhodně optimalizovat.

Motivací a současně cílem práce je navržení takového softwaru, který umožňuje automatické dávkové zpracování naměřených dat i pro větší množství datových souborů současně. Je však nutné zajistit, aby výsledný produkt uměl data nejen zpracovat, ale aby zároveň poskytoval dostatečnou výpovědi-schopnost výsledků datové analýzy. Je tedy nutné navrhnout vhodné metody matematické analýzy pro automatické zjišťování polohy a výšky vrcholů z naměřených dat. Dále je žádoucí výsledky zpracovaných dat vhodně prezentovat uživateli tak, aby byl schopen snadného porovnání dílčích voltametrických křivek jednotlivě mezi sebou a v rámci celé dávky zpracovaných datových souborů. Případně také umožnit uživateli z výsledků vyloučit chybně naměřená nebo zkreslená data.

Pro vývoj programu splňujícího uvedené požadavky je zapotřebí zvolit vhodné programové nástroje, které nabízejí dostatečnou funkcionalitu a zároveň takové dílčí vývojové prvky, které jsou vzájemně kompatibilní. Předpokladem pro snadnou obsluhu a vhodnou vizuální prezentaci počítačové aplikace je implementace programu v grafickém uživatelském prostředí. Z podstaty, že se jedná o nezávislý software, je výhodou zvolit pokud možno univerzální způsob čtení souborů naměřených dat napříč výstupy z měřicích systémů různých výrobců a umožnit snadné nastavení vstupních parametrů pro jejich

efektivní a rychlé zpracování. Výsledkem zpracování naměřených dat jsou potom nově získané informace. Kromě jejich prezentace v grafickém prostředí aplikace by mělo být umožněno uživateli data uchovat pro zpětnou kontrolu nebo další zpracování. Není vhodné, aby při požadavku na zpětné získání informací o výsledcích analýzy dat bylo nutné provádět jejich zpracování v programu znovu nebo ručně vyhodnocená data zapisovat.

Řešení problematiky je v této práci rozděleno do čtyř částí. První část je věnována teorii elektrochemie. Jsou zde vysvětleny základní pojmy, některé zákonitosti a konvence. Následně je uveden základní přehled elektroanalytických metod.

Ve druhé části jsou blíže specifikovány voltametrické a ampérometrické metody elektrochemických měření, definice aplikovaných a závislých fyzikálních veličin a výstupní křivky naměřených dat. Dále je nastíněno základní uspořádání měřicího systému a vliv použitých prvků. Následuje podrobnější popis jednotlivých voltametrických a amperometrických metod.

Třetí část je již prakticky zaměřená na zkoumání vhodných matematicko-analytických metod pro rozbor naměřených dat k určení požadovaných výsledků v rámci jejich zpracování. Z předchozích teoretických poznatků o voltametrických měřeních jsou definována vstupní data a požadované výsledky analýzy. Následuje podrobný popis zamýšlených metod včetně návrhu postupů k jejich implementaci do programu.

Poslední a nejrozsáhlejší část práce se věnuje vlastnímu vývoji softwaru pro zpracování dat z voltametrických měření. V první řadě jsou podrobně popsány základní požadavky na funkce programu, od definice vstupních dat po požadavky na vizualizaci a formát výsledků analýzy. Poté je pozornost věnována výběru vhodného programovacího jazyka a dalších programových nástrojů dostačujících k provedení zamýšlené aplikace. V několika dalších podkapitolách je po uvedení základní struktury navrženého programu detailní rozbor jednotlivých částí aplikace, funkcionality, implementace matematické analýzy dat a způsob zobrazení výsledků provedeného zpracování dat, včetně exportu výstupů aplikace k dalšímu použití.

1 Elektroanalytická chemie a elektrochemie

Elektroanalytická chemie je částí analytické chemie, využívající elektrochemických procesů pro identifikaci a definici látek. Elektrochemie je fyzikálně chemická vědní disciplína zabývající se chemickými procesy za účasti elektricky nabitých částic. Základními sledovanými ději jsou transportní procesy, tedy přenos nabitých částic na rozhraní roztok-elektroda a dále oxidačně-redukční děje, kdy dochází k elektrostatickým interakcím částic [1] [2].

Základním předpokladem využitelnosti elektrochemických dějů pro analýzu je možnost takové děje experimentálně sledovat za předem definovaných a opakovatelných podmínek. Měřený signál musí být potom přiřaditelný určité složce (či alespoň sumě určitých složek) systému a musí existovat vzájemný vztah mezi signálem a složkami tohoto systému [1].

Získávání informací o měřeném objektu je založeno na principu jeho odezvy při definovaným způsobem dodaném množství energie a měření vhodných vlastností. V elektrochemii se jedná zejména o měření elektrických veličin [1].

1.1 Základní pojmy, zákonitosti a konvence

Základní dvoufázový systém je tvořen elektrodou ponořenou v elektrolytu. Rozdíl potenciálů mezi těmito dvěma fázemi se nazývá elektrodový potenciál. [1].

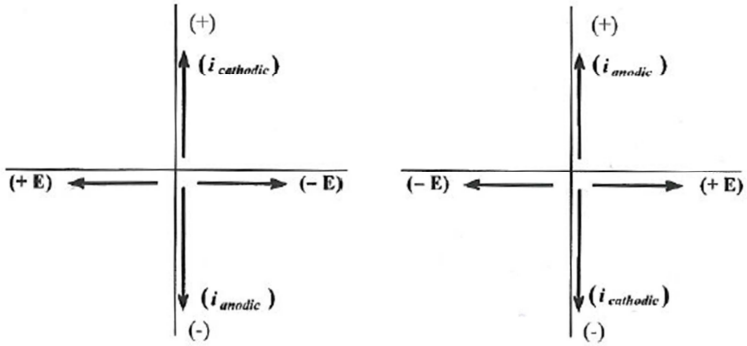
Individuální elektrodový potenciál je neměřitelný. Z tohoto důvodu se zapojují dvě elektrody do jednoho celistvého systému, který se nazývá galvanický (elektrochemický) článek (*galvanic cell*). Jedna elektroda se potom nazývá poločlánek (*half-cell*). Napětí článku je již měřitelné a je dáno rozdílem potenciálů dvou poločlánků – elektrod [1].

Aby bylo možné definovat základní vlastnosti měřicího systému, je nutné zavést několik konvencí. Zápis článku, tedy přechody mezi jednotlivými fázemi, je možné zapsat zleva doprava, ale i naopak. Toto má však následně vliv na značení kladného a záporného potenciálu. Shodným způsobem je nutné zavést konvenci vzhledem k označení proudů tekoucích elektrodami. V chemii se tyto konvence řídí dle mezinárodní profesní organizace. Pro Evropu je to Mezinárodní unie pro čistou a užitou chemii (*International Union for Pure and Applied Chemistry - IUPAC*). Dle IUPAC je hodnota napětí galvanického článku dána odečtením změny potenciálu poločlánku levého od změny potenciálu poločlánku pravého, tedy

$$U = \Delta\varphi\ (pravá) - \Delta\varphi\ (levá) \qquad (1.1)$$

Pro určení proudů platí následující. Elektroda, na které dochází k oxidaci, se nazývá anoda. Proud vůči této elektrodě se tedy nazývá anodický a má kladné znaménko. Elektroda, na které dochází k redukci, se nazývá katoda a proud vůči této elektrodě se nazývá katodický a má záporné znaménko [1] [2].

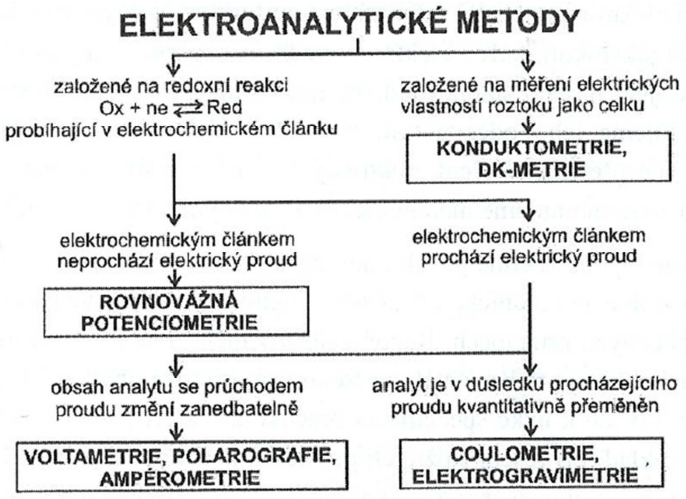
Pro znázornění výsledků měření se použije graf, tzv. voltamogram. Na následujícím obrázku (Obrázek 1) je znázorněno konvenční uspořádání grafu dle IUPAC, spolu s porovnáním značení dle amerických zvyklostí [2].



Obrázek 1- Uspořádání grafu dle amerických zvyklostí (vlevo) a dle IUPAC (vpravo) [2]

1.2 Základní přehled elektroanalytických metod

Elektroanalytické metody lze nejprve rozdělit do dvou skupin. První a nejpočetnější jsou metody elektrochemické, založené na oxidačně-redukčních reakcích. Druhá skupina jsou metody elektrometrické, při nichž je měřena určitá elektrická vlastnost roztoku jako celku. Základní rozdělení elektroanalytických metod je znázorněno na následujícím obrázku (Obrázek 2) [1].



Obrázek 2 - Základní rozdělení elektroanalytických metod [1]

Podrobnější klasifikací metod je dělení dle veličiny, která je kontrolována a dle veličiny, která je měřena, s vyjádřením funkční závislosti měřené veličiny. Podrobný přehled je uveden na následujícím obrázku (Obrázek 3). Tato práce se dále zaměřuje podrobněji na metody voltametrické a amperometrické.

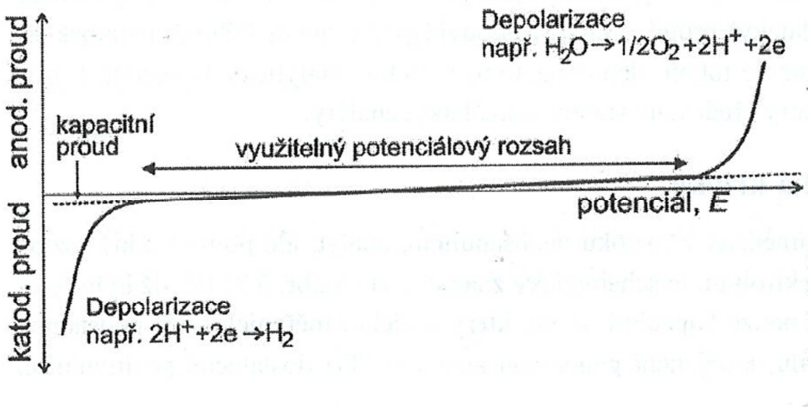
Kontrolovaná veličina	Měřená funkce	Název metody
1. Metody založené na elektrochemické reakci		
1.1 Nulový elektrolytický proud		
	$E = f(a)$	rovnovážná potenciometrie a potenciometrické titrace
1.2 Elektrolytický proud má konečnou hodnotu		
1.2.1 Stacionární metody		
E	$I = f(E)$	voltametrie (a polarografie) za konstantního potenciálu
1.2.1.1 Ampérometrické titrace		
E	$I = f(V)$	a) ampérometrické titrace
U	$I = f(V)$	b) biampérometrické titrace
I	$E = f(V)$	c) potenciometrické titrace za konstantního proudu
I	$U = f(V)$	c) bipotenciometrické titrace
1.2.1.1 Metody založené na hmotnostní bilanci elektrochemické reakce		
E	$Q = \int_0^t I \, dt$	coulometrie za konstantního potenciálu
E	$Q = f(c)$	polarografická coulometrie
I	$Q = It$	coulometrie za konstantního proudu
E nebo I	m	elektrogravimetrie
1.2.1 Nestacionární metody		
I	$E = f(t)$	chronopotenciometrie
E	$I = f(c)$	chronoampérometrie (polarografie)
$E + E(t)$	$I(t) = f(E)$	polarografie a voltametrie se střídavou složkou napětí (a.c, square-wave, pulsní, diferenčně pulsní)
$E = E_i - vt$	$I = f(E)$	voltametrie s lineárně se měnícím napětím, resp. s napětím, které je periodickou funkcí času (single-sweep, multi-sweep)
2. Metody založené na elektrických vlastnostech roztoku		
$I(t)$	$\kappa = f(c_i)$	nízkofrekvenční konduktometrie
$I(t)$	$\kappa = f(V)$	konduktometrické titrace
W_v	$Z = f(\kappa + \varepsilon)$	vysokofrekvenční měření impedance

a – aktivita analytu, c_i – koncentrace nabitých částic, E_i – počáteční potenciál, $E(t)$ – potenciál jako periodická funkce času, $I(t)$ – proud jako funkce času, V – objem přidaného titračního činidla, m – hmotnost, v – rychlost změny potenciálu (V s^{-1}), W_v – energie elektromagnetického záření v oblasti radiofrekvencí, Z – impedance, κ – nízkofrekvenční elektrická vodivost, ε – relativní permitivita

Obrázek 3 - Podrobné rozdělení metod dle kontrolované veličiny a měřené funkce [1]

2 Voltametrické a amperometrické metody

Základním principem voltametrických metod je sledování závislosti proudu procházejícího pracovní elektrodou ponořenou v analyzovaném roztoku na potenciálu, přiloženého na tuto elektrodu z vnějšího zdroje. Tuto závislost je následně možné vizualizovat do voltamogramu jako polarizační křivku (viz Obrázek 4).

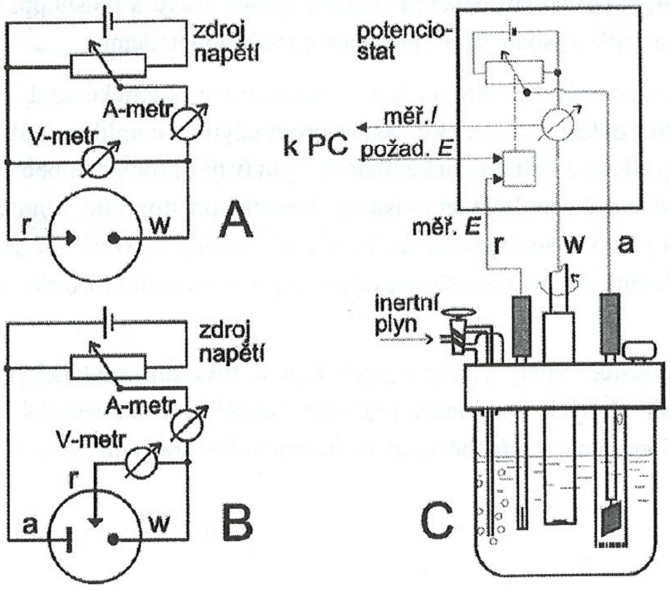


Obrázek 4 - Polarizační křivka [1]

Proud procházející v přítomnosti analyzované látky (analytu) při určitém potenciálu je potom analytickým signálem. Projev na proudové změny analytu v roztoku na polarizační křivce se dá nazvat jako voltametrická křivka [1]. O amperometrických metodách je popsáno více v kapitole 2.3.

Uspořádání měřicího systému

Pro voltametrická měření se využívají zapojení elektrochemického článku ve dvouelektrodovém nebo tříelektrodovém uspořádání (viz Obrázek 5). Dvouelektrodové uspořádání obsahuje pracovní elektrodu ponořenou do analyzovaného roztoku a referenční (srovnávací) elektrodu spojenou s analyzovaným roztokem solným můstkem. Referenční elektroda se musí chovat jako ideální nepolarizovaná elektroda. V tříelektrodovém uspořádání se přidává ještě třetí elektroda, pomocná. Pomocná elektroda se zapojuje proto, že potenciál pracovní elektrody není přesně znám z důvodu částečné ztráty napětí na odporu analyzovaného roztoku. Skutečný potenciál pracovní elektrody se potom měří mezi pracovní a referenční elektrodou za bezproudového stavu. Proud prochází pouze mezi pracovní a pomocnou elektrodou a elektrochemické reakce probíhající na referenční elektrodě se nesledují. V praxi se tedy používá zejména tříelektrodové uspořádání. V případě systému s velkým odporem se využívají ještě speciální mikroelektrody [1] [2].



Obrázek 5 - Základní schéma zapojení v dvouelektrodovém (A) a tříelektrodovém (B) zapojení a příklad realizace voltametrické nádobky (C) v tříelektrodovém zapojení (a – pomocná elektroda, r – referenční elektroda, w – pracovní elektroda) [1]

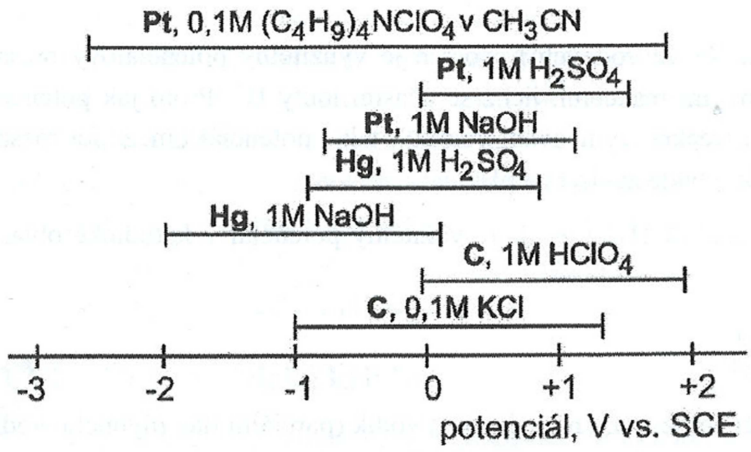
Zdrojem napětí v tříelektrodovém zapojení je elektronický potenciostat, který udržuje potenciál pracovní elektrody na požadované hodnotě. Potenciostat neustále porovnává požadovaný potenciál pracovní elektrody s aktuálním změřeným potenciálem a případný rozdíl automaticky vyrovnává změnou napětí na pomocné elektrodě. Rozdíl potenciálů může být způsoben např. úbytkem napětí na odporu roztoku [1].

Vliv použitých prvků měřicího systému

Typ použité pracovní elektrody, rozpouštědla a základního elektrolytu má vliv na rozsah potenciálů, který lze analyticky využít. Využitelný rozsah potenciálů se potom nazývá potenciálové okno (*potential window*). Potenciálové okno ovlivňuje průběh polarizační křivky a naopak je změřenou polarizační křivkou toto okno znázorněno. Na obrázku (Obrázek 4) je uveden příklad polarizační křivky (změřená při DC voltametrii, viz dále). Křivka je změřena v roztoku neobsahujícím analyt, pouze základní elektrolyt. Exponenciální změny proudu na této křivce vyjadřují oblasti, kdy již dochází k depolarizaci složky základního elektrolytu nebo k oxidaci materiálu elektrody [1].

Na dalším obrázku (Obrázek 6) je pro srovnání znázorněno několik potenciálových oken pro různé druhy elektrod v různých typech roztoků.

Výběr vhodných prvků měřicího systému může tedy mít podstatný vliv na průběh a výsledky dalších měření či analýz, a je nutné volit tyto prvky vhodně vzhledem k určovanému analytu v roztoku.



Obrázek 6 – Přibližné potenciálové rozsahy na různých typech elektrod [1]

2.1 Stacionární (DC) voltametrie

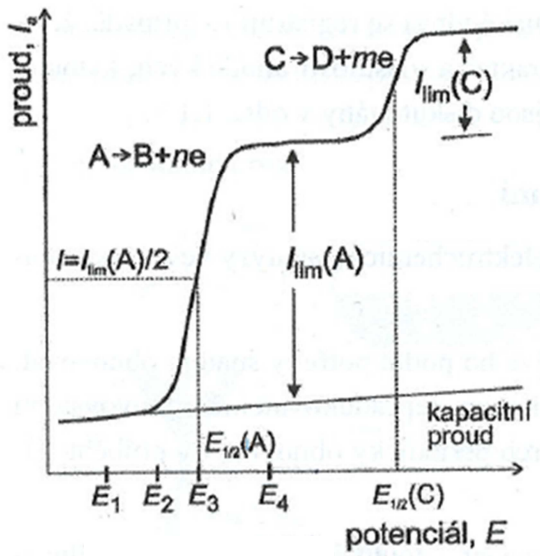
Průběh měření při stacionární (DC) voltametrii je založen na lineární změně přikládaného potenciálu na pracovní elektrodu. Metodu lze považovat za stacionární při konstantním potenciálu, neboť rychlost změny potenciálu je dostatečně malá vzhledem k ustanovování rovnováhy u elektrody. Závislost potenciálu, vkládaného na pracovní elektrodu, na čase se pro účely této metody nazývá potenciálový program. Závislost měřeného proudu procházejícího pracovní elektrodou se potom nazývá proudová odezva. Vzhledem k závislosti přikládaného potenciálu na čase je potenciálová osa v podstatě současně i osou časovou [1].

Jak již bylo řečeno dříve, vyjádřením výsledků měření za přítomnosti analytu v roztoku je voltametrická křivka. Polarizační křivka je změřena v roztoku základního elektrolytu bez výskytu analytu v tomto roztoku. V rozsahu potenciálového okna prochází pracovní elektrodou pouze kapacitní (nabíjecí) proud, který je způsoben elektrodou chovající se jako kondenzátor svou elektrickou dvojvrstvou. Tento kapacitní proud tvoří základní složku „šumu“. Jedná se tedy o signál, který není generován analytem [1].

V případě, že je v roztoku přítomen analyt, který se při určitém potenciálu na pracovní elektrodě oxiduje či redukuje, a dochází tak k depolarizaci elektrody, začne pracovní elektrodou procházet elektrolytický (faradaický) proud. Velikost odpovídajícího katodického či anodického proudu je potom mírou koncentrace analytu v roztoku. Při potenciálech mimo potenciálové okno rovněž dochází k depolarizaci elektrody, jak již bylo popsáno dříve [1].

Pokud tedy vezmeme v úvahu polarizační křivku s obsaženým analytem v roztoku, jeho přítomnost se na polarizační křivce projeví voltametrickou vlnou. Možný průběh

voltametrické křivky je znázorněn na obrázku (Obrázek 7) [1].



Obrázek 7 - Voltametrická vlna registrovaná při DC voltamerii [1]

2.2 Nestacionární voltametrické metody

Následující popisované voltametrické metody se řadí do skupiny nestacionárních metod voltametrie. U těchto metod je měřená veličina funkcí času. Při měření se jedna veličina kontroluje a sleduje se závislost druhé veličiny na čase. V případě voltametrických metod je kontrolovanou veličinou potenciál a měřenou veličinou proud [1].

Oproti stacionárním metodám se využívá přikládání pulzního potenciálu nebo střídavého potenciálu o frekvenci v řádu desítek Hz. Potenciálová změna je tedy oproti stacionárním metodám skoková. Tomu odpovídá větší proud – analytický signál. S růstem změny potenciálu se voltametrická vlna mění v proudovou špičku (dále jako „peak“). Proud po dosažení maxima klesá z důvodu, že difúze nestačí pokrýt úbytek látky spotřebované elektrodovou reakcí [1].

2.2.1 Cyklická voltametrie

Přímé analytické využití metody cyklické voltametrie je malé, využívá se spíše při zkoumání elektrodových reakcí. Principem této metody je přikládání potenciálu trojúhelníkovitého průběhu na pracovní elektrodu s rychlostí změny (polarizace) dE/dt. Výstupem této metody je cyklická voltametrická křivka, kdy v jednom směru dochází k oxidaci a v opačném směru k redukci. Následně se vyhodnocují rozdíly potenciálů anodického a katodického peaku a podíly těchto peaků [1].

2.2.2 Metody voltametrie se střídavou složkou

Metody střídavé voltametrie (*alternating current voltammetry - ACV*) se v současné době v elektroanalytické chemii příliš nevyužívají. Jejich testování probíhalo za účelem zvýšení citlivosti a zlepšení poměru mezi signálem a šumem. Potenciál vkládaný na pracovní elektrodu se mění lineárně s časem a moduluje se střídavým napětím sinusového průběhu malé amplitudy a nízké frekvence. Je také ovlivněn určitým stejnosměrným potenciálem vzhledem k referenční elektrodě. Měřena je závislost střídavého proudu procházejícího pracovní elektrodou na jejím potenciálu. Výhodou AC voltametrie je relativně snadný výpočet odporu roztoku a kapacity elektrodové dvojvrstvy [1] [2].

Další metodou střídavé voltametrie je SW voltametrie (*square wave voltammetry - SWV*). Při této metodě se na pracovní elektrodu vkládá lineárně se měnící (s časem) potenciál modulovaný střídavým napětím pravoúhlého tvaru. V tomto případě se proud měří pouze po dobu asi 2 milisekund a to ke konci každého vloženého pravoúhlého pulsu – jedná se tedy o vzorkování průběhu proudu. Pro analýzu se porovnává závislost dvou po sobě změřených vzorků a výstupem závislosti je peak [1].

2.2.3 Pulsní metody voltametrie

Podobně, jako bylo popsáno u SW voltametrie, tak i v případě pulsních metod se proud měří vzorkováním po určitou dobu, která je přesně definována. Ke vzorkování dochází, pokud je potenciál konstantní již po určitou dobu, typicky 40 milisekund. To vede k výsledným voltamogramům ve tvaru schodovitých průběhů, které jsou různě „jemné“ či „hrubé“ a to v závislosti na rychlosti vzorkování a změny přikládaného potenciálu [1] [2].

Jak již bylo zmíněno v případě AC metod, také u pulsních metod je díky vývoji úspěšně zlepšován poměr signálu a šumu. Šumem je tedy kapacitní proud elektrody, který je potlačován na základě jeho exponenciálního poklesu po vložení pulsu podle vzorce

$$I_c = \frac{\Delta E}{R} e^{\frac{-t}{RC}}. \qquad (2.1)$$

Po době asi pětinásobku časové konstanty představované součinem RC, dochází k tomu, že kapacitní proud klesá k hodnotě téměř zanedbatelné. Signál, tedy faradaický proud, klesá s druhou odmocninou času, tzn. podstatně pomaleji než šum. Dále dochází k eliminaci nabíjecího proudu vhodnou synchronizací vkládání pulsů a koncentrační gradient u elektrody je po vložení pulsu podstatně větší než před jeho vložením. Tedy i faradaický proud je větší. Kombinací těchto skutečností je docíleno lepšího poměru signálu a šumu [1] [2].

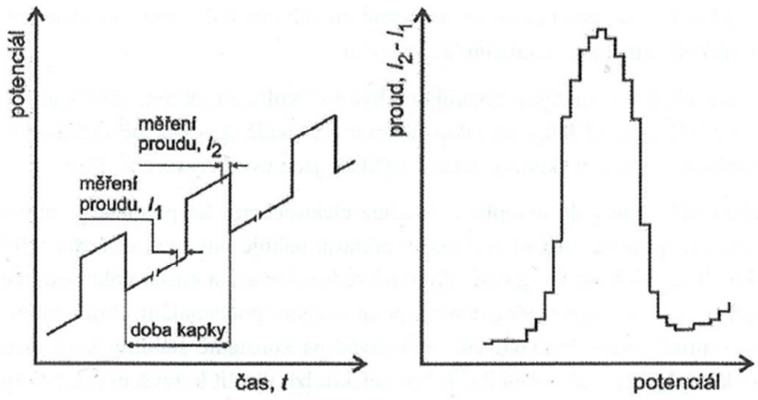
Normální pulsní voltametrie (NPV)

V normální pulsní voltametrii se na elektrodu pulsy vkládají v šířce desítek milisekund o postupně rostoucí amplitudě. Proud se vzorkuje na konci vloženého pulsu po dobu jednotek až desítek milisekund [1].

Diferenční pulsní voltametrie (DPV)

Diferenční pulsní voltametrie je v současnosti z pulsních metod nejvíce využívána a má pro praktickou elektroanalytickou chemii největší význam [1].

Oproti normální pulsní metodě se přikládaný proud lineárně mění s časem. Jsou vkládány napěťové pulsy v trvání řádově desítek milisekund. Vzorkování proudu probíhá dvakrát, těsně před vložením napěťového pulsu a následně na jeho konci (viz Obrázek 8). Uchovává se hodnota rozdílu těchto vzorků. Vyhodnocení analytu v roztoku je na voltamogramu opět ve tvaru peaku. Výška peaku roste s rostoucí amplitudou vkládaného pulsu, ale zároveň dochází k jeho rozšiřování. Ve vyhodnocení to znamená horší selektivitu stanovení analytu a je nutné volit vhodný poměr citlivosti a selektivity měření [1].



Obrázek 8 - Potenciálový program a proudová odezva při diferenční pulsní voltametrii [1]

Rozpouštěcí metody voltametrie

Následující popsané metody využívají další způsob zlepšení poměru signálu a šumu. Principem je zvýšení koncentrace analytu v roztoku na pracovní elektrodě. Následná zvýšená koncentrace analytu na pracovní elektrodě oproti koncentraci v roztoku vede při rozpouštění z povrchu elektrody zpět do roztoku k většímu měřenému signálu – proudu [1].

Anodická rozpouštěcí voltametrie (ASV) funguje na principu předběžné elektrolýzy za podmínky řízeného potenciálu, kdy se kation kovu katodicky vylučuje ve formě kovu, který tvoří buď amalgam (se rtuťovou elektrodou) nebo film (na povrchu elektrody). Po určité době se vyloučený kov anodicky rozpouští z elektrody zpět do roztoku, tedy oxiduje [1] [2].

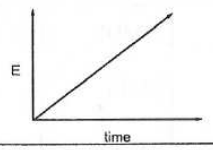
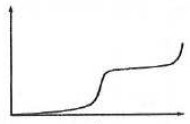
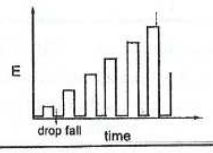
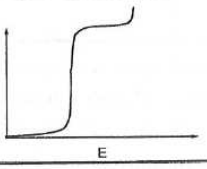
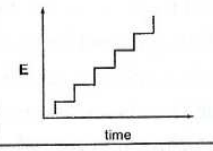
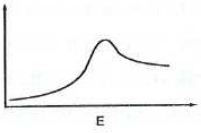
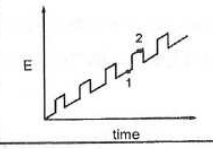
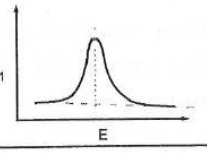
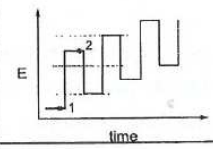
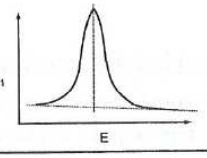
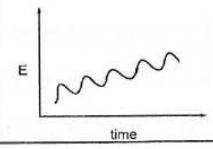
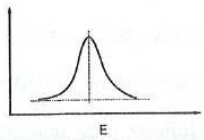
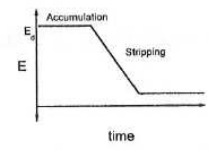
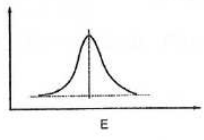
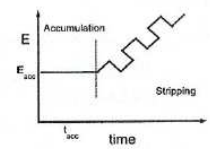
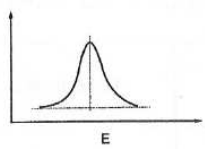
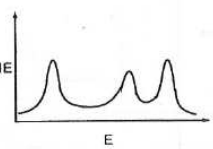
Oproti tomu katodická rozpouštěcí voltametrie (CSV) funguje ve dvou variantách. V první variantě je iont kovu předběžnou elektrolýzou převeden do vyššího oxidačního stavu (se složkou obsaženou v roztoku). Tato sloučenina se zachytí na elektrodě a následně je depozit katodicky rozpuštěn zpět do roztoku. Ve druhé variantě se zvýší koncentrace analytu na pracovní elektrodě, a to při potenciálu rozpouštění materiálu elektrody. Vznikají tak ionty, které reagují s analytem za vzniku nerozpustné sloučeniny. Ta tvoří na elektrodě film a následně je opět katodicky rozpouštěna zpět do roztoku. CSV může být použita pro analýzu organických a anorganických sloučenin [1] [2].

Metoda adsorbční rozpouštěcí voltametrie, jak již z názvu vyplývá, funguje na základě adsorpce při určitém potenciálu, kdy se zvyšuje koncentrace analytu na povrchu elektrody. Z rozpouštěcích metod má tato širší rozsah druhů látek, které je možné rozpouštěcími metodami analyzovat. Elektrochemicky aktivní látky lze stanovit z výšky peaku po jejich redukci či oxidaci. Elektrochemicky neaktivní látky se mohou po dosažení určitého potenciálu desorbovat, potom lze analyticky využít peak tenzametrický. Jedná se o kapacitní proud procházející elektrodou, při změně její kapacity (elektrické dvojvrstvy) při desorbci [1] [2].

Poslední zmiňovanou metodou je galvanostatická a potenciometrická rozpouštěcí analýza (PSA). V obou případech se při rozpouštění sleduje závislost potenciálu pracovní elektrody na čase. V závislostní křivce potenciálu a času se rozpouštění látky projeví přechodovým časem, po jehož dobu se látka rozpouští. Tento přechodový čas je signálem, pomocí něhož lze na základě kalibrace látku obsaženou v analytu stanovit [1].

2.2.4 Souhrnný přehled voltametrických metod a možné průběhy

Na následujícím obrázku (Obrázek 9) je souhrnný přehled základních voltametrických metod, které jsou podrobněji popsány v předchozích kapitolách. Na obrázku je možné zejména sledovat možné průběhy přikládaného potenciálu a proudové odezvy na voltamogramu.

Technika	Potenciálový program	Proudová odezva	Pracovní elektroda	Mez stanovitelnosti (mol l^{-1})
TAST			DME	~ 10^{-6}
NPP (NPV)			DME (HMDE) TE, PE	~ 10^{-7} (~10^{-7})
SCV			HMDE TE, PE	~ 10^{-7}
DPP (DPV)			DME (HMDE) TE, PE	~ 10^{-7} ~ 10^{-8}
SWP (SWV)			DME (HMDE) TE, PE	~ 10^{-8} ~ 10^{-8}
ACP (ACV)			DME (HMDE) TE, PE	~ 10^{-7} (~10^{-8})
ASV [a,c,d] (CSV) [a,b,c,d]			HMDE, MFE.,TE	~10^{-10} (~10^{-9})
AdSV [b,d]			HMDE, MFE, TE, PE	~10^{-11} ~10^{-12}
PSA			HMDE, MFE, TE, PE	~ 10^{-12}

Vysvětlivky: a – elektrolytická prekoncentrace, b – adsorpční prekoncentrace, c – DC rozpouštěcí krok, d – DP rozpouštěcí krok, 1 – vzorkování proudu před vloženým pulsem, 2 – vzorkování proudu na konci vloženého pulsu; TE – tuhé elektrody, PE – pastové elektrody; SCV – tzv. "staircase voltammetry".

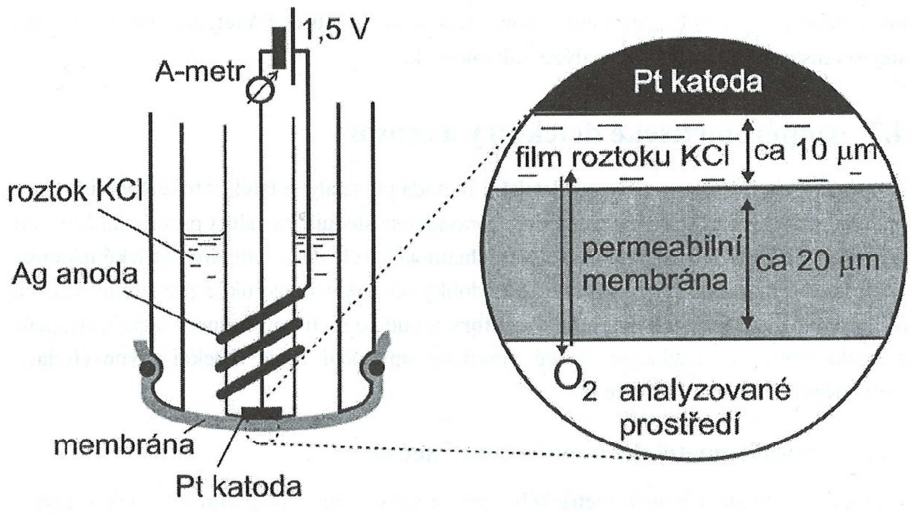
Obrázek 9 - Přehled základních voltametrických metod [1]

2.3 Amperometrické metody

Metody amperometrie jsou založené na určení analytu při konstantním potenciálu, měřením proudu procházejícího pracovní elektrodou. Oproti voltametrickým metodám se tedy potenciál přikládaný na elektrodu s časem nemění, ale jeho hodnota se volí tak, aby elektrodou procházel limitní (neboli difúzní) proud analytu [1].

Měřicí systém používaný pro amperometrické metody se principiálně neliší od systému pro voltametrické metody. Protože je ale amperometrie ve velké míře používaná pro určování látek v toku kapaliny, je nutné přizpůsobení elektrochemického článku. Používají se speciální amperometrické nádobky (detektory) umožňující měření v kapalném i plynném prostředí. Rovněž elektrody jsou pro účely měření plynných látek upraveny tak, aby byl z analyzovaného prostředí k elektrodě umožněn průchod pouze plynným molekulám. Příkladem je permeabilní polymerní membrána, která odděluje elektrodu od analyzovaného prostředí, a obecně jsou tyto senzory nazývány amperometrické membránové senzory (viz Obrázek 10) [1].

Metody amperometrie jsou využívány i v lékařské chemii např. pro určování kyslíku v krvi či pro biosenzory stanovující glukosu v krvi [1].



Obrázek 10 - Schéma amperometrického senzoru (Clarkův) [1]

3 Metody matematické analýzy pro zpracování dat

Jak již bylo popsáno v předchozích kapitolách, výstupem voltametrických měření je voltametrická křivka. Jedná se o závislost proudu procházejícího pracovní elektrodou za přítomnosti analytu v roztoku, na přiloženém potenciálu z vnějšího zdroje. Projevem analytu ve zkoumaném roztoku je průchod faradického proudu, který se na voltametrické křivce projeví jako proudový peak. Z polohy peaku lze určit druh látky obsažené v analytu. Z výšky peaku lze potom určit koncentraci dané látky. Po dohodě s vedoucím práce byla pozornost zaměřena na metody měření s monotónním průběhem voltamogramu, zejména pak metody DPV.

Cílem zpracování naměřených dat je tedy nalezení proudových peaků z voltamogramu a určení jejich polohy na ose potenciálu (horizontální osa) a výšky odečtením z osy proudu (vertikální osa).

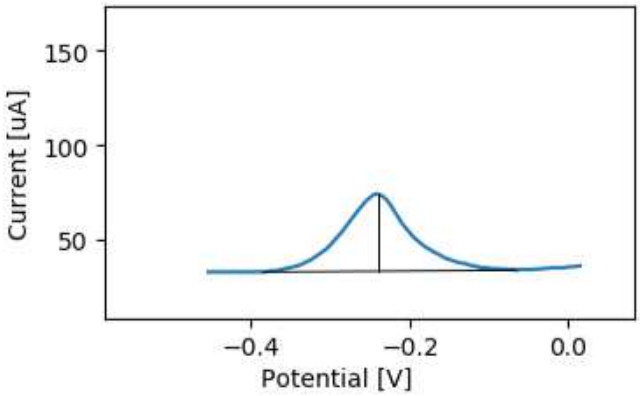
3.1 Definice požadovaných výsledků zpracování dat

Poloha proudového peaku je dána lokálním maximem proudu, tedy odečtení hodnoty potenciálu v místě, kde proud dosahuje v lokálním měřítku svého maxima. Aby mohla být zjištěna výška analyzovaného peaku, nestačí pouze odečíst hodnotu proudového maxima.

Z předchozího textu dále vyplývá, že průběh voltametrické křivky je zatížen šumem, jehož základní složkou je kapacitní proud způsobený elektrickou dvojvrstvou elektrody. Aby mohla být správně určena skutečná výška peaku, je nutné ve výsledcích hodnotu proudu způsobeného šumem odečíst.

Po určení polohy a maximální proudové hodnoty peaku zbývá tedy určit jeho skutečnou základnu. Základnu peaku bude tvořit úsečka, vycházející z určení lokálních minim kolem analyzovaného peaku. Nejdříve je nutné nalézt nejbližší lokální minimum vlevo a nejbližší lokální minimum vpravo od nalezeného proudového maxima. Následně je nutné vhodným způsobem úsečku přepočítat tak, aby nikde svým průběhem neprotínala průběh voltametrické křivky.

Ze známé základny peaku je již možné určit jeho výšku. Tu vypočítáme rozdílem mezi proudovým maximem a hodnotou proudu určenou souřadnicemi polohy proudového maxima na ose potenciálu a průsečíku základny peaku na ose proudu. Na následujícím obrázku (Obrázek 11) je pro ilustraci zobrazen proudový peak s vyznačenou základnou a výškou peaku.



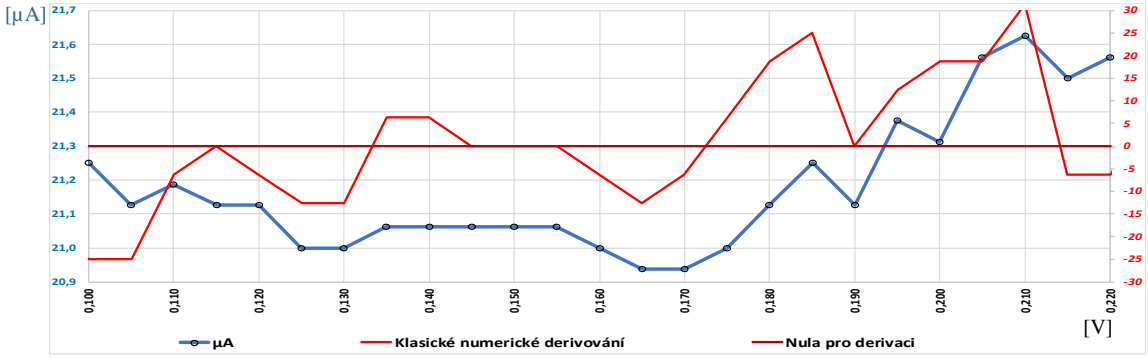
Obrázek 11 - Proudový peak s vyznačenou základnou a jeho výškou

3.2 Metoda filtrované první derivace

První metoda vychází z klasických postupů matematické analýzy o hledání maxim a minim matematických funkcí. Funkce má své lokální maximum tam, kde $f'(a_{(t)}) = 0$ a současně $f'(a_{(t-1)}) > 0$. Funkce má své lokální minimum tam, kde $f'(a_{(t)}) = 0$ a současně $f'(a_{(t-1)}) < 0$. V obou případech je t pořadí naměřených hodnot. Problém nastává v případě, kdy naměřené hodnoty jsou zatíženy byť i minimálním šumem. Klasické numerické derivování dle vzorce

$$der(x_{(t)}) = \frac{y_{(t-1)} - y_{(t+1)}}{x_{(t-1)} - x_{(t+1)}}, \qquad (3.1)$$

působí v takovém případě jako „zesilovač šumu“, jak je vidět na následujícím obrázku (Obrázek 12), kde jsou detekovány i šumové extrémy [3].



Obrázek 12 - Klasické numerické derivování
(modrá – naměřený proudový signál voltamogramu, červená – průběh první derivace)

V takovém případě je vhodné nahradit výše uvedený postup jiným algoritmem, který má vlastnost vyhlazování šumových efektů. Hodnota derivace je vlastně hodnotou tečny k průběhu. Potom se nabízí proložení některého okolí regresní přímkou (viz Obrázek 13) a převzetí její směrnice za náhradu derivace.

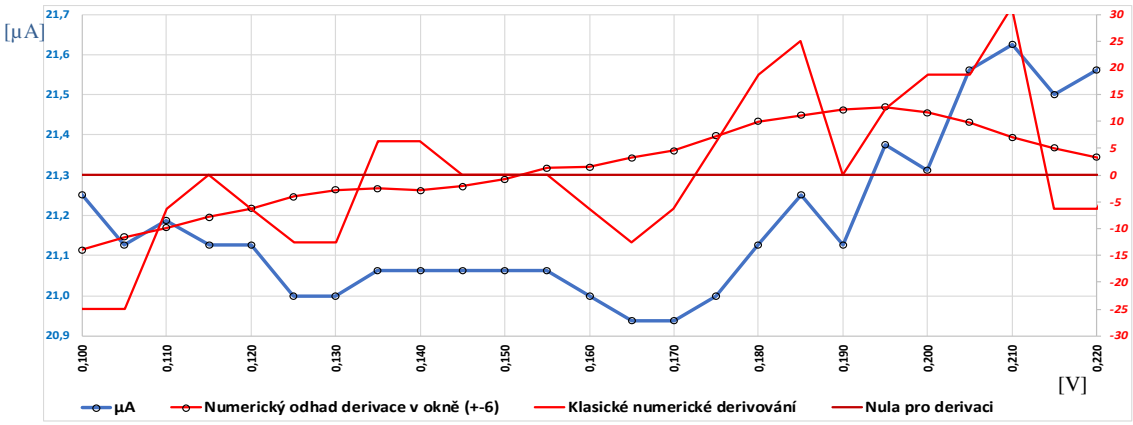
Pro rovnici takové přímky

$$\tilde{y}(x_{(t)}) = a_1 x_{(t)} + a_0 \qquad (3.2)$$

je uvedenou směrnicí hodnota a_1. Tu lze odhadnout ve zvoleném okolí $\pm k$ měřených hodnot následujícím výrazem pro bod $x_{(t)}$:

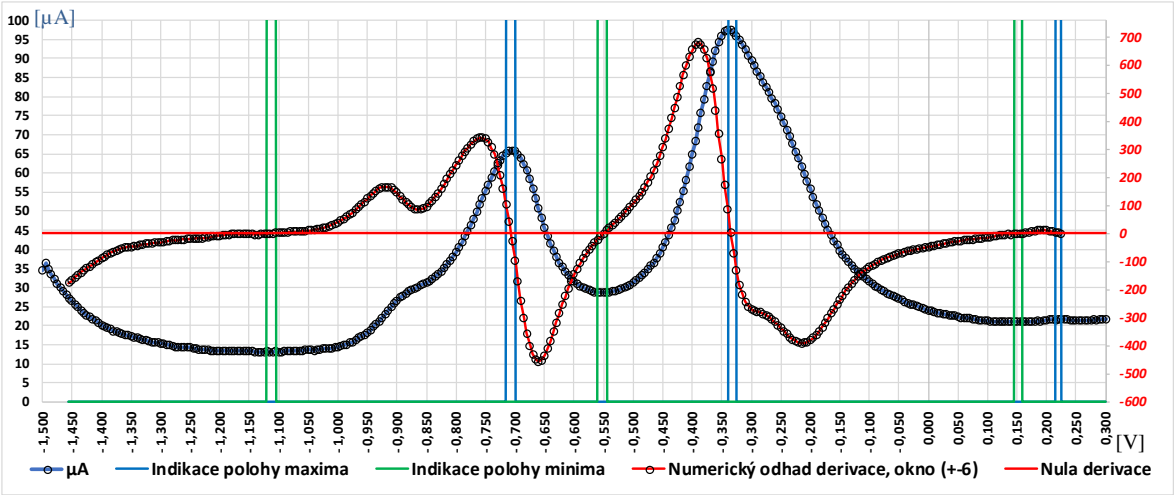
$$\widehat{a_1} = \frac{\sum_{i=t-k}^{t+k}(x(i) - \bar{x})\, y(i)}{\sum_{i=t-k}^{t+k}(x(i) - \bar{x})^2}, kde\ \bar{x} = \frac{1}{2k+1}\sum_{i=t-k}^{t+k} x(i) \qquad (3.3)$$

Uvedené vztahy vycházejí z obecných postupů uvedených v [4].



Obrázek 13 - Aplikace numerického odhadu derivace

Pokud využijeme shora popsaný postup k nalezení lokálních extrémů, může výsledek analýzy vypadat jako na následujícím obrázku (Obrázek 14).



Obrázek 14 - Metoda první filtrované derivace s indikací extrémů

Až na velmi vzácné případy, kdy odhad derivace bude nulový, je za extrém označena vlastně dvojce bodů, viz následující případ (viz Tabulka 1).

Tabulka 1 - příklad indikace polohy maxima

		Numerický odhad derivace, okno (+-6)	Indikace polohy maxima
V	**µA**		
…	**…**	…	
-0,720	**64,438**	160,1	
-0,715	**65,188**	104,3	
-0,710	**65,688**	41,6	*
-0,705	**65,938**	-26,5	*
-0,700	**65,688**	-96,8	
…	**…**	…	

V tomto případě je možné uvažovat za lokální extrém bod, ve kterém dochází při numerickém odhadu derivace ke změně znaménka (z kladného na záporné pro maximum a ze záporného na kladné pro minimum).

3.3 Metoda filtrované první a druhé derivace

V předchozím případě efekt filtrace náhodné složky zaručovalo proložení přímky metodou nejmenších čtverců okolím hodnoceného bodu zvolené velikosti. Pokud je do analýzy zahrnuta i druhá derivace, jedná se vlastně o proložení okolí hodnoceného bodu parabolou. Před zavedením druhé derivace zavedeme předpoklad, že napěťový krok měření je ve všech případech jednoho souboru dat shodný a teoreticky neměnný. Tento předpoklad vychází z parametrů měřicích metod, kdy je pro každé měření nastavený napěťový krok po celou dobu měření konstantní. V různých souborech dat však může být napěťový krok odlišný. Náhodné vlivy, působící na hodnotu napěťového kroku, lze pro účely analýzy zanedbat. Potom je možné zavést úpravu *x-ových* (napěťových) souřadnic. Platí tedy, že

$$x_{(t)} - x_{(t-1)} = konst. \qquad (3.4)$$

Potom transformace *x-ových* souřadnic v okolí vyhodnocovaného bodu bude

$$\begin{gathered} x_{(t-k)} \to -k \\ x_{(t-k+1)} \to -k+1 \\ \cdots \\ x_{(t-1)} \to -1 \\ x_{(t)} \to 0 \\ x_{(t+1)} \to +1 \\ \cdots \\ x_{(t+k-1)} \to k-1 \\ x_{(t+k)} \to k \end{gathered} \qquad (3.5)$$

kde k je šíře (jedné strany) zvoleného okna [4].

Tato transformace je pouze posunutím a změnou měřítka na ose *x* (napěťové). Nemění poměry na ose *y* (proudové). Takto transformovanými měřeními (metodou nejmenších čtverců) budeme prokládat parabolu:

$$y(x) = a_2x^2 + a_1x + a_0 \qquad (3.6)$$

jejíž první derivace bude

$$y^{(1)}(x) = 2a_2x + a_1 \qquad (3.7)$$

a její druhá derivace bude

$$y^{(2)}(x) = 2a_2. \qquad (3.8)$$

Následně je cílem určit hodnoty v bodě $x_{(t)} \rightarrow 0$

$$\begin{aligned} y_{(0)} &= a_0 \\ y^{(1)}{}_{(0)} &= a_1 \\ y^{(2)}{}_{(0)} &= 2a_2. \end{aligned} \qquad (3.9)$$

Hodnoty derivací po transformaci v ose *x*, v hodnoceném bodě potom budou přímo koeficienty prokládané paraboly. Protože v případě druhé derivace postačí pro další vyhodnocení znát pouze její znaménko, budeme dále pracovat pouze s koeficientem a_2, namísto $2a_2$. Obecný vztah pro proložení parabolou a výpočet koeficientů a_1 a a_2 se potom dle [4] zjednoduší na následující uvedené vztahy:

$$a_1 = \sum_{j=-k}^{+k} \frac{(j - \bar{J})}{\sum_{r=-k}^{+k}(r - \bar{J})^2} y(t-j) \qquad (3.10)$$

$$a_2 = \sum_{j=-k}^{+k} \frac{\left(j^2 - \bar{J^2}\right)}{\sum_{r=-k}^{+k}\left(r^2 - \bar{J^2}\right)^2} y(t-j) \qquad (3.11)$$

kde

$$\bar{J} = \frac{1}{2k+1}\sum_{j=-k}^{+k} j = \mathbf{0} \qquad (3.12)$$

$$\bar{J^2} = \frac{1}{2k+1}\sum_{j=-k}^{+k} j^2. \qquad (3.13)$$

Z uvedených vztahů lze potom vypočítat koeficienty dále označené jako b_j a c_j, které nezávisí na měřených hodnotách, ale pouze na zvoleném okně – tedy zvoleném okolí bodů *k*.

$$b_j = \frac{j}{\sum_{r=-k}^{+k} r^2} \qquad (3.14)$$

$$c_j = \frac{\left(j^2 - \bar{J^2}\right)}{\sum_{r=-k}^{+k}\left(r^2 - \bar{J^2}\right)^2} \qquad (3.15)$$

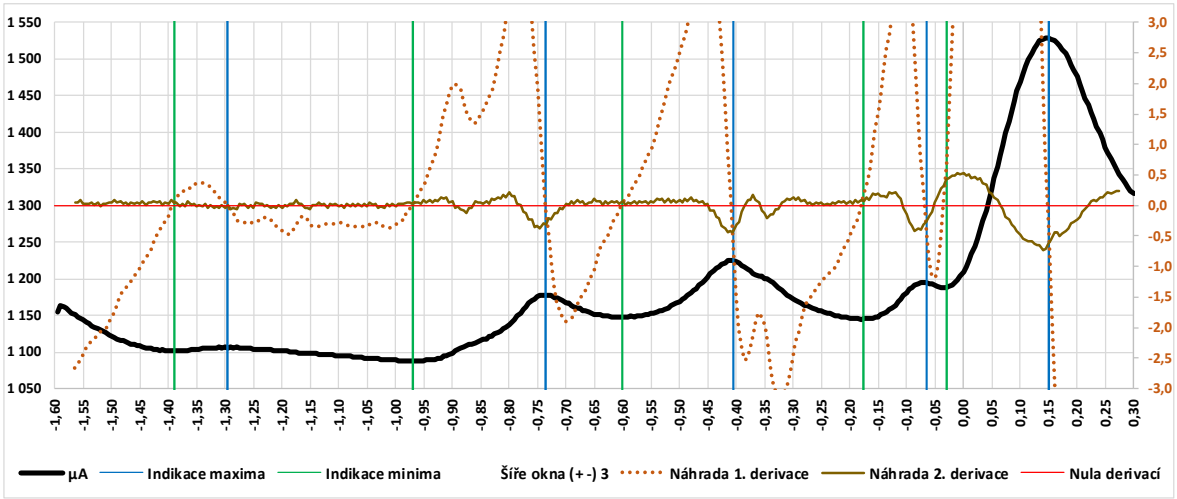
Po úpravě dostaneme vzorce pro výpočet koeficientů a_1 a a_2 takové:

$$a_1 = a_1(t) = \sum_{j=-k}^{+k} b_j y(t-j) \qquad (3.16)$$

$$a_2 = a_2(t) = \sum_{j=-k}^{+k} c_j y(t-j). \qquad (3.17)$$

Na základě těchto výpočtů dostaneme průběhy numerických odhadů první a druhé derivace, z níž lze již identifikaci maxim a minim provést následujícím způsobem. Bod [x(t), y(t)] označíme jak bod maxima, pokud se znaménko $a_1(t)$ liší od znaménka $a_1(t+1)$ a zároveň platí, že $a_2(t) > 0$. Odobným způsobem bod [x(t), y(t)] označíme jako bod minima, pokud se znaménko $a_1(t)$ liší od znaménka $a_1(t+1)$ a zároveň platí, že $a_2(t) < 0$.

Použití metody filtrované první a druhé derivace je potom znázorněno na následujícím obrázku (Obrázek 15).



Obrázek 15 - Metoda filtrované první a druhé derivace s indikací maxim a minim

3.4 Metoda pro zjištění základny a výšky peaku

Metoda pro zjištění základny peaku vychází z předchozích metod o nalézání maxim a minim funkce a předpokladem jsou tedy již známé polohy bodů určující maxima a minima z naměřeného souboru dat. Nejdříve se ke zvolenému peaku (detekovanému maximu) vybere nejbližší minimum vlevo na ose *x*, které má souřadnice [x(d), y(d)]. Současně se k tomuto peaku vybere nejbližší minimum vpravo na ose *x*, které má

souřadnice [x(h), y(h)].

Na základě takto určených bodů se spočte rovnice úsečky definovaná těmito okrajovými body [3].

$$y(t) = ax(t) + b \qquad (3.18)$$

Dosazením krajních bodů dostaneme

$$y(h) = ax(h) + b \qquad (3.19)$$

$$y(d) = ax(d) + b \qquad (3.20)$$

Odečtením druhé rovnice od první potom

$$y(h) - y(d) = a\big(x(h) - x(d)\big) \qquad (3.21)$$

Odtud lze vypočítat koeficienty úsečky *a* a *b*.

$$a = \frac{y(h) - y(d)}{x(h) - x(d)} \qquad (3.22)$$

$$b = y(h) - ax(h) \; nebo \; b = y(d) - ax(d) \qquad (3.23)$$

Když známe rovnici základní úsečky spojující nejbližší minima, lze provádět testování na body průběhu, které leží pod touto úsečkou. K tomuto účelu se ještě vypočítá pomocná veličina

$$\tilde{x} = \frac{x(h) + x(d)}{2} \qquad (3.24)$$

kde $\tilde{x}$ je střed *x-ových* souřadnic mezi body [x(d), y(d)] a [x(h), y(h)].

Vyhodnocení bodů pod úsečkou se potom určuje postupem:

1. Zavede $k = 0$, tedy počet bodů ležících pod úsečkou a $\bar{x} = 0$, tedy průměr *x-ových* bodů ležících pod úsečkou.
2. Pro $t = d + 1, \ldots, h - 1$ se spočtou hodnoty úsečky $\hat{y}(t) = ax(t) + b$. Pokud bude platit $\hat{y}(t) > y(t)$, započte se $k \leftarrow k + 1$ a $\bar{x} \leftarrow \bar{x} + x(t)$, jinak se pokračuje pro další *t*.
3. Pokud po skončení cyklu v bodě 2 bude $k = 0$, proces končí a $[x(d), y(d)]$ a $[x(h), y(h)]$, jsou krajními body základny a pokračuje se výpočtem výšky peaku viz dále.
4. Pokud po skončení cyklu v bodě 2 bude $k > 0$, dopočte se $\bar{x} \leftarrow \frac{\bar{x}}{k}$. Následně se rozhodne o poloze bodů pod danou úsečkou. Pokud bude
    a. $\bar{x} > \tilde{x}$ (body pod úsečkou se soustřeďují blíže $x(h)$), ubere se jedno měření

zprava, tj. $h \leftarrow h - 1$ a pokračuje výpočtem nové úsečky dle postupu v úvodu kapitoly s takto upravenými body úsečky. Následně se provede nové testování od bodu 1,

b. $\bar{x} \leq \tilde{x}$ (body pod úsečkou se soustřeďují blíže $x(d)$), ubere se jedno měření zleva, tj. $d \leftarrow d + 1$ a pokračuje výpočtem nové úsečky dle postupu v úvodu kapitoly s takto upravenými body úsečky. Následně se provede nové testování od bodu 1.

Po ukončení testování bodů ležících pod úsečkou již známe takovou úsečku, pod kterou žádné body neleží. Nyní lze vypočítat výšku peaku ze známých souřadnic maxima, zde jako $[x(p), y(p)]$ a základny peaku.

$$výška_{peaku} = y(p) - a\,x(p) - b \qquad (3.25)$$

4 Software pro zpracování dat

Hlavním cílem této práce je vytvoření počítačového programu, který automaticky zpracovává sady naměřených dat získaných z voltametrických měření. Výstupem zpracování dat je základní analýza polohy a velikosti proudových peaků z voltametrické křivky, které jsou směrodatné pro určování analytu obsaženého v analyzovaném roztoku. Nejdříve však bylo třeba definovat základní vlastnosti vstupních dat, požadavky na jejich zpracování a požadovaný výstup z jejich analýzy. Na základě těchto požadavků lze určit vhodné programové nástroje a strukturu připravovaného softwaru.

4.1 Definice základních požadavků na vývoj softwaru

Cílem práce je vytvořit samostatně fungující software na běžné počítačové platformě. Ve spolupráci s vedoucím práce tedy byly definovány základní požadavky na účel a funkčnost výsledného programu, které byly následující:

1) Software bude určen pro zejména pro automatické předzpracování voltametrických dat získaných z rozsáhlých souborů měření. Jde zejména pro rychlý náhled opakovatelnosti jednotlivých měření, příp. rychlou analýzu chybných měření či odlehlých hodnot.
2) V případě potřeby bude program umožňovat i analýzu jednotlivých vzorků.
3) Veškerá komunikace s uživatelem bude probíhat prostřednictvím grafického rozhraní, neboli GUI (*Graphic User Interface*), které bude umožňovat i individualizaci konfigurace formátu vstupních dat.
4) Zpracované výstupy budou opět v grafické podobě grafů s jednoznačnou identifikací nalezených peaků v grafu a jejich numerickou interpretací.
5) Shromážděné numerické hodnoty peaků z jednotlivých průběhů bude možné hromadně exportovat pro další potřeby ve formátu umožňujícím jejich další zpracování v tabulkovém kalkulátoru nebo textovém procesoru.

4.1.1 Definice vstupních dat, automatické nastavení parametrů

Protože realizovaný program má za úkol zpracovávat již získaná data z voltametrických měření, je třeba formát a strukturu vstupních dat přizpůsobit formátu dat, která lze získat ze specializovaných programů dodávaných k přístrojům pro elektrochemická měření. Vedle proprietárních formátů většinou všechny umožňují ukládat nebo alespoň exportovat naměřené hodnoty v prosté textové formě, kdy jsou data zapsána do souboru na řádcích a jednotlivé datové prvky (sloupce) jsou odděleny textovým oddělovačem, např. čárkou,

středníkem, apod. Soubory tohoto typu jsou nejčastěji ukládány s příponou „.txt“ nebo „.csv“.

Takový formát umožňuje jednoduché čtení a zpracování dat na jakékoliv počítačové platformě, případně také s využitím běžných kancelářských aplikací. Nevýhodou je, že není předepsána žádná struktura zápisu dat. Každý soubor tak např. v závislosti na použitém měřicím zařízení může používat jiný oddělovač dat, a vlastní naměřená data nemusí a zpravidla nezačínají na shodném řádku souboru. Předpokladem pro vývoj software tedy je, že vstupující data budou výhradně ve formátu prostého textu, jak je popsáno výše.

Požadavkem je uživatelsky snadné nastavení parametrů souborů, které do programu vstupují. Vzhledem k velké variabilitě struktury dat je nutné, aby uživatel tyto parametry nastavil ručně. Protože program nebyl primárně určen pro obecné použití, ale pro rychlou analýzu datových souborů z konkrétních zařízení, používaných na Katedře technologií a měření FEL ZČU, vznikl požadavek na předkonfigurované vstupní nastavení pro tyto datové soubory, které si uživatel jen navolí ve vstupním okně programu.

Definice základních nastavitelných parametrů vstupního souboru je následující:

- typ oddělovače jednotlivých prvků (sloupců);
- typ oddělovače desetinných míst u číselných dat;
- typ oddělovače číselných řádů (tisíců), pokud je použit;
- číslo řádku, na kterém začínají naměřená data;
- číslo řádku, který obsahuje datovou hlavičku (popisky datových sloupců), pokud je taková hlavička v souboru obsažena.

Na následujícím příkladu vstupních dat je použita mezera jako oddělovač sloupců dat a oddělovač desetinných míst u čísel je tečka. Číselné údaje jsou zapsány v exponenciální formě, avšak tento způsob zápisu čísel není neobvyklý. Naměřená data začínají na třetí řádce a předchozí údaje neobsahují hlavičku dat.

```
1  test
2  Sensor: SPE|0
3  -1.60000E+000 8.66625E+002
4  -1.59500E+000 1.04888E+003
5  -1.59000E+000 1.05381E+003
6  -1.58500E+000 1.05038E+003
7  -1.58000E+000 1.04706E+003
8  -1.57500E+000 1.04294E+003
```

Příklad zápisu naměřených dat ve formě prostého texu

Jak vyplývá z uvedeného příkladu naměřených dat, dalším požadavkem na software je uživatelská identifikace sloupců, tedy přiřazení významu obsažených dat k měřené veličině a jejích fyzikálních jednotek. Uživatel zpravidla ze zkušenosti zná význam dat, nebo je schopný je určit z dokumentace či nastavení měřicího přístroje (zejména pokud měřicí přístroj zaznamenává více údajů než je přikládaný potenciál a proud procházející pracovní elektrodou, např. viz dále Obrázek 23 - Okno aplikace s načtenými daty – struktura dat ve formátu z měřicího systému Autolab).

4.1.2 Dávkové zpracování dat

Zejména v případech, kdy je zapotřebí analyzovat větší množství obdobných datových sad, je nevhodné, aby uživatel vkládal data po jednom datovém souboru, opakovaně nastavoval vstupní parametry a prováděl vyhodnocení pro každý datový soubor zvlášť.

Z těchto důvodů byl zaveden požadavek, že software bude schopen data zpracovávat v dávkách, při hromadném nastavení vstupních parametrů a výstupem budou data a vizualizace pro všechny zpracovávané soubory najednou.

Současně je však zaveden nutný předpoklad, že všechny vkládané soubory v jedné dávce budou mít totožnou strukturu i obdobný rozsah naměřených dat. Bez tohoto předpokladu by nebylo možné nastavit parametry struktury dat pro všechna měření současně a vedlo by to k programovým chybám nebo nevypovídajícím výsledkům analýzy.

4.1.3 Vizualizace dat a výsledky analýzy

Poté, co budou data dle výše uvedených požadavků a předpokladů zpracována, je možné provést jejich analýzu. Vlastní analýza je teoreticky popsána v kapitole 3 a její implementace do softwaru bude popsána dále. Výstupem analýzy jsou opět data, která je zapotřebí předat uživateli vhodnou formou tak, aby měla vypovídající hodnotu o provedené analýze.

Vhodným způsobem předání informace uživateli je grafická vizualizace dat, jak bylo řečeno v úvodu této podkapitoly. Ve spojení s numericky či textově předanými informacemi je možné výsledky analýzy správně interpretovat.

Požadavkem na výstup dat je tedy nalezení vhodného prostředku a způsobu vizualizace dat. Protože se jedná o dvourozměrnou vizualizaci, bude vhodné data a výsledky zobrazit formou grafů a případně tabulek.

Protože platí, že data mohou být zpracovávána z jednoho souboru, ale i formou dávky

většího množství souborů, je nutné zvolit vhodnou formu zobrazení výsledků, aby byla za všech podmínek přehledná a jednoznačná.

Posledním požadavkem je použitelnost výsledků pro další účely nebo pozdější kontrolu. V tomto případě je zaveden požadavek na export dat a to jak v kombinované grafické formě, tak i v prosté textové formě, např. ve formátu „.txt“ nebo „.csv“.

4.2 Programovací jazyk a použité nástroje

Vývoj software je zpracován na stále nejběžnější platformě osobních počítačů s operačním systémem Microsoft Windows. Vzhledem k povaze software se nepředpokládá jeho využití na mobilních platformách. Předpoklad použití software je tedy v systémech Windows, ale je třeba vzít v úvahu i široce využívané systémy Mac OS či Linux.

Pro vývoj aplikace byl výhodně zvolen programovací jazyk Python, který je multiplatformní, objektově orientovaný a současně poměrně snadno uchopitelný. Přesto tento programovací jazyk nabízí širokou škálu knihoven a rozšíření, včetně možnosti tvorby grafických uživatelských rozhraní. Více o Pythonu, včetně dokumentace je uvedeno v [5].

Při výběru vhodné knihovny pro tvorbu grafického rozhraní bylo uvažováno více kritérií a to zejména kompatibilita se současně hledanou knihovnou pro tvorbu grafů a vizualizaci dat.

I když je software vyvíjen primárně pro Windows, měla by být ponechána možnost do budoucna upravit program i pro Mac OS, příp. Linux. Další kritérium při výběru grafických knihoven tedy je, aby i zde byla zachována multiplatformní vlastnost jazyka Python.

Všechny zmíněné požadavky splňuje grafická knihovna Kivy, určená pro jazyk Python. Jedná se o open source knihovnu založenou na OpenGL ES 2 akceleraci, která je kromě výše uvedených kritérií také poměrně snadná [6].

Kivy disponuje dostatečnou dokumentací a má velkou komunitu začínajících i zkušenějších programátorů. Jedná se o velkou výhodu, neboť většinu řešení konkrétních problémů a nesrovnalostí lze nalézt na množství internetových diskuzí, vč. uváděných příkladů použití.

Grafické rozhraní v Kivy se vytváří skládáním jednotlivých grafických prvků (*widgets*). S tou výhodou, že je prvky možné definovat buďto přímo v kódu Python, ale současně lze

použít vlastní jednoduchý jazyk – Kv Design Language. Pomocí tohoto integrovaného jazyka lze základní grafické rozhraní nadefinovat mimo hlavní zdrojový kód programu a tím je základní grafická část oddělena od výkonné části kódu. Lze tak docílit větší přehlednosti a současně to vede k jednodušším úpravám struktury grafického rozhraní [6].

Poslední jmenovanou výhodou Kivy je jeho kompatibilita se zvolenou knihovnou pro tvorbu grafické vizualizace. Tou je knihovna Matplotlib. Jak již napovídá název, jedná se o knihovnu pro tvorbu grafů podobných programu MATLAB. Matplotlib nabízí možnosti tvorby velké škály statických i interaktivních grafů, je opět snadný a dobře modifikovatelný. Opět je k dispozici podrobná dokumentace vč. příkladů použití [7].

Aby byl výsledný program samostatně spustitelný, bez nutnosti instalovat Python a všechny potřebné knihovny, je zapotřebí použít nástroj k jeho sestavení. V dokumentaci ke knihovně Kivy je také popsán způsob sestavení aplikace pro Windows pomocí PyInstaller. Tento nástroj vytvoří spustitelný soubor „.exe“ a přiřadí programu všechny potřebné knihovny a soubory. Díky tomu je program samostatně spustitelný a přenositelný [6].

V následujícím přehledu je uvedena rekapitulace použitých nástrojů vč. verzí, které jsou v době vývoje programu kompatibilní:

- Python 3.6.7
- Kivy 1.10.1
- Kivy-Garden 0.1.4 (pro knihovnu garden.matplotlib, viz [8])
- Matplotlib 3.0.3
- PyInstaller 3.4

4.3 Základní struktura softwaru, vývojové diagramy

Pro zachování přehlednosti zdrojových kódů a pro jejich snadnější úpravy je software rozčleněn do několika dílčích skriptů.

4.3.1 Hlavní řídicí skript DAVM.py

Hlavní skript obsahuje vlastní definici aplikace a propojuje všechny dílčí skripty a pomocné knihovny do jednoho funkčního celku. Strukturálně je rozdělen do tříd, kdy každá třída definuje konkrétní okno uživatelského rozhraní a jeho funkce.

Je zde tedy obsažena hlavní logika celého softwaru, ale není zde přímo definován vzhled aplikace a grafické nastavení vizuálních prvků.

4.3.2 Skript grafického rozhraní DAVM.kv

Díky poznatkům z kapitoly 4.2 je využito jednoduchého jazyka Kv Design Language k separaci vlastní grafické části softwaru od jeho logické části.

Skript grafického rozhraní definuje, jak bude vlastní aplikace vypadat. Zde se definují jednotlivé grafické prvky aplikace (*widgets*) a jejich vlastnosti [6]. Kromě nastavení rozměrových parametrů, barev, písem, umístění, apod., je zde také možné nastavit základní chování prvků. V případě událostí, které mají nastat např. po stisku tlačítka, se zde definuje odkaz na funkci hlavního řídicího skriptu, který opět vykonává logickou část události.

DAVM.kv se načítá automaticky s hlavním skriptem DAVM.py při spuštění aplikace a provede základní nastavení a zobrazení grafického rozhraní.

V některých případech je však nutné měnit definované vlastnosti grafických prvků přímo za běhu aplikace, nebo je dokonce nutné některé grafické prvky přidat či odebrat.

Grafická knihovna Kivy má již zmíněnou výhodu, že je možné grafické prvky definovat jak v jazyce Kv Design Language, tak přímo v jazyce Python. Změny vlastností těchto prvků jsou tedy v případě potřeby prováděny přímo v hlavním řídicím skriptu. Shodným způsobem jsou někdy ale vytvářeny i úplně nové prvky, a to většinou z důvodu, že jejich vlastnosti nebo počet prvků se odvíjejí až na základě uživatelem vkládaných dat.

4.3.3 Konfigurační soubor config.cfg

Konfigurační soubor neobsahuje žádné vlastní funkce aplikace, avšak slouží k uložení a zpětnému načtení některých nastavení a parametrů. O použití konkrétních parametrů je více uvedeno v dalších kapitolách.

```
1  *****DAVM - Data Analysis of Voltammetric Measures, ver. 0.91 -
   configuration file*****
2  *****DAVM - Datová Analýza Voltametrických Měření - Bc. Oldřich Holý, FEL
   ZČU 2019****
3
4  # Domácí adresář
5  PATH = C:\
6
7  # Povolené typy souborů
8  FILETYPES = .txt,.csv,.pss
9
10 # Minimánlní šířka okna pro derivaci
11 DWIN = 3
12
13 # Filtr nechtěných peaků menších než: (Výška největšího peaku = 1)
14 PEAKFILTER = 0.2
15
16 # Profily CSV
```

```
17 PROFILES HINT:
18 profile_name column_divider first_data_line header_line decimal_point
   thousands_divider is_header?
19
20 <CSVPROFILES>
21 PalmSens mezera 3 '' . neni True
22 Autolab ; 2 1 , neni False
23 </CSVPROFILES>¨
```

Konfigurační soubor config.cfg

Jedná se o parametry softwaru, jejichž modifikace není uživateli přístupná přímo z uživatelského grafického rozhraní. Současně však tento způsob umožňuje změnu těchto parametrů, aniž by bylo nutné vstupovat přímo do zdrojových kódů aplikace.

```
1  class DAVM(App):
2      def build(self):
3          config.loadcfg()
4          return ScreenManager()
```

Načtení konfiguračního souboru ve skriptu DAVM.py

Načtení konfiguračního souboru je provedeno ihned při spuštění a sestavení aplikace. Provádí se funkcí v řádku 3 kódu výše. Tato funkce je součástí skriptu popsaného v další kapitole.

4.3.4 Konfigurační skript config.py

Konfigurační skript config.py je v podstatě rozhraním mezi konfiguračním souborem a vlastními funkcemi aplikace. Obsahuje funkce, které slouží jednak k načtení konfiguračních parametrů do aplikace a jednak funkce pomocí kterých si aplikace sama aktualizuje konfigurační soubor.

```
1  def loadcfg():
2      global path
3      global d_win
4      global peak_filter
5      global filetypes
6      global csvprofiles
7      csvprofiles = []
8      s1 = 'PATH'
9      s2 = 'FILETYPES'
10     s3 = '<CSVPROFILES>'
11     s4 = '</CSVPROFILES>'
12     s5 = 'DWIN'
13     s6 = 'PEAKFILTER'
```

Definice načítaných parametrů konfigurace – config.py

Ve funkci *loadcfg* jsou nadefinovány globální proměnné pro pozdější použití napříč všemi třídami aplikace. Těmto proměnným jsou přiřazeny hodnoty z parametrů konfiguračního souboru. Parametry jsou definovány klíčovými slovy, díky kterým jsou v konfiguračním

souboru identifikovány. Je zapotřebí dodržet syntaxi zápisu těchto parametrů, aby je načítací funkce správně identifikovala.

Jednořádkové parametry jsou zapsány v syntaxi:

```
1  klíčové_slovo = hodnota_1,(hodnota_2),…
```

Pokud se jedná o skupinu parametrů, zapisují se způsobem podobným HTML kódu:

```
1  <klíč_začátek>
2  param1 param2 param3 …
3  param1 param2 param3 …
4  </klíč_konec>
```

4.3.5 Skript pro práci s datovými soubory csvreader.py

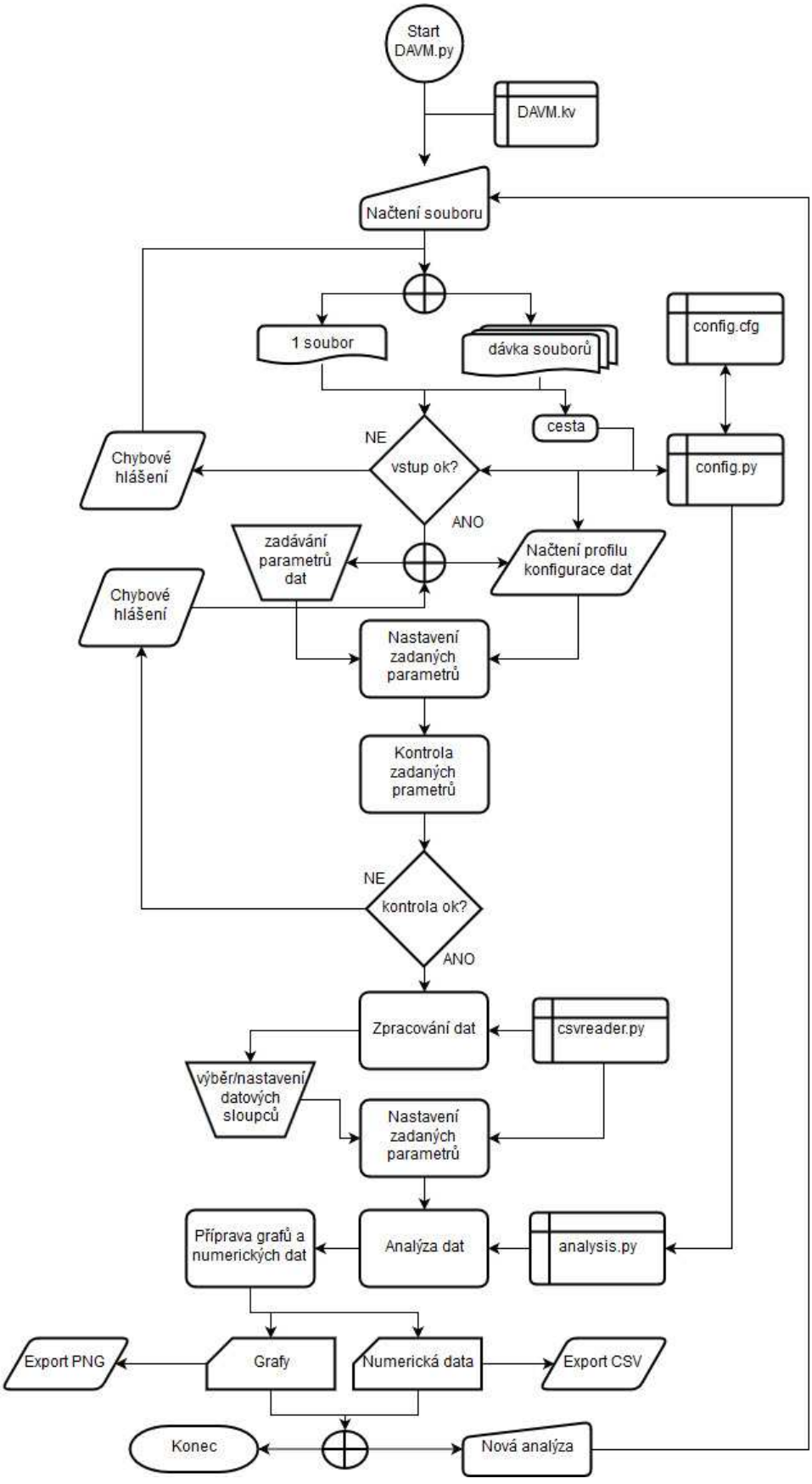
Tento skript byl zaveden zejména z důvodu lepší přehlednosti hlavního skriptu DAVM.py. Během načítání datových souborů dochází k řadě výpočtů a změn parametrů aplikace. Funkce, které data načítají do aplikace, jsou samy o sobě dosti rozsáhlé. Z tohoto důvodu probíhá samotné načtení datového souboru, jeho úprava a uložení hodnot do proměnných vstupujících do aplikace mimo hlavní řídicí skript.

4.3.6 Skript matematické analýzy dat analysis.py

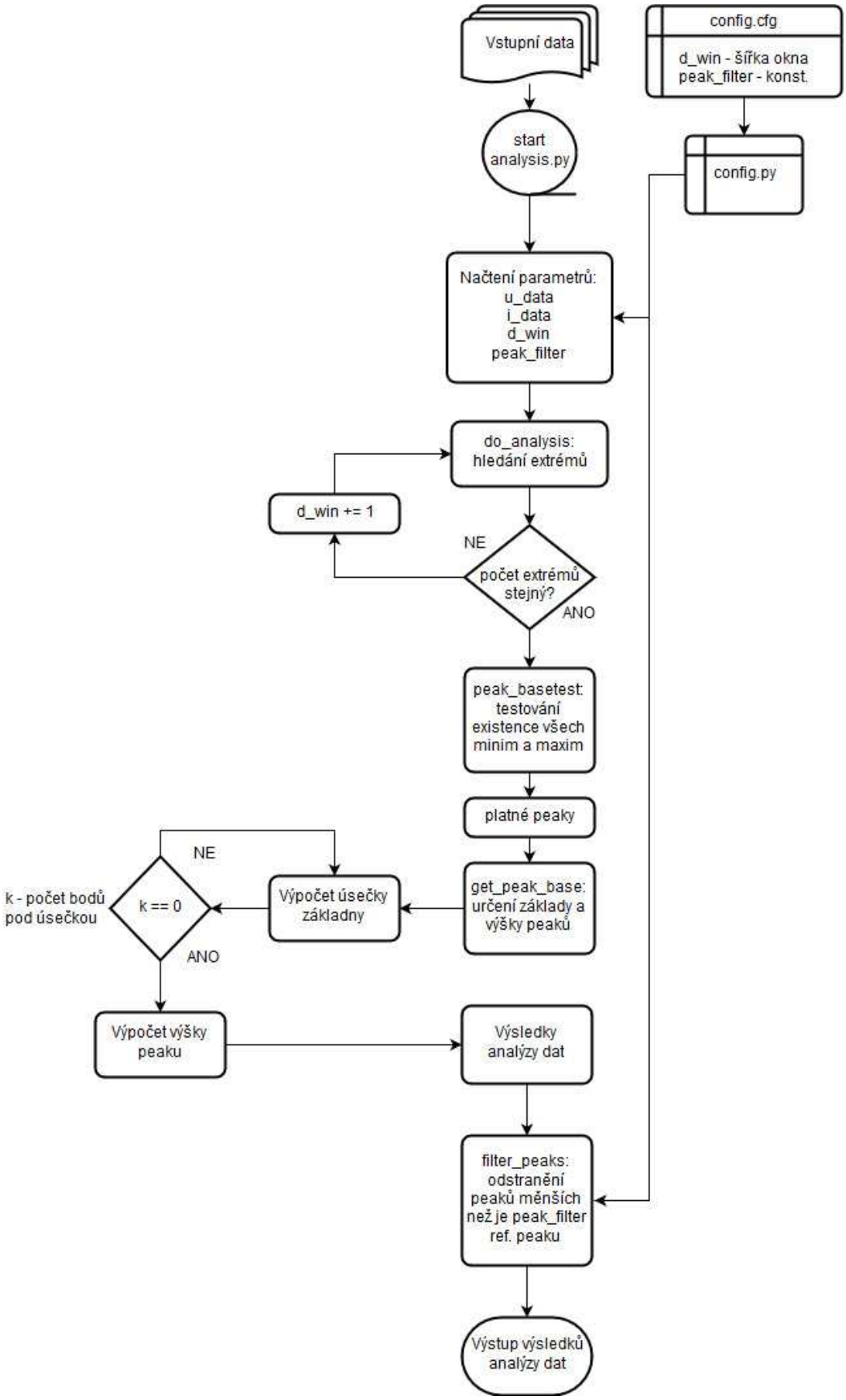
Do analytického skriptu vstupují dříve připravená data, která jsou validní a jsou identifikována svým fyzikálním významem. Skript obsahuje dílčí funkce, které jsou v rámci analýzy volány i vícekrát. Dále je zde řídicí funkce, do které data vstupují, jsou analyzována pomocí dílčích funkcí skriptu a výstupem funkce jsou výsledky analýzy.

Opět se tedy jedná o složitější kód, který by působil nepřehlednost v hlavním skriptu aplikace. Podstatnějším důvodem, proč je skript matematické analýzy oddělen, je možnost upravovat použitou matematickou metodu, případně ji do budoucna v případě potřeby nahradit jednoduše jinou metodou. Stačí přitom pouze zachovat formát vstupních dat a formát výsledků.

4.3.7 Vývojové diagramy



Obrázek 16 - vývojový diagram chodu aplikace DAVM



Obrázek 17 - Vývojový diagram běhu analýzy dat - analysis.py

4.4 Načtení vstupních dat a konfigurace parametrů

V úvodním okně aplikace uživatel volí datové soubory, které se mají načíst a s nimiž se bude dále pracovat. Po načtení souboru nebo více souborů je zobrazen uživateli náhled na strukturu datového souboru. Na základě vizuální kontroly struktury dat uživatel nastaví parametry struktury tak, aby je bylo možné programem zpracovat k dalšímu použití.

4.4.1 Základní vzhled aplikace

Nejprve je nutné stanovit základní vzhled a rozložení vstupní části aplikace. Je zavedena zásada, aby celá aplikace byla uživatelsky přátelská, konzistentní, intuitivní a nenáročná.

Z tohoto důvodu je stanovena základní koncepce aplikace následovně. Rozložení okna aplikace je v celém rozsahu zachováno obdobným stylem. Uživateli je vždy povoleno používat pouze prvky aplikace, které mají v daném okamžiku význam, resp. jsou nutnou podmínkou pro ovládání dalších prvků. S tím souvisí také posloupnost ovládacích oken. Software je navržen tak, aby se choval jako průvodce. Uživatel je tedy naveden, aby nejdříve nahrál vstupní data, poté nastavil jejich parametry a až následně je mu umožněno pokračovat ke specifikaci dat a jejich přípravě do analýzy apod.

V prvním kroku je navržena minimální velikost okna aplikace optimálně tak, aby se do okna daly přehledně umístit všechny ovládací a grafické prvky v celém rámci běhu aplikace a současně aby byla aplikace ovladatelná na počítačích s běžným rozlišením obrazovky.

```
1  from kivy.config import Config
2  Config.set('graphics', 'resizable', True)
3  Config.set('graphics', 'minimum_width', '1100')
4  Config.set('graphics', 'minimum_height', '650')
5  Config.set('graphics', 'width', '1200')
6  Config.set('graphics', 'height', '650')
```

Nastavení minimální velikosti grafického okna aplikace – DAVM.py

Za běžné rozlišení dnešních displejů se v této práci uvažuje 1366x768 pixelů, avšak aplikace by měla být ovladatelná i na rozlišeních o něco nižších jak je zřejmé z kódu výše. Toto nastavení se doporučuje provést již v hlavičce hlavního skriptu aplikace, a sice za pomoci třídy *kivy.config*. V některých případech totiž nastává problém, že pokud je konfigurace okna vyvolána až po načtení ostatních knihoven, přestane být funkční. Toto nastalo např., pokud byla konfigurace volána až po načtení grafické knihovny Matplotlib.

Specifikace a rozložení základních ovládacích prvků okna se s výhodou provádí za pomoci Kv Design Language ve skriptu DAVM.kv [6].

```
<ScreenManager>
    analysis: analysis
    csvscreen: csvscreen
    main: main
    id: screen_manager

    Main:
        id: main
        name: 'Main'
        manager: 'screen_manager'
    Csvscreen:
        id: csvscreen
        name: 'Csvscreen'
        manager: 'screen_manager'
    Analysis:
        id: analysis
        name: 'Analysis'
        manager: 'screen_manager'
```

Úvodní definice oken aplikace v jazyce Kv Design Language – DAVM.kv

Na začátku souboru DAVM.kv je nadefinován tzv. správce oken (*ScreenManager*). Tento správce má na starosti jednak vizuální přechod mezi jednotlivými grafickými okny a současně umožňuje z jednoho okna přistupovat k prvkům jiného okna, i když není vizuálně dostupné. Každé grafické okno je definováno vlastní třídou v hlavním skriptu psaném v jazyce Python. Oknem není v tomto případě myšlena samotná instance okna prostředí Windows, ale pouze obsah aktuální instance aplikace [6].

Následně se v okně definuje rozložení ovládacích a grafických prvků.

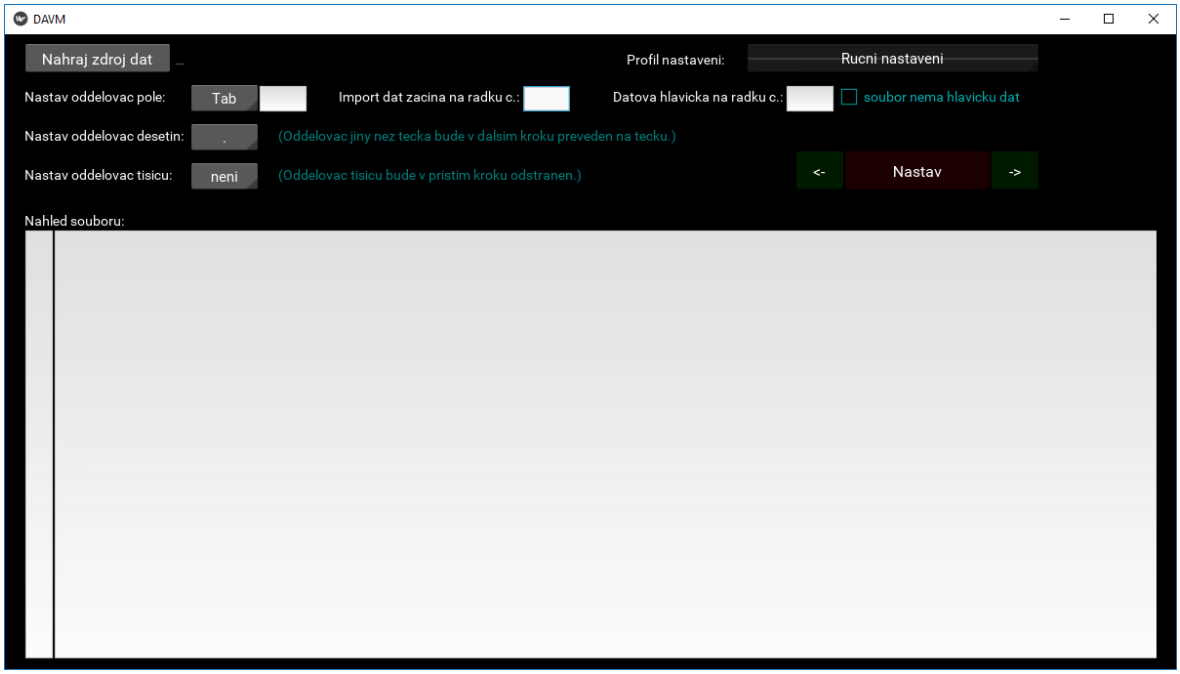
```
    BoxLayout:
        orientation: 'vertical'

        BoxLayout:
            size_hint: (1, None)
            height: 200
            orientation: 'vertical'

            BoxLayout:
                height: 70
                orientation: 'horizontal'
                padding: (20, 10, 0, 0)

                Button:
                    id: load_btn
                    text: "Nahraj zdroj dat"
                    font_size: 16
                    size_hint: (None, 1)
                    width: 150
                    on_release: root.show_load()

                Label:
                    id: project
                    text: "..."
                    font_size: 11
```

```
26                  text_size: self.width, None
27                  italic: True
28                  max_lines: 3
29                  padding_x: 5
30                  halign: 'left'
31                  size_hint: (None, 0.9)
32                  width: 440
```

Ukázka části kódu definující rozložení aplikace a konfiguraci prvků – DAVM.kv

Kivy nabízí pro určení základního rozložení grafických prvků řadu přednastavených grafických rozložení (*Layout*). Každé rozložení má svoje předdefinované vlastnosti a je blíže popsáno v dokumentaci [6].

Pro aplikaci bylo zvoleno rozložení *BoxLayout*. Toto rozložení vytvoří v grafickém okně neviditelný kontejner, do něj jsou vkládány grafické prvky jeden po druhém, tak jak jsou postupně definovány ve skriptu. Volbou orientace lze určit, zda se prvky vkládají vedle sebe, nebo pod sebe. Toto rozložení je vhodné, pokud není nutné vkládat prvky na nahodilé pozice okna a naopak je snahou docílit jednoduchého vkládání prvků v řadách či sloupcích [6]. Tímto způsobem jsou nadefinovány všechny prvky úvodního okna a jejich parametry, případně výchozí hodnoty.



Obrázek 18 - Úvodní obrazovka aplikace DAVM (Datová Analýza Voltametrických Měření)

Na obrázku výše (Obrázek 18) je zobrazena výsledná podoba úvodního okna aplikace. Zde je uživatel nabádán k nahrání zdroje dat a následně je možné nastavovat příslušné parametry datového souboru, na základě náhledu dat zobrazených v textovém poli obrazovky.

4.4.2 Načtení naměřených dat do aplikace

V kapitole 4.1.1 jsou vydefinovány předpoklady a podmínky na typy souborů, se kterými bude software schopen pracovat. Jedná se tedy o soubory s naměřenými daty, které jsou zapsány ve formátu prostého textu.

Na základě vymezení typu vstupních souborů nahrávaných do aplikace je vhodné zamezit uživateli, aby do aplikace mohl nahrávat jiné typy souborů. To by vedlo s největší pravděpodobností k pádu aplikace nebo chybám, které jsou v rozporu s dalšími postupy. Samozřejmě může dojít k situaci, kdy uživatel omylem vybere soubor jiného typu. To se může snadno přihodit, pokud se ve složce nachází více souborů stejného názvu a liší se pouze svou koncovkou.

Vhodným řešením předcházení těmto chybám je přímo určit programu, jaké typy souboru povolí nahrát. V případě, kdy se uživatel pokouší nahrát nepodporovaný typ, je vhodné jej o tom informovat chybovou hláškou a navrhnout mu dostupné možnosti.

Nejčastějšími typy souborů, do kterých jsou zapisována data v textovém formátu, jsou soubory typu „.txt“ a „.csv“. Mimo tyto základní typy souborů je do základu aplikace přidán ještě typ souboru „.pss“. V tomto případě se jedná o výstupní soubory softwaru PSTrace, určeného pro měřicí přístroje firmy PalmSens, využívané v laboratořích Katedry technologií a měření. Výrobce se v tomto případě rozhodl ukládat data do souboru se svou vlastní koncovkou. Rozhodující však je, že data jsou v souborech „.pss“ zapsána ve formátu prostého textu, shodně jako v případě prvních dvou jmenovaných typů. Je tedy splněn předpoklad a požadavek na zpracovávaný typ souboru.

Jak vyplývá z předchozího odstavce, je zapotřebí vzít v úvahu, že uživatel může chtít použít typ souboru, který není aplikací podporován. Uživatel ale ze zkušenosti ví, že se jedná o soubor dat zapsaných ve formátu prostého textu. Pokud by tedy byla aplikace pevně určena pouze pro některé typy souborů, vyžadovalo by případné rozšíření těchto typů nutný zásah do zdrojového kódu programu.

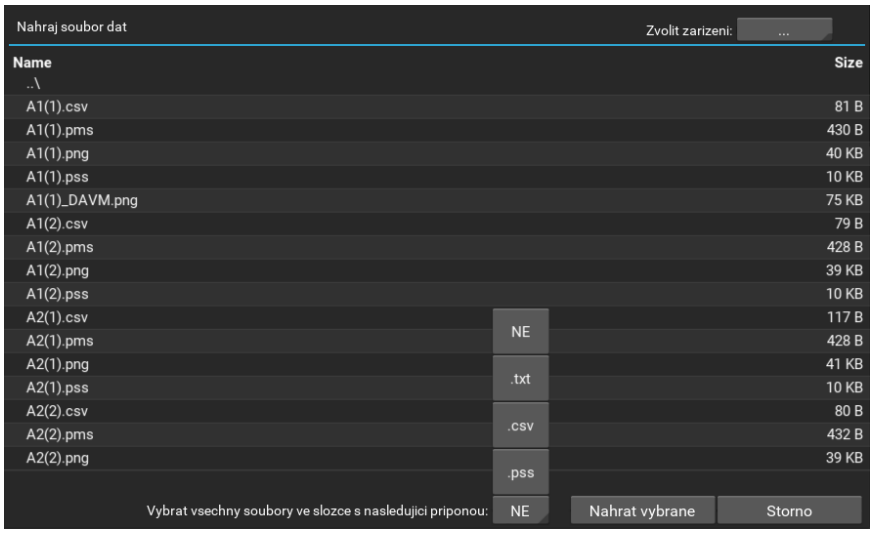
Aby byla zachována rozšiřitelnost aplikace pro budoucí použití s dalšími typy souborů při současném zajištění určité uživatelské bezpečnosti, je vhodné nalézt řešení v rovině mezi uživatelským rozhraním a zdrojovým kódem.

Z výše uvedených důvodů jsou povolené typy souborů zapsány do konfiguračního souboru aplikace (viz kapitola 4.3.3), odkud software parametry načítá do svých funkcí, avšak nedochází k přímé interakci s uživatelem v grafickém rozhraní aplikace.

Dalším požadavkem na chod aplikace je možnost načítat soubory do aplikace jako dávku souborů nebo jako samostatný soubor. Výše uvedené požadavky na povolené typy souborů je nutné dodržet i v případě nahrávání většího množství souborů.

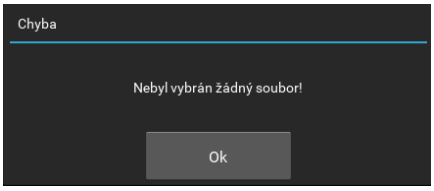
V kapitole 4.1.2 byl zaveden nutný předpoklad, že soubory nahrávané v režimu dávky budou mít vždy shodnou strukturu. Rovněž se tedy předpokládá, že v rámci dávky se budou analyzovat soubory z jednoho typu měření a budou všechny shodného typu souboru.

Vzhledem k těmto předpokladům a vzhledem k podporované funkcionalitě prvku správce souborů v Kivy (*FileChooser*) jsou zavedeny dvě možnosti výběru souborů [6]. V první variantě vybírá uživatel jediný soubor ke zpracování, a to klasickým označením souboru ve správci souborů a potvrzením stiskem tlačítka k načtení dat. Ve variantě dávkového zpracování dat si uživatel vybere, jaký typ souborů chce z vybrané složky načíst (viz Obrázek 19). Ze složky jsou potom načteny všechny soubory daného typu obsažené ve složce.



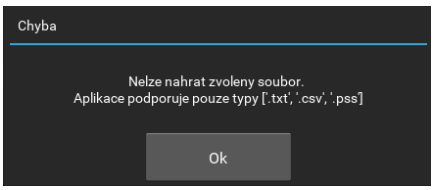
Obrázek 19 - Správce souborů a výběr typu souboru

V obou případech potvrdí uživatel svůj výběr stiskem tlačítka „Nahrát vybrané“, a program načte soubory k dalšímu zpracování.Pokud není při potvrzení načtení dat vybrán žádný konkrétní soubor, nebo složka neobsahuje zvolený typ souboru, uživatel je o tom informován chybovou hláškou (viz Obrázek 20).



Obrázek 20 - Chybová hláška, pokud není vybrán žádný soubor

V případě, že uživatel vybere jediný soubor, který však není svým typem aplikací podporován, opět je zamezeno nahrání dat informací o chybě (viz Obrázek 21).



Obrázek 21 - Chybová hláška, pokud vybraný soubor není povoleného typu

Po validním výběru datových souborů aplikace načítá vybraný soubor nebo první soubor vybrané dávky. Zatím není nutné načítat všechny soubory do paměti, čímž se uspoří paměť a doba potřebná k načítání dat. Software si prozatím pouze uloží do paměti seznam souborů a adresářovou cestu, aby je v dalších krocích mohl načíst ke zpracování.

```
1  with open(datasheets[0], newline='') as csvfile:
2      output, maxvalue = csvreader.rawdata(csvfile)
3      linecount = 0
4      for row in output:
5          if linecount <= 50:
6              outputtext = ",".join(row)
7              self.loadfile.insert_text(outputtext, False)
8              self.loadfile.insert_text("\n", False)
9              self.numlines.insert_text(str(linecount+1)+"\n", False)
10             self.savesets_1.disabled = False
11             linecount += 1
12         else:
13             break
14 self.load_w.dismiss()
15 self.loadfile.readonly = True
16 self.numlines.readonly = True
17 self.importstart.readonly = False
18 self.headerline.readonly = False
19 self.profile.disabled = False
20 if self.loadfile.width < self.loadfile._lines_labels[0].width + 30:
21     self.loadfile.width = self.loadfile._lines_labels[0].width + 30
22 csvfile.close()
```

Vlastní načtení souboru do náhledu aplikace – DAVM.py

V uvedené části kódu je také nastaveno, že se do náhledu načítá pouze prvních 50 řádků souboru. Je to dostatečný počet řádků pro určení struktury zapsaných dat, i určení první řádky počátku importu dat. Opět je to učiněno z důvodu rychlejšího chodu aplikace.

V řádcích 14-21 tohoto kódu lze sledovat, jak řídicí kód jednoduše přistupuje ke grafickým prvkům aplikace a mění jejich parametry v závislosti na načítaných datech a zpřístupňuje ty, které budou používány v dalším kroku.

4.4.3 Nastavení parametrů struktury dat

V této části běhu aplikace musí uživatel dle náhledu načtených dat nastavit parametry struktury dat, díky kterým může software data správně identifikovat a připravit ke zpracování.

Parametry, které je nutné nastavit, jsou opět vydefinovány v základních požadavcích na vyvíjený software v kapitole 4.1.1. K nastavení parametrů souboru slouží příslušné ovládací prvky okna, které jsou svým typem zvoleny s ohledem na požadovaný vstup dat od uživatele.

Nastavení oddělovačů (pole, desetinných míst, ev. tisíců) jsou provedena formou rozbalovacích menu (*Spinner*), které nabízí volbu nejčastějších typů používaných oddělovačů [6]. Pouze v případě oddělovače pole (sloupce dat), je teoreticky možné, že jím může být jakýkoliv jeden znak. Pokud tedy uživatel nenalezne příslušný znak v nabídce rozbalovacího pole, může si zvolit možnost „jiné“ a v tom okamžiku dojde ke zpřístupnění textového pole, kam je možné zadat oddělovací znak ručně.

Uživatel může do ručně zadávaného pole zapsat pouze jediný znak, to je ošetřeno následujícím kódem.

```
1  def divider_t_filter(self):
2      if len(self.divider_t.text) > 1:
3          self.divider_t.text = self.divider_t.text[0]
```

Filtr textového pole pro zadávání oddělovače pole – DAVM.py

Dále je nutné zadat číslo prvního řádku, ve kterém se již nachází číselná data, která se budou dále zpracovávat. Shodně je zadáváno číslo řádku, ve kterém se nachází datová hlavička neboli popisky sloupců dat. V případě, že datový soubor hlavičku sloupců neobsahuje, zaškrtne se políčko (*CheckBox*), které dává informaci o neexistující datové hlavičce [6].

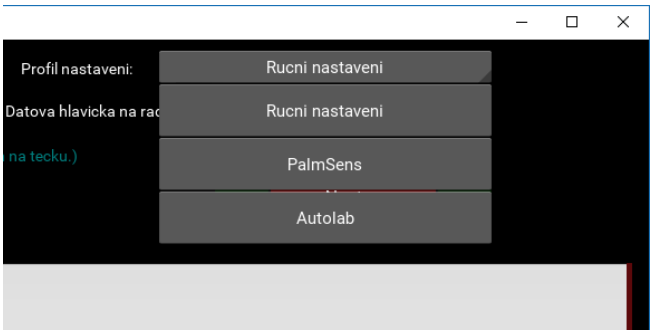
U obou číselných polí je opět hlídána validita zadávaných dat. Do pole lze zadat pouze číselný údaj, který nemůže být vyšší, než je počet řádků souboru. Současně se hlídají obě pole navzájem, tedy číslo řádku s hlavičkou dat nemůže být logicky vyšší než číslo řádky, na které již začínají číselná data.

```
1  def importstart_filter(self, value):
2      try:
3          if int(value) >= maxvalue-1:
4              self.error_popup("Hodnota nemůže být vyšší, než je počet
   řádků souboru!")
5              self.importstart.text = '0'
```

```
6               print("Překročeno. Maximální hodnota je: "+str(maxvalue-1))
7           if int(value) <= int(self.headerline.text):
8               self.headerline.text = ''
9       except ValueError:
10          pass
11
12  def headerline_filter(self, value):
13      try:
14          if self.importstart.text == "":
15              datavalue = 0
16          else:
17              datavalue = int(self.importstart.text)
18          if int(value) >= datavalue or int(value) <=0:
19              self.error_popup("Hodnota nemůže být vyšší, než je počátek
    importu dat \n"
20                   " a nesmí být 0 nebo záporné číslo! \n"
21                   "Zkontroluj nebo nastav počátek importu dat či hlavičku.")
22                      self.headerline.text = ''
23          except ValueError:
24              pass
```

Hlídání validity zadávaných hodnot polí pro import dat – DAVM.py

Je pravděpodobné, že uživatel, který bude s aplikací pracovat pravidelně, bude často využívat stejné nastavení parametrů datové struktury. To zejména z důvodu, pokud bude vyhodnocovat sady dat měřených na shodných typech měřicích přístrojů. Protože není uživatelsky příjemné zadávat všechny zmíněné parametry při každém novém spuštění aplikace, je do programu zavedena funkce „Profil nastavení“.



Obrázek 22 - Nastavení parametrů pomocí profilů

Profil nastavení parametrů lze vybrat na úvodní obrazovce pomocí rozbalovacího menu (Obrázek 22). Výběrem profilu dojde ke vložení uložených hodnot do všech parametrizačních polí. Uživatel následně může konfiguraci pouze vizuálně překontrolovat nebo případně některé parametry ručně upravit.

Uložení nového profilu do aplikace nebo úprava stávajících profilů je možná editací konfiguračního souboru *config.cfg*.

```
1  # Profily CSV
2  PROFILES HINT:
```

```
3  profile_name column_divider first_data_line header_line decimal_point
   thousands_divider is_header?
4
5  <CSVPROFILES>
6  PalmSens mezera 3 '' . neni True
7  Autolab ; 2 1 , neni False
8  </CSVPROFILES>
```

Forma uložení profilů nastavení v souboru config.cfg

V konfiguračním souboru aplikace je uvedena posloupnost zadávaných parametrů. Z důvodu, že oddělovače můžou být jakýmkoliv znakem, jsou jednotlivé parametry odděleny mezerou a pokud je oddělovačem právě mezera, tak se parametr zapíše slovně, jak je možné vidět v uvedeném příkladu.

V současné době jsou jako přednastavené konfigurace uvedeny volby „PalmSens“ pro datový formát souborů „.pss“ ze softwaru PSTrace firmy PalmSens (zařízení *PalmSens4*) a „Autolab“ pro soubory exportované ze softwaru NOVA 2.0 firmy Metrohm (zařízení *Metrohm Multi Autolab/M204*).

Po nastavení všech parametrů struktury dat se pokračuje tlačítkem „Nastavit“. Stiskem tohoto tlačítka dojde k automatické kontrole nastavených parametrů. Pokud nějaký povinný parametr chybí, uživatel je upozorněn chybovou hláškou a vyzván k nápravě. Stejným způsobem je provedena kontrola, zda nejsou některá nastavení v konfliktu mezi sebou. Příkladem kolize je nastavení oddělovače pole a oddělovače desetinných míst shodným znakem. Taková struktura dat není platná a aplikace by nebyla schopna správně identifikovat data.

```
1  def save_sets_1(self):
2      if self.importstart.text == "" or int(self.importstart.text) <= 0:
3          self.error_popup("Chybí nastavení počátku importu dat!")
4      else:
5          if self.headerline.text == "" and self.cb_no_header.active is
   False:
6              self.error_popup("Chybí nastavení datové hlavičky!")
7          else:
8              t_A = self.divider_s.text
9              t_B = self.divider_t.text
10             t_C = self.thousands.text
11             t_D = self.point.text
12             if t_A in [t_C, t_D] or t_B in [t_C, t_D] or t_C in [t_A,
   t_B, t_D]:
13                 self.error_popup("Oddělovače nemohou být stejné!")
```

Kontrola nastavení parametrů na chybějící nebo kolizní data – DAVM.py

Pokud jsou všechna nastavení parametrů správná a úplná, je možné pokračovat do dalšího okna nastavení dat.

4.5 Zpracování dat ke vstupu do analýzy

V předchozí kapitole byl popsán způsob načtení souborů s naměřenými daty a nastavení parametrů struktury dat. Přechodem do dalšího okna je automaticky spuštěna funkce, která provede znovunačtení souboru, avšak nyní se použijí nastavené parametry struktury. Program je nyní schopen datový soubor rozdělit do jednotlivých datových sloupců, oddělit vlastní data a hlavičku dat. Pokud datový soubor nemá hlavičku dat, jsou sloupcům přiřazeny písmena abecedy, aby mohly být správně identifikovány uživatelem.

```
1  with open(datasheets[0], newline='') as file:
2      global csv_params
3      csv_params = [csv_divider, csv_importstart, csv_headerline,
   csv_point, csv_thousands, cb_h]
4      headers, values = csvreader.takedata(file, *csv_params)
5      file.close()
```

Načtení dat s nastavenými parametry – DAVM.py

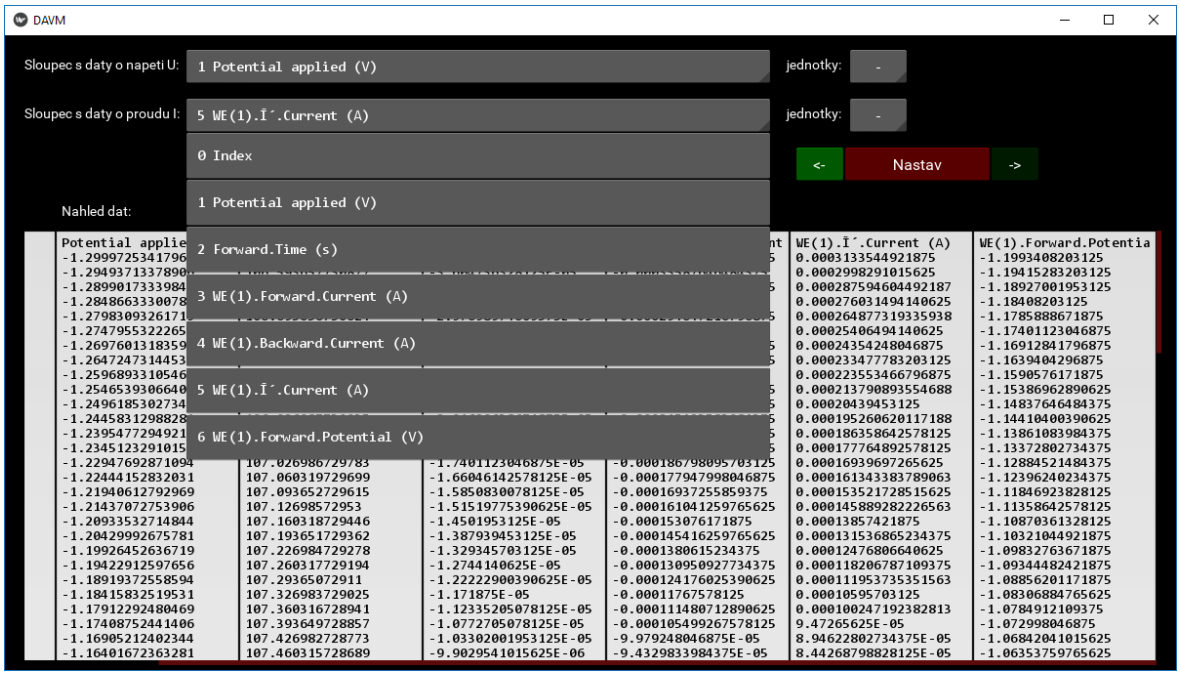
Vlastní načtení dat s použitými parametry provádí funkce *takedata* ze skriptu *csvreader.py*. Stejně jako v předchozím okně platí, že se k zobrazení načítá pouze prvních 100 řádek jednoho souboru i v případě dávkového zpracování dat. V této sekci se nastavují fyzikální významy naměřených dat před vstupem do matematické analýzy. Stále platí nutný předpoklad, že struktura dat celé dávky souborů je shodná.

Data jsou načtena do vizuálně oddělených sloupců. To slouží uživateli pro lepší orientaci v datech k dalšímu nastavení a současně pro kontrolu, že uživatel nastavil parametry struktury dat správně. Pokud by došlo např. k nastavení špatného oddělovače polí a data by tak zůstala nerozdělena pouze v jednom sloupci, je možné se navigačním tlačítkem vrátit do předchozího okna a parametr jednoduše změnit na správný.

```
1  cls = []
2  for i in range(len(values[0])):
3  cls.append(TextInput(text='', multiline=False, font_name="consola.ttf",
   show_errors=False, is_focusable=False, font_size=14))
4      root_manager.csvscreen.csvbox.add_widget(cls[i])
```

Vytvoření datových sloupců pro načtení dat – DAVM.py

V ukázce kódu viz výše lze poukázat na definování nových grafických prvků aplikace přímo v kódu jazyka Python. Jedná se o textová pole (*TextInput*), která jsou vytvořena v počtu sloupců dat v datovém souboru [6]. Tyto prvky nemohly být vytvořeny již při zavádění programu v souboru DAVM.kv, protože nelze předem určit, kolik textových polí bude zapotřebí pro zobrazení všech sloupců dat.



Obrázek 23 - Okno aplikace s načtenými daty

Jak je zřejmé z výše uvedeného obrázku (Obrázek 23), před vstupem naměřených dat do analýzy je nutné provést výběr sloupců, které nesou správné údaje. Opět na základě zkušenosti uživatele s naměřenými daty se vybere sloupec, který obsahuje data o potenciálu přikládaném na pracovní elektrodu. Následně se vybere sloupec s daty o proudu, který prochází pracovní elektrodou v závislosti na tomto potenciálu a v závislosti na chemických dějích vyvolaných obsahem analytu v roztoku. Princip voltametrických měření je detailně popsán v kapitole 2 a také v [1] a [2].

Po provedeném výběru sloupců je možné ještě přiřadit zvoleným sloupcům správné fyzikální jednotky. Toto nastavení je volitelné a je určeno zejména pro případy, kdy v datové hlavičce nejsou jednotky uvedeny, případně pokud datová hlavička úplně chybí. Je nutné zdůraznit, že přiřazením jednotek nedochází k žádnému přepočítání ani ovlivnění načtených dat. Toto nastavení slouží pouze pro zobrazení jednotek na osách grafů a jejich správné nastavení náleží pouze uživateli.

Uživatel dále opět pokračuje stiskem tlačítka „Nastavit“. Tím je spuštěna funkce načtení vybraných dat ze všech souborů vybraných pro dávkové zpracování. Zde je již vhodné načítat data ze všech souborů. Do paměti programu jsou načteny vždy pouze vybrané sloupce dat z každého souboru, takže nedochází k ukládání nadbytečných dat do paměti. Současně je vhodné načíst data ještě před vstupem do analýzy, která je na výpočtový čas nejnáročnější z celého běhu aplikace.

```
1  delit = []
2  for v in range(len(i_values)):
3      try:
4          float(u_values[v])
5          float(i_values[v])
6      except:
7          delit.append(int(v))
8          continue
9      else:
10         u_values[v] = float(u_values[v])
11         i_values[v] = float(i_values[v])
12
13 for d in sorted(delit, reverse=True):
14     del u_values[d]
15     del i_values[d]
16 all_u_vals.append(u_values)
17 all_i_vals.append(i_values)
```

Kontrola dat na číselné hodnoty před vstupem do analýzy – DAVM.py

Při načítání dat probíhá ještě kontrola, zda jsou všechna data číselné hodnoty. Pokud jsou z nějakého důvodu data chybná, je odstraněn celý řádek dat, tedy údaj o potenciálu i proudu. Aplikace se tak vyhne pádu nebo chybě při analýze dat, ovšem může i tak dojít ke zkreslení výsledků analýzy.

Po načtení dat je ještě okno aplikace znovu překresleno a jsou zobrazeny pouze vybrané sloupce dat pro poslední vizuální kontrolu uživatelem. V případě chybného výběru se lze opět navigačním tlačítkem vrátit zpět a opakovat akci. Pokud jsou data vybrána správně, je možné opět navigačním tlačítkem pokračovat do výpočtu analýzy a k zobrazení výsledků.

4.6 Matematická analýza dat

Připravená data vstupují do algoritmu matematické analýzy, která je definována ve zvláštním skriptu *analysis.py.* Do algoritmu je načítán jeden soubor po druhém, voláním funkce *data_analysis* příkazem z řídicího skriptu aplikace. Celý proces výpočtu analýzy běží na pozadí aplikace a dílčí výsledky jsou ukládány do paměti programu.

Matematická analýza dat vychází z metody filtrované první a druhé derivace, popsané v kapitole 3.3. Jedná se o rozšíření metody filtrované první derivace z kapitoly 3.2, a je vyhodnocena jako přesnější a komplexnější pro účely vyhodnocení naměřených dat. Pro účely vyhodnocení základny a výšky peaků je potom použita jednoznačně definovaná metoda z kapitoly 3.4.

4.6.1 Řídicí funkce algoritmu analýzy

Skript *analysis.py* obsahuje několik funkcí, které jsou postupně volány řídicí funkcí *data_analysis*. Řídicí funkce má na starosti postupné předávání mezivýsledků analýzy do

dalších dílčích funkcí, případně opakování provádění funkce při testování vlivu na změnu vstupních parametrů. Výstupem řídicí funkce jsou konečné výsledky analýzy pro daný soubor dat.

```
1  def data_analysis(u_data, i_data, dataset, d_win, peak_filter):
2      results = []
3      extrems = 1000
4      peaks = []
5      minims = []
6      print("Provádění analýzy souboru dat", dataset, "...")
7      while True:
8          a_peaks, a_minims = do_analysis(u_data, i_data, d_win)
9          if d_win == 1 and (len(a_peaks)+len(a_minims)) == 0:
10             print("Extrémy nelze analyzovat\n")
11             break
12         else:
13             if extrems > (len(a_peaks)+len(a_minims)):
14                 extrems = (len(a_peaks)+len(a_minims))
15                 print("Testování... šířka okna:", d_win, "- počet extrémů
   nalezeno: ", extrems)
16                 peaks = a_peaks
17                 minims = a_minims
18                 d_win += 1
```

Ukázka části řídicí funkce matematické analýzy dat – analysis.py

Na vstupu do funkce *data_analysis* je definováno několik parametrů funkce, které mají následující význam:

- *u_data* – datová řada hodnot přikládaného potenciálu
- *i_data* – datová řada hodnot závislého proudu
- *dataset* – název souboru dat
- *d_win* – konstanta základní šířky okna (+/-) pro výpočet náhrady derivací
- *peak_filter* – konstanta pro filtrování analyzovaných peaků

Uvedené konstanty *d_win* a *peak_filter* jsou do algoritmu načteny z konfiguračního souboru *config.cfg* (viz kapitola 4.3.3) a jejich význam je podrobněji popsán dále.

4.6.2 Určení lokálních minim a maxim analyzovaných dat

Data jsou nejdříve nahrána do první části analýzy – funkce *do_analysis.* Jedná se o aplikovanou metodu filtrované první a druhé derivace. Výsledkem analýzy je identifikace lokálních maxim (pro určení peaků) a lokálních minim (pro určení základen peaků).

```
1  # Šířka okna
2  k = d_win
3  j = -k
4
5  # Výpočet koeficientů b_j, c_j
```

```
6  sum_kj = 0
7  sum_kj2 = 0
8  rng_kj = list(range(j, k + 1))
9  for n in rng_kj:
10     sum_kj = sum_kj + n
11     sum_kj2 = sum_kj2 + n * n
12 avg_kj = sum_kj / len(rng_kj)
13 avg_kj2 = sum_kj2 / len(rng_kj)
14
15 sq_kj = 0
16 sq_kj2 = 0
17 for n in rng_kj:
18     sq_kj = sq_kj + (n * n - avg_kj)
19     sq_kj2 = sq_kj2 + (n * n - avg_kj2) * (n * n - avg_kj2)
20
21 b_j = []
22 c_j = []
23 for n in rng_kj:
24     b_jn = (n - avg_kj) / sq_kj
25     b_j.append(b_jn)
26     c_jn = (n * n - avg_kj2) / sq_kj2
27     c_j.append(c_jn)
```

Definice a výpočet koeficientů pro náhrady derivací – analysis.py

V první fázi analýzy jsou předem vypočítány koeficienty *b_j* a *c_j,* jejichž význam je popsán v kapitole 3.3, konkrétně se jedná o vzorce (3.14) a (3.15). Do výpočtu je vnesena konstanta *d_win*, tedy šířka okna. Šířka okna určuje počet hodnot v okolí zkoumaného bodu. Nastavení počáteční hodnoty konstanty *d_win* je experimentální, na základě zkušenosti uživatele.

```
1  # Výpočet náhrady první (a1) a druhé (a2) derivace
2  a1 = []
3  a2 = []
4  peaks = []
5  minims = []
6
7  for n in range(len(u_data)):
8      if n < k or n >= (len(u_data)-(k+1)):
9          n += 1
10         a1.append("")
11         a2.append("")
12         continue
13     else:
14         a1s = 0
15         a2s = 0
16         for m in range(len(rng_kj)):
17             a1m = i_data[n+rng_kj[m]] * b_j[m]
18             a1s = a1s + a1m
19             a2m = i_data[n+rng_kj[m]] * c_j[m]
20             a2s = a2s + a2m
21         a1.append(a1s)
22         a2.append(a2s)
23         n += 1
```

Výpočet náhrady první a druhé derivace – analysis.py

Na základě vypočítaných koeficientů *b_j* a *c_j* jsou v druhé části algoritmu vypočítány průběhy náhrady první a druhé derivace, jak je uvedeno v kódu výše. Použijí se postupy (3.16) a (3.17).

```
# Indikace maxim (peaků) a minim (pro určení základen)
for v in range(len(a1)):
    if v == 0:
        v += 1
        continue
    else:
        try:
            float(a1[v-1])
            float(a1[v])
        except:
            continue
        else:
            tst1 = copysign(1, a1[v - 1])
            tst2 = copysign(1, a1[v])
            if tst1 != tst2 and a2[v] < 0:
                if i_data[v + 1] > i_data[v]:
                    peaks.append([(v + 1), u_data[v + 1], i_data[v + 1]])
                elif i_data[v - 1] > i_data[v]:
                    peaks.append([(v - 1), u_data[v - 1], i_data[v - 1]])
                else:
                    peaks.append([v, u_data[v], i_data[v]])
            elif tst1 != tst2 and a2[v] > 0:
                minims.append([v, u_data[v], i_data[v]])

return peaks, minims
```

Rozhodovací část algoritmu identifikující lokální maxima a minima – analysis.py

V poslední části algoritmu jsou pak nalezena všechna lokální minima a maxima zkoumaného průběhu, jak je popsáno v závěru kapitoly 3.3. Výsledkem analýzy jsou všechny nalezené lokální extrémy, které jsou určeny hodnotou potenciálu a proudu, při kterých nastávají. K výsledkům je přidán identifikátor polohy údajů mezi ostatními daty. Tento identifikátor slouží ke snadnému vyhledání těchto údajů v datech a je určen pro další použití.

Následně je v řídicí funkci provedeno vyhodnocení, zda je počet nalezených extrémů konečný. V případě, že ne, celá analýza je provedena znovu pro hodnotu okna *d_win* zvýšeného o 1. Pokud řídicí algoritmus rozhodne, že je počet nalezených extrémů konečný, jsou výsledky analýzy použity dále.

4.6.3 Testování plného určení peaků

V případě, kdy předchozí analýza skončila výsledkem konečného množství a učení hodnot lokálních extrémů, jsou tyto výsledky předány funkci *peak_basetest*. Tato funkce má za úkol zjistit, zda jsou nalezené peaky (lokální maxima) plně definované. To znamená, že

podmínkou existence peaku je, že k nalezenému maximu existuje také nalezené minimum a to zleva i zprava.

```
1  # Definice základny peaků
2  def peak_basetest(peaks, minims):
3      delpeak = []
4      for peak in peaks:
5          x_low = ''
6          x_high = ''
7          for v_min in minims:    # Nalezení nejbližších minim
8              if v_min[1] < peak[1]:
9                  x_low = v_min[0]
10             if v_min[1] > peak[1]:
11                 x_high = v_min[0]
12                 break
13         peakid = peaks.index(peak)
14         if x_high == '' or x_low == '':
15             delpeak.append(peakid) # Peak určen k odstranění, nemá
   všechny minima
16         else:
17             peaks[peakid].append(x_low)
18             peaks[peakid].append(x_high)
19     for d in sorted(delpeak, reverse=True):
20         del peaks[d]
21     return peaks
```

Testování plného určení peaků – analysis.py

V případě že nalezené maximum (případný peak) nemá identifikované nejbližší extrémy zleva i zprava jako lokální minima, potom není možné určit základnu takového peaku a tento peak je z výsledků další analýzy vypuštěn. Znamená to totiž, že peak není naměřen celý, a tudíž by jeho výška byla nejspíše určena chybně. V případě, že uživatele jeho hodnota zajímá, je nutné měření opakovat s širším oknem měřicího potenciálu tak, aby obsáhl i počátek sledovaného peaku. Výstupem testu jsou tedy platné peaky s identifikátory na jejich nejbližší minima zleva a zprava.

4.6.4 Určení základny a výšky peaku

Po předchozích částech algoritmu jsou známy všechny peaky, které je možné plně definovat. To znamená, že je možné určit jejich základnu ze známých nejbližších minim a následně ze známé základny lze určit i výšku peaku.

Výpočet základny peaku je definován v kapitole 3.4. Tento výpočet je implementován do funkce *get_peak_base.* Do funkce vstupují data z předchozí analýzy a originální data o naměřených datech potenciálu a proudu. Pomocí indexů dat z analýzy jsou načteny počáteční hodnoty z naměřených dat.

```
# identifikace okrajových bodů úsečky
x_low = u_data[lo_id]
y_low = i_data[lo_id]
x_high = u_data[hi_id]
y_high = i_data[hi_id]
```

Načtení počátečních souřadnic minima zleva a zprava

Následně je spočtena základní rovnice úsečky spojující okolní minima peaku a pomocné koeficienty pro další analýzu.

```
# definice rovnice úsečky základny y(t) = ax(t) + b
a = (y_high - y_low) / (x_high - x_low)
b = y_high - a * x_high
yt_high = a * x_high + b
yt_low = a * x_low + b
xc = (x_high + x_low) / 2  # střed x-ových souřadnic úsečky
k = 0  # počet bodů pod úsečkou - poč. hodnota
xp = 0  # průměr x-ových souřadnic bodů pod úsečkou - poč. hodnota
```

Výpočet úsečky spojující okolní minima a pomocné koeficienty – analysis.py

Když je určena výchozí úsečka, algoritmus hledá body ležící pod úsečkou, tedy směrem od počátečních bodů úsečky k vrcholu peaku. Pokud nejsou žádné takové body nalezeny, výchozí úsečka je vyhodnocena jako platná základna peaku. V opačném případě dojde k odebrání jednoho bodu zprava nebo zleva. Určení strany odebrání (resp. i přidání) bodu je stanoveno na základě rozhodnutí, zda je vypočítaný průměr x-ových souřadnic menší než střed x-ových souřadnic úsečky (přidá 1 hodnotu zleva) nebo větší (odebere jednu hodnotu zprava).

Poté se cyklus vrací na začátek, spočte se nová hodnota úsečky a opět se zkoumají body ležící pod úsečkou. Ve chvíli, kdy už takové body nejsou, je poslední nově vypočítaná úsečka stanovena jako základna daného peaku. Pokud taková úsečka nalezena není, jedná se opět o neidentifikovatelný peak, který je z výsledků analýzy odstraněn. To znamená, že proces hledání základny peaku je ukončen ve chvíli, kdy jsou body odebírány/přidávány tak dlouho, až se dosáhne vrcholu peaku.

```
for n in range(hi_id-lo_id-1):
    yt_tst = a * u_data[lo_id+n+1] + b
    if yt_tst > i_data[lo_id+n+1]:
        k += 1
        xp = xp + u_data[lo_id+n+1]
if k != 0:
    xp = xp / k
    if xp > xc:
        hi_id = hi_id - 1
        if u_data[hi_id] <= peak_x:
            delit = True
            break
        print("peak č. ", m+1, ": odebral 1 hodnotu zprava")
```

```
14      elif xp <= xc:
15          lo_id = lo_id + 1
16          if u_data[lo_id] >= peak_x:
17              delit = True
18              break
19          print("peak č. ", m+1, ": přidal 1 hodnotu zleva")
20  else:
21      if delit:
22          break
23      else:
24          peak_height = peak_y - a * peak_x - b
25          print("Peak č. ", m+1, "analyzován - OK\n")
26          results.append([peak_id, peak_x, peak_y, peak_height, x_low,
    y_low, x_high, y_high])
27          break
```

Testování bodů ležících pod úsečkou – analysis.py

Ve chvíli, kdy jsou stanoveny základny všech peaků nebo jsou chybné peaky odstraněny, proces končí výstupem parametrů plně definovaných peaků, jak je zřejmé z řádky č. 26 výše uvedeného kódu.

4.6.5 Filtrace šumových peaků

Do algoritmu matematické analýzy je na závěr přidána ještě experimentální funkce *filter_peaks.* Porovnáváním výsledků analýzy bylo zjištěno, že jsou často detekovány peaky, které jsou svou výškou zanedbatelné v porovnání se „zájmovými“ peaky. Jedná se většinou o průběh proudu vyvolaný nějakým typem šumu. Tato funkce umožňuje, aby byly některé šumové peaky z výsledků analýzy vyloučeny.

Funkce funguje na jednoduchém principu. Výsledky analýzy jsou vloženy do funkce, která porovnává jednotlivé peaky mezi sebou na základě jejich výšky. Největší analyzovaný peak datové řady je označen jako referenční. Následně platí, že pokud je některý z peaků menší než je poměrová velikost k referenčnímu peaku, je tento peak z výsledků analýzy odstraněn. Poměrová velikost je stanovena na základě konstanty *peak_filter*. Konstanta může teoreticky nabývat hodnot v rozmezí 0 až 1 (resp. 0 - 100 % ref. velikosti).

```
1   def filter_peaks(results, peak_filter):
2       delit = []
3       if peak_filter <= 0:
4           pass
5       else:
6           peaks = []
7           for result in results:
8               peaks.append(result[3])
9           if peaks == []:
10              pass
11          else:
12              ref_peak = max(peaks)
13              for n in range(len(peaks)):
```

```
14                  if peaks[n] < peak_filter*ref_peak:
15                      delit.append(n)
16              for d in sorted(delit, reverse=True):
17                  print("Filtrace - smazán peak č. ", d + 1)
18                  del results[d]
19      return results
```

Poměrová filtrace peaků – analysis.py

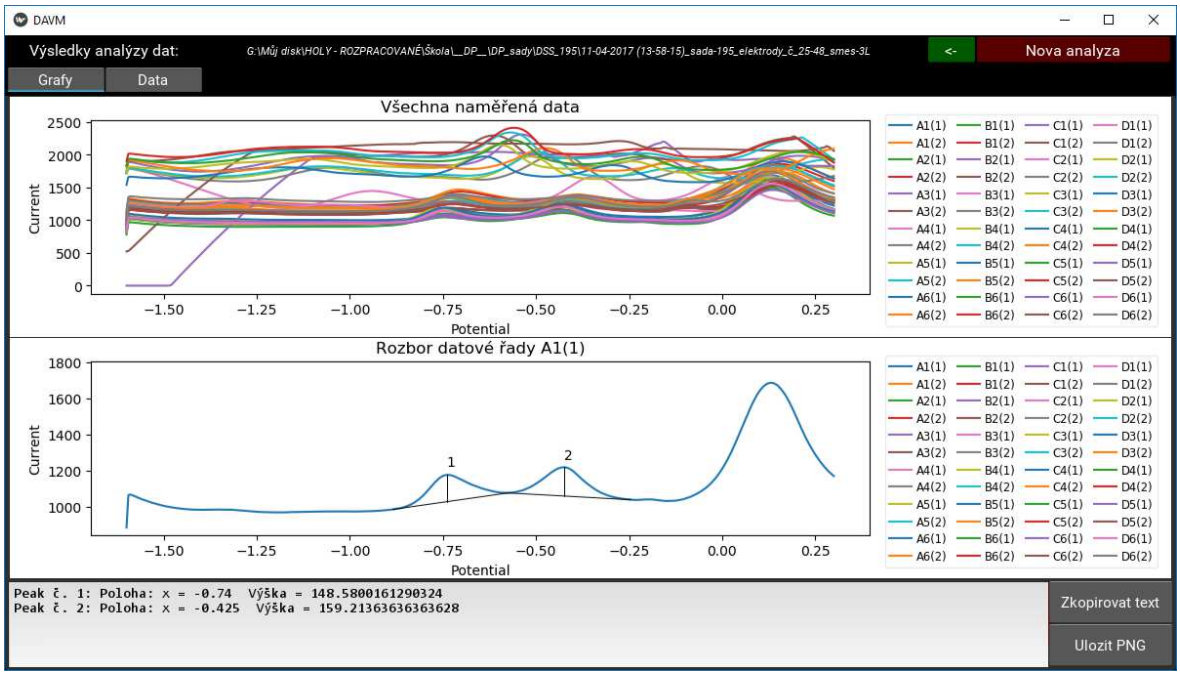
Protože se jedná o experimentální funkci, je nutné ji zavést do algoritmu tak, aby mohla být upravována nebo deaktivována bez zásahu do zdrojového kódu aplikace. Toto je jednoduše provedeno přidáním parametru (konstanty) *filter_peaks* do konfiguračního souboru *config.cfg*, odkud funkce jeho hodnotu načítá. Pokud uživatel nechce, aby byl filtr peaků aplikován, postačí konstantě v konfiguračním souboru přiřadit nulovou hodnotu.

4.7 Zobrazení výsledků analýzy a export výstupních dat

Provedením analýzy dle předchozí kapitoly jsou dostupná naměřená data a výsledky identifikace peaků. V aplikaci je načteno poslední okno, které má za úkol předat uživateli v grafické i textové podobě naměřená data se zobrazením výsledků analýzy. Základní požadavky na výstup dat jsou popsány v kapitole 4.1.3. Vizualizace dat je provedena prostřednictvím základních funkčních prvků knihovny Kivy pro zobrazení dat v numerické a textové podobě a pomocí knihovny Matplotlib je provedeno zobrazení naměřených dat a výsledků analýzy formou grafů.

Aby byla zajištěna přehlednost zobrazovaných dat zejména s ohledem na větší množství dávkově zpracovávaných souborů dat, je grafická prezentace výsledků provedena ve dvou grafech. První graf slouží k souhrnnému zobrazení všech načtených souborů naměřených dat. Uživatel tak získá přehledné porovnání datových řad mezi sebou s možností určit, které datové řady se vizuálně vymykají svými hodnotami.

Na obrázku dále (Obrázek 24) je zobrazena ukázka zobrazení výsledků naměřených dat a jejich analýzy. V prvním grafu shora jsou zobrazeny všechny řady naměřených dat. Pokud se některá data značně vymykají ostatním, může to vést ke zkreslení měřítka grafu, a tím ke špatnému zobrazení validních datových řad. Z tohoto důvodu je do aplikace zanesena funkcionalita vypínání datových řad, které jsou uživatelem vyhodnoceny jako nevalidní.



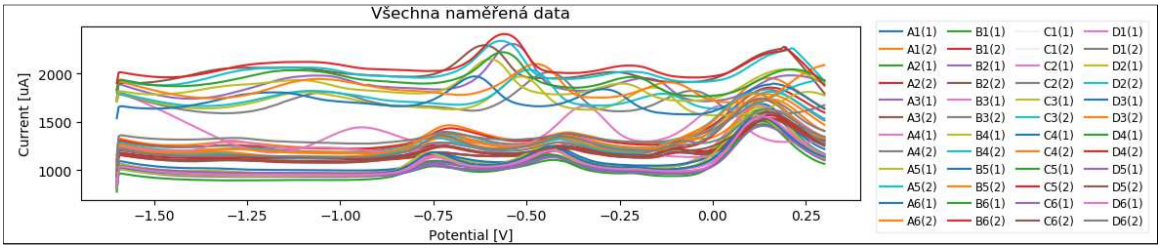
Obrázek 24 - Vizualizace výsledků analýzy

Matplotlib umožňuje širokou škálu možností práce s grafy, vč. interaktivity, a potvrdila se tak správnost výběru grafické knihovny. V souboru *DAVM.py* jsou nadefinovány funkce typu událost (*event*) [7]. Tyto funkce fungují tak, že po kliknutí uživatele ukazatelem myši na příslušnou datovou řadu nebo její barevný identifikátor v legendě, dojde k vypnutí příslušné datové řady a současně je tato řada vynechána při výpisu a zobrazení výsledků.

```
1  def ax_rescale_y():
2      ymaxs = []
3      ymins = []
4      for li in range(len(pltset)):
5          if pltset[li] not in hiddenp:
6              ymaxs.append(max(pltset[li].get_ydata()))
7              ymins.append(min(pltset[li].get_ydata()))
8      ymax = max(ymaxs)
9      ymin = min(ymins)
10     yrng = ymax - ymin
11     ax.set_ylim(top=ymax+yrng*0.05)
12     ax.set_ylim(bottom=ymin-yrng*0.05)
```

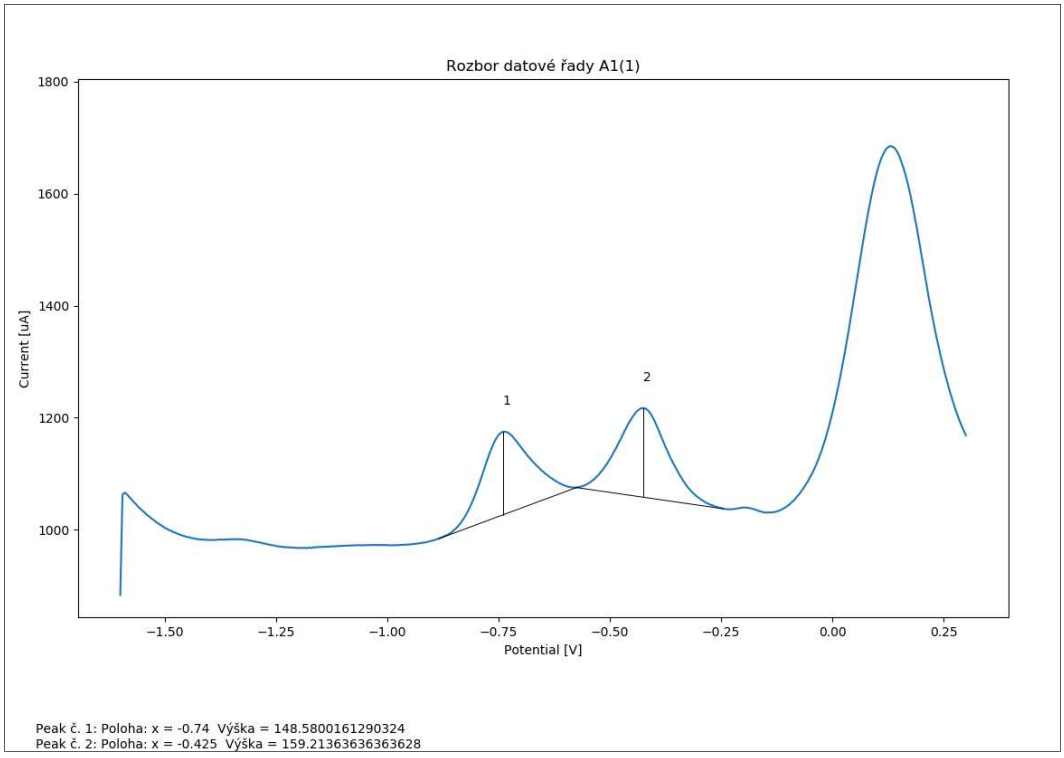
Provedení přepočítání měřítka grafu – DAVM.py

Dále je nadefinována funkce (viz kód), která po skrytí datové řady přepočítá měřítko grafu tak, aby zbylé datové řady byly co nejlépe zobrazeny (viz Obrázek 25). Vypnutou datovou řadu je možné vyvolat zpět kliknutím na její legendu.



Obrázek 25 - Přepočítaný graf po vypnutí datových řad

Spodní graf (viz. Obrázek 24 - Vizualizace výsledků analýzy) potom zobrazuje konkrétní analýzu datové řady, včetně zobrazení vyhodnocených peaků s číselným identifikátorem. Ten odkazuje na textové a numerické výsledky analýzy v textovém poli pod grafem. Mezi analýzami jednotlivých datových řad lze přepínat opět kliknutím na legendu u tohoto grafu. Řady, které byly vypnuty v prvním souhrnném grafu, jsou zde neaktivní a jejich výsledky nelze zobrazit a jsou vynechány. To, že je řada skrytá, pozná uživatel vizuálně tak, že u příslušné popisky v legendě není zobrazen barevný identifikátor.

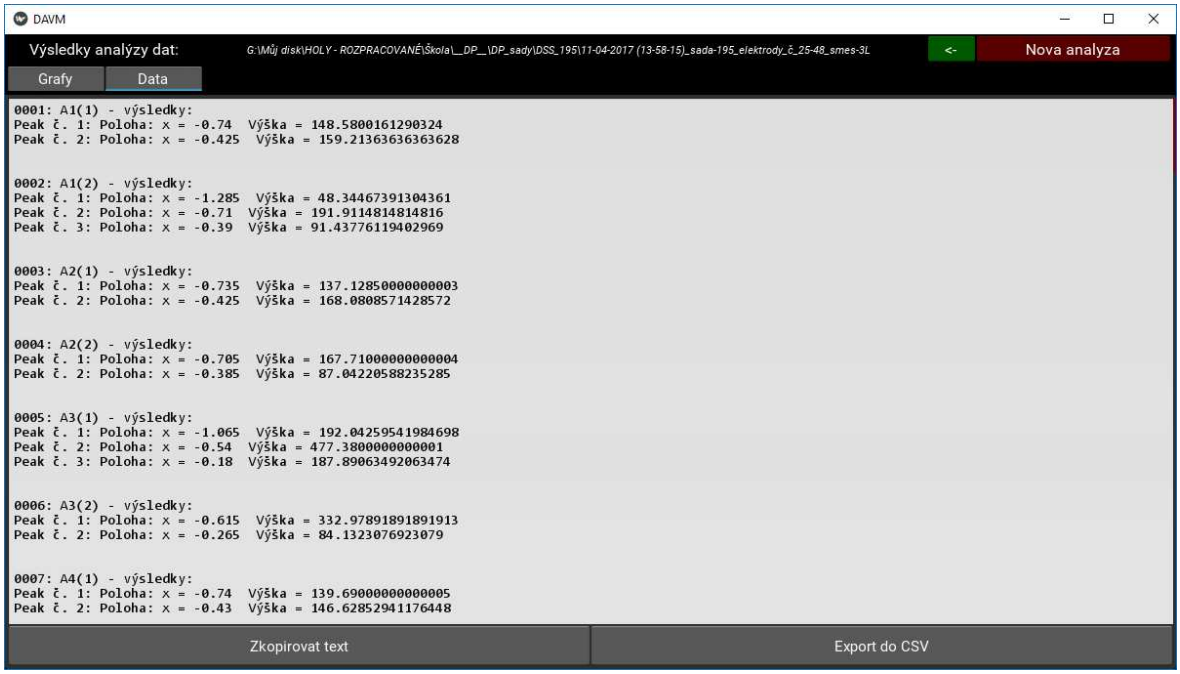


Obrázek 26 - Export analýzy do obrázku

Pod grafy jsou dvě funkční tlačítka pro export právě zobrazených dat. V prvním případě dojde pouze ke zkopírování výsledků z textového pole do schránky. Ve druhém případě je graf analýzy zobrazené datové řady uložen jako obrázek PNG do pracovní složky, ze které byla načtena data (viz Obrázek 26).

Okno výsledků analýzy je dále ještě rozděleno na dvě záložky – „Grafy“ a „Data“. Po

přepnutí na datovou záložku (viz Obrázek 27) se zobrazí výsledky všech analyzovaných řad ve formě textových a numerických údajů. Opět jsou vynechány výsledky vypnutých datových řad.



Obrázek 27 - Zobrazení záložky s daty

Zde je možné data zkopírovat do schránky ve formátu, v jakém jsou zapsány do textového pole v okně. V případě, že chce uživatel s vyhodnocenými daty dále pracovat, nabízí se možnost exportu dat do souboru CSV. Data jsou zapsána do tabulky ve formátu vhodném k dalšímu zpracování (viz Obrázek 28).

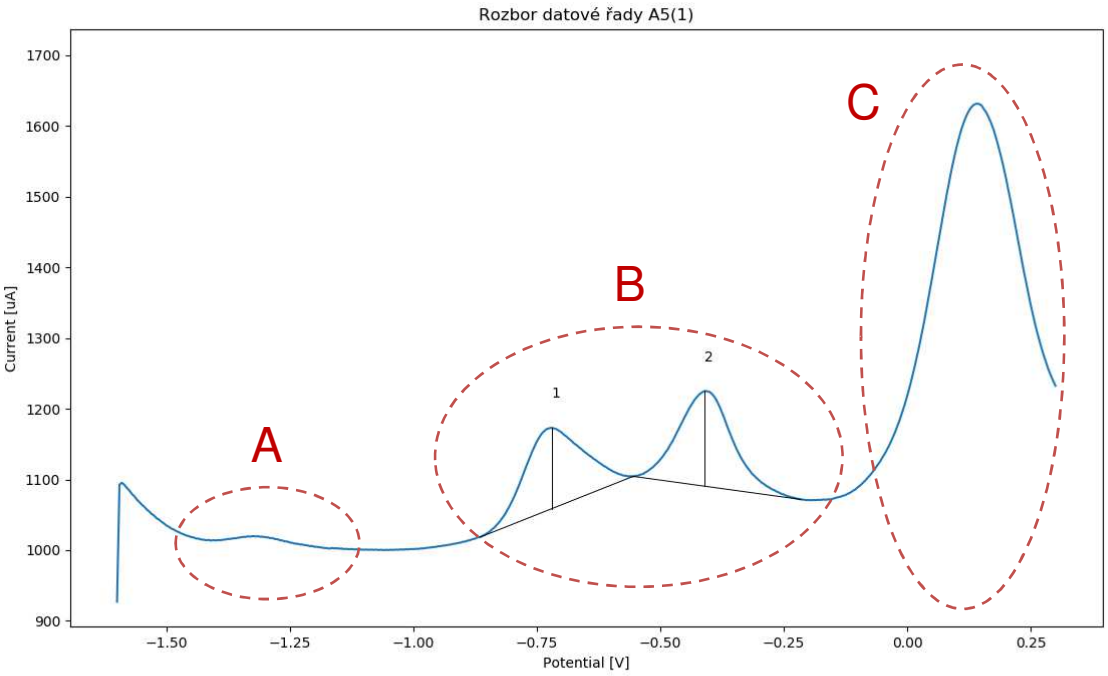
Id	Datafile	Peak_id	Peak_x	Peak_height
1	A1(1)	1	-0,74	148,5800161
1	A1(1)	2	-0,425	159,2136364
2	A1(2)	1	-1,285	48,34467391
2	A1(2)	2	-0,71	191,9114815
2	A1(2)	3	-0,39	91,43776119
3	A2(1)	1	-0,735	137,1285
3	A2(1)	2	-0,425	168,0808571

Obrázek 28 - Ukázka exportu ve formátu CSV (v MS Excel)

4.8 Ověření vyhodnocených dat

Testování vytvořeného programu a ověření výsledků analýz prováděných navrženým analytickým algoritmem bylo prováděno na souborech reálných naměřených dat z měřicího zařízení PalmSens. Jednalo se zejména o testování funkčnosti dávkového zpracování dat při analýze většího počtu datových souborů (např. viz Obrázek 24)

a současně bylo ověřováno, jak si vytvořený algoritmus poradí s určením peaků.



Peak č. 1: Poloha: x = -0.72 Výška = 114.58419354838702
Peak č. 2: Poloha: x = -0.41 Výška = 134.69132352941187

Obrázek 29 - Rozbor ověřování dat vyhodnocených programem DAVM
(A – filtrovaný peak, B – plně určené peaky, C – chybně naměřený peak)

Na obrázku (Obrázek 29) je proveden rozbor jednoho z testovaných vzorků. V sekci grafu označeného „A“ lze vizuálně rozpoznat proudový peak malé výšky v porovnání s ostatními viditelnými peaky na tomto vzorku. V tomto případě byl peak správně vyhodnocen funkcí filtrace peaků (viz kapitola 4.6.5) jako šum a je z výsledků analýzy vynechán. Konstanta *peak_filter* byla experimentálně nastavena na hodnotu 0,2, tedy filtrace peaků menších než je 20 % velikosti největšího rozpoznaného peaku.

V sekci „C“ je peak, který rovněž nebyl vyhodnocen. V tomto případě se jedná o chybně změřený peak. Měření bylo zřejmě ukončeno dříve, než bylo dosaženo pravého minima peaku a algoritmus jej vyhodnotil jako peak, který není plně určen. To potvrzuje správnost jeho funkce dle kapitoly 4.6.3.

Sekce „B“ potom obsahuje plně určené peaky, které prošly všemi kontrolami algoritmu analýzy dat a byla určena jejich poloha a výška. Vizuální kontrola nasvědčuje jejich správnému určení výšky i polohy. V následující tabulce (Tabulka 2) je potvrzeno, že algoritmus určil polohu peaků správně, tedy že na poloze určené programem (viz Obrázek 29), se skutečně nalézají lokální maxima proudu.

Tabulka 2 - Ověření polohy nalezených peaků

Peak č. 1	
Potential [V]	**Current [uA]**
-0,770	1121,250
-0,765	1129,690
-0,760	1138,560
-0,755	1146,310
-0,750	1153,810
-0,745	1159,690
-0,740	1164,880
-0,735	1168,380
-0,730	1171,380
-0,725	1172,250
-0,720	**1173,190**
-0,715	1172,060
-0,710	1171,060
-0,705	1168,750
-0,700	1166,500
-0,695	1163,250
-0,690	1160,750
-0,685	1157,440
-0,680	1154,810
-0,675	1151,130
-0,670	1148,810

Peak č. 2	
Potential [V]	**Current [uA]**
-0,460	1175,130
-0,455	1181,810
-0,450	1189,190
-0,445	1195,560
-0,440	1202,250
-0,435	1207,630
-0,430	1213,250
-0,425	1217,380
-0,420	1221,000
-0,415	1223,380
-0,410	**1225,130**
-0,405	1224,690
-0,400	1223,690
-0,395	1220,630
-0,390	1216,440
-0,385	1210,190
-0,380	1203,190
-0,375	1194,500
-0,370	1185,810
-0,365	1175,880
-0,360	1167,130

Hodnocení a závěr

Výstupem předkládané práce je funkční program pro automatické zpracování dat z voltametrických měření, umožňující dávkové zpracování souborů dat s hromadným zobrazením výsledků analýzy v přehledné formě. K hlavním přínosům patří kromě dávkového zpracování více souborů dat také možnost univerzálního použití napříč naměřenými daty ze softwarů měřicích zařízení více různých výrobců. To je možné díky funkcionalitě uživatelské parametrizace vstupních dat ve formátu prostého textu, do kterého lze data z obslužných programů většiny měřicích systémů ukládat nebo alespoň exportovat. Program je vytvořen v jednoduchém grafickém uživatelském prostředí, které umožňuje uživateli jak snadné nastavení parametrů souborů dat vstupujících ke zpracování, tak přehledné zobrazení výsledků.

Definováním základních požadavků na vývoj software byla utvořena základní struktura programu, včetně vymezení formátu vstupních dat, automatizace nastavení jejich parametrů či analýzy a vizualizace dat v grafickém prostředí aplikace.

Pro vlastní naprogramování aplikace byl zvolen objektově orientovaný jazyk Python, který je snadno uchopitelný svou syntaxí a současně nabízí širokou škálu možností a dostupných knihoven pro vytváření komplexních aplikací. To včetně grafického uživatelského rozhraní, pro jehož tvorbu byla použita knihovna Kivy. Tato grafická knihovna umožnila oddělit logickou a základní vizuální část aplikace, díky možnosti definovat grafické prvky aplikace ve vlastním jednoduchém programovacím jazyce mimo hlavní skript programu. Jako velice výhodné se také ukázalo použití knihovny Matplotlib pro vizualizaci výsledků zpracování dat formou přehledných grafů, umožňující uživatelskou interaktivitu.

Pro nastavení vstupních parametrů struktury dat jsou výhodně vytvořeny předem uložené konfigurační profily, které umožňují okamžité nastavení struktury dat a jejich vstup do vyhodnocovacího algoritmu bez nutnosti při každém použití programu tyto parametry znovu nastavovat. V základu aplikace byly utvořeny profily struktury dat pro využívané datové formáty souborů na Katedře technologií a měření FEL ZČU, a to formáty výstupních souborů ze softwaru PSTrace určeného pro měřicí systémy PalmSens, či formáty ze softwaru NOVA 2.0 pro měřicí zařízení Metrohm Multi Autolab/M204. Samozřejmostí je potom možnost editace nebo přidání dalších takto přednastavených konfigurací, např. pro použití programu s daty jiného výrobce softwaru měřicího zařízení.

Pro vytváření a úpravu uživatelských profilů byl vytvořen konfigurační soubor aplikace, vč. příslušných ovládacích funkcí, který umožňuje změnu parametrů programu bez nutného zásahu do zdrojových kódů. Program díky vytvořenému konfiguračnímu souboru nabízí ještě další možnosti nastavení mimo grafické uživatelské prostředí. Lze vyzdvihnout možnost rozšíření aplikace na další podporované přípony souborů, či dále zmíněné úpravy některých konstant vstupujících do vlastní analýzy dat.

Ovládání programu je jednoduché a intuitivní díky jeho koncepci „průvodce“, kdy je uživatel od nahrání souborů až po zobrazení výsledků prováděn jednotlivými obrazovkami aplikace a současně jsou přístupné pouze ty ovládací prvky, které v danou chvíli mají funkční smysl. V každé dílčí obrazovce aplikace je dostupná vizuální kontrola nastavených parametrů s možností se v kterémkoliv kroku vrátit zpět a parametry upravit. Navíc je v aplikaci zavedeno množství naprogramovaných kontrolních funkcí, hlídající např. kolizní nastavení vstupních parametrů, chybějící parametry nastavení apod.

Navržené metody matematické analýzy dat se projevily jako vhodné k získání potřebných vypovídajících výsledků. Vlastní algoritmus analýzy je vytvořen odděleně od zbylé části programu ve vlastním skriptu, což působí nejen velice přehledně, ale je možné algoritmus pro budoucí využití ve zdrojovém kódu snadno upravovat, rozšiřovat či případně i nahradit jiným. Vlastní metodou určování proudových peaků a minim potřebných k určení základen peaků v naměřené voltametrické křivce je metoda filtrované první a druhé derivace, kdy je použito prokládání zvoleného okolí hodnoceného bodu regresní funkcí k filtraci šumu, resp. k zamezení lokalizace neužitečných extrémů. Algoritmus má naprogramovánu vlastní metodu určení šíře okna neboli vhodného okolí hodnoceného bodu, avšak v některých případech je vhodné ruční nastavení počáteční velikosti tohoto okna. Toto je umožněno vytvořenou nastavitelnou počáteční konstantou v konfiguračním souboru programu.

Dalším rozšířením je experimentální filtrace vyhodnocených peaků, jejímž principem je automatická volba největšího zjištěného peaku jako referenčního, se kterým jsou následně porovnávány další nalezené peaky. Ty, které jsou svojí výškou v poměru k referenčnímu peaku zanedbatelné, jsou z výsledků analýzy vynechány. Tento poměr je nastaven pomocí konstanty určující nejmenší relativní velikost k referenční velikosti peaku. Nastavení konstanty je opět výhodně provedeno pomocí konfiguračního souboru. Samozřejmostí je možnost tuto experimentální filtraci vynechat jednoduchým nastavením nulové hodnoty.

Grafická prezentace výsledků je přehledně provedena formou dvou pod sebou umístěných grafů, kdy horní graf zobrazuje souhrnně všechny dávkově zpracované datové řady a slouží především jako rychlý náhled opakovatelnosti jednotlivých měření s možností uživatelské analýzy chybných měření a odlehlých hodnot. Nevyhovující datové řady lze díky vytvořeným interaktivním funkcím grafů vypínat přímo z grafu či legendy a tyto datové řady jsou z výsledků analýzy tímto vynechány. Současně dojde k automatickému přepočítání zobrazení grafu, aby vždy nejlépe vyhovoval zobrazení viditelných datových řad. Spodní graf potom umožňuje analýzu jednotlivých vzorků se zobrazením základen a výšek zjištěných peaků, včetně označení jednotlivých peaků s odkazem na numerické vyjádření výsledků analýzy, tedy polohy a výšky daného peaku. Dále je možné přepnutím záložky zobrazit souhrnné numerické výsledky analýzy všech analyzovaných voltametrických křivek.

Výsledky zpracování dat lze exportovat jako jednotlivé obrázky obsahující graf vybraného vzorku včetně numerických výsledků. Případně lze také exportovat všechna výsledná numerická data pro další potřebu ve formátu prostého textu, čitelném v textových procesorech a současně i tabulkových kalkulátorech.

Ověření správnosti vyhodnocení voltametrických signálů vytvořeným programem proběhlo úspěšně, na reálných datech z měřicího systému PalmSens. Testování prokázalo správné určení proudových peaků ze zkoumaného vzorku. Současně byla ověřena správná funkce testovacích a filtrovacích sekvencí algoritmu a nevyhovující nebo neúplně určené peaky jsou z výsledků analýzy vyloučeny.

Seznam literatury a dalších informačních zdrojů

[1] BAREK, Jiří, OPEKAR, František a ŠTULÍK, Karel. Elektroanalytická chemie. Praha: Karolinum, 2005. 188 s. ISBN 80-246-1146-5.

[2] ZOSKI, Cynthia G. Handbook od Electrochemistry. Elsevier, 2007. ISBN-13 978-0-444-51958-0. ISBN-10 0-444-51958-0.

[3] REKTORYS, Karel a spolupracovníci. Přehled užité matematiky I. Sedmé vydání. Praha: Prometheus, 2007. 720 str. ISBN 978-80-7196-180-2.

[4] HÁTLE Jaroslav, LIKEŠ Jiří. Základy počtu pravděpodobnosti a matematické statistiky. Druhé, nezměněné vydání. Praha: SNTL 1974. 464 str.

[5] 3.6.7 Documentation. [online]. Python Software Foundation, ©2001-2018. [Cit. 20.5.2019]. Dostupné z: https://docs.python.org/release/3.6.7/

[6] Kivy framework — Kivy 1.10.1 documentation. [online]. kivy.org. [Cit. 20.5.2019]. Dostupné z: https://kivy.org/doc/stable/api-kivy.html

[7] API Overview — Matplotlib 3.1.0 documentation. [online]. matplotlib.org, ©2002-2012. [Cit. 20.5.2019]. Dostupné z: https://matplotlib.org/api/#

[8] GitHub - kivy-garden/garden.matplotlib: Matplotlib backends using kivy. [online]. GitHub, Inc., ©2019. [Cit. 20.5.2019]. Dostupné z: https://github.com/kivy-garden/garden.matplotlib

Přílohy

Přiložený nosič CD-R – obsah:

DAVM – adresář obsahující zdrojové kódy aplikace

DAVM_v1.0 – adresář obsahující spustitelný program vč. potřebných knihoven