



Politechnika Śląska
Wydział Automatyki, Elektroniki i Informatyki

Praca dyplomowa magisterska

Aplikacja internetowa do szacowania niepewności pomiaru
elektrochemicznego.

Autor: Mateusz Herok
Kierujący pracą: dr inż. Józef Wiora

Gliwice, wrzesień 2010

Spis treści

Spis rysunków	4
Wykaz ważniejszych oznaczeń	5
Rozdział 1. Cel i zakres pracy	6
Rozdział 2. Podstawy teoretyczne	7
2.1. Potencjometria a instrumentalne metody analizy chemicznej	7
2.2. Wykorzystanie metod potencjometrycznych	8
2.3. Stanowisko pomiarowe i sposób dokonywania pomiarów	13
2.4. Modele opisujące zależność ISE od aktywności jonów	15
2.4.1. Model Nernsta	15
2.4.2. Model Nikolskiego-Eisenmana	16
2.5. Pomiar wieloskładnikowy	16
2.6. Błędy oraz statystyczna ocena wyników	17
2.6.1. Błędy w analizach chemicznych	17
2.6.2. Statystyczna ocena poprawności wyników	19
2.6.3. Metoda Monte Carlo	23
Rozdział 3. Aplikacja internetowa	25
3.1. Wykorzystane środowiska	25
3.1.1. HTML	25
3.1.2. CSS	26
3.1.3. PHP	26
3.1.4. JavaScript	27
3.1.5. Octave	28
3.2. Elementy aplikacji	28
3.2.1. Wprowadzanie danych	28
3.2.2. Komunikacja i sesje	36
3.2.3. Wyznaczanie aktywności	38
3.2.4. Wyznaczanie niepewności	41
3.2.5. Wyświetlanie wyników	44
3.3. Testy	45
3.3.1. Testowanie wyznaczania aktywności	46
3.3.2. Testowanie wyznaczania niepewności	48

Rozdział 4. Zakończenie	52
4.1. Podsumowanie i wnioski	52
4.2. Propozycje dalszego rozwoju pracy	53
Dodatek A. Kod źródłowy niektórych plików	56
Dodatek. Bibliografia	67

Spis rysunków

2.1	Strona w nowatorskiej publikacji F. Habera i Z. Klemensiewicza.	9
2.2	Układ pomiarowy.	13
2.3	Schemat elektryczny układu pomiarowego.	14
2.4	Schemat modelu wieloskładnikowego.	17
2.5	Krzywe Gaussa w zależności od parametrów.	20
3.1	Początkowy wygląd formularza.	33
3.2	Przykładowy wygląd formularza.	33
3.3	Przykładowo wyświetlone wyniki.	46
3.4	Przykładowe obliczenia aktywności 1.	47
3.5	Przykładowe obliczenia aktywności 1.	48
3.6	Przykładowe szacowania niepewności 1.	49
3.7	Przykładowe szacowania niepewności 2.	50
3.8	Przykładowe szacowania niepewności 3.	51

Wykaz ważniejszych oznaczeń

- E : siła elektromotoryczna ogniwa, mV;
 E° : napięcie odniesienia elektrody, mV;
 S : współczynnik Nernsta, $S = \ln(10) \frac{RT}{z_i F}$, mV;
 R : stała gazowa, $R = 8.3143$, $J \cdot mol \cdot K^{-1}$;
 T : temperatura bezwzględna, K;
 z_i : ładunek jonu i ;
 F : stała Faraday'a, $F = 96484$, $C \cdot mol^{-1}$;
 $K_{i,j}^{Pot}$: współczynnik selektywności elektrody i na jon j ;
 a_i : aktywność jonu i .
 a_{0i} : granica wykrywalności dla jonu i ;
 u_c : niepewność standardowa;
 U : niepewność rozszerzona.

Rozdział 1

Cel i zakres pracy

Celem pracy jest przygotowanie aplikacji internetowej ułatwiającej wyznaczenie w łatwy i intuicyjny sposób aktywność i jej niepewność w pomiarach potencjometrycznych. Wyznaczanie niepewności w sposób analityczny najczęściej jest czynnością trudną i czasochłonną składającą się z wielu etapów. Dlatego istnieje zapotrzebowanie na system umożliwiający zastąpienie pracochłonnych obliczeń analitycznych dobrym przybliżeniem numerycznym.

Niezbędne w wykonaniu tego zadania było nabycie wystarczającej wiedzy na temat samej potencjometrii i sposobu przeprowadzania tego typu pomiarów. Należało uwzględnić zachodzące zjawiska fizykochemiczne, budowę układu pomiarowego oraz zastosowanych ogniw. Dochodzi do tego wiedza z zakresu statystyki przy szacowaniu i wyznaczaniu błędów i niepewności pomiarowych. Trzecią niezbędną dziedziną jest umiejętność napisania odpowiedniego oprogramowania zarówno dla obliczeń numerycznych, jak i dla zbudowania odpowiedniego interfejsu do komunikacji z użytkownikiem [11].

Najważniejszymi wytycznymi aplikacji są:

- wyznaczanie aktywności i szacowanie niepewności z dużą dokładnością;
- prosty interfejs z podstawowymi funkcjami sprawdzającymi wprowadzane dane;
- możliwość wykorzystania zewnętrznych plików przy wprowadzaniu danych wejściowych;

Niniejsza praca składa się z kilku części: wstępu teoretycznego, przybliżającego omawiane zagadnienia i problematykę, szczegółowego opisu elementów oprogramowania, części testowej i podsumowującej osiągnięte rezultaty wraz z wytycznymi, które udało się zrealizować. Na końcu zamieszczono również dodatek w skład którego wchodzi ważniejsze listy programów.

Rozdział 2

Podstawy teoretyczne

Rozdział ten przybliża zagadnienia teoretyczne związane z potencjometrią. Opisuje dziedziny, w których jest wykorzystywana oraz sposób dokonywania pomiarów. Zawiera także przegląd modeli matematycznych, które pozwalają powiązać wyniki pomiarów z aktywnością jonów oraz opis pojęć związanych z błędami i niepewnością pomiarową aktywności.

2.1. Potencjometria a instrumentalne metody analizy chemicznej

Pomiary elektrochemiczne to instrumentalne metody analizy chemicznej służące do wyznaczania składu chemicznego roztworu [13].

Sam *pomiar składu chemicznego* dążący do dużej dokładności i pewności wyników jest bardzo ważną, trudną i dość kosztowną czynnością [14]. Trudność wynika z wykorzystywania przez aparatury pomiarowe złożonych i różnorodnych zjawisk fizycznych, z częstego stosowania mikroskopijnych próbek oraz wpływu warunków otoczenia. Badanie składu chemicznego jest bardzo istotne w wielu dziedzinach życia. Dzięki tym pomiarom można orzekać skład leków, żywności, wód, czy wielu produkowanych materiałów i wyrobów. Sama aparatura pomiarowa, zwłaszcza ta złożona, dająca wiele możliwości zastosowania jest kosztowna.

Natomiast *instrumentalne metody analizy chemicznej* to takie, które wykorzystują zjawiska fizyczne lub fizykochemiczne, do których przeprowadzania wykorzystuje się odpowiednią aparaturę fizyczną lub fizykochemiczną [13]. Jest to niestety pojęcie bardziej ogólne, ponieważ do przeprowadzenia chociażby metody wagowej niezbędne jest zastosowanie dwóch ważeń, za pomocą wagi analitycznej, która też jest swego rodzaju „aparaturą”. Jednakże taką definicję przyjęło się właśnie stosować. *Metody instrumentalne* to takie metody, w których sygnał analityczny uzyskuje się za pomocą aparatury o różnym stopniu złożoności [15].

Metody elektrochemiczne to takie metody instrumentalne, które opierają się na pomiarze jakiejś wielkości elektrycznej, która jest w zależności funkcjonalnej od stężenia badanej

substancji [13]. Najczęściej wykorzystuje się do tego typu pomiarów dwie elektrody połączone do układu pomiarowego. Zanurzone w badanym roztworze tworzą tzw. ogniwo galwaniczne.

Potencjometria to jeden ze sposobów pomiarów elektrochemicznych. Jest jedną z najstarszych metod instrumentalnych w analizie chemicznej [16]. Wykorzystuje się w niej pomiar siły elektromotorycznej (SEM) ogniwa galwanicznego. Siła ta zależy od potencjałów elektrod jonoselektywnych (ISE) ogniwa galwanicznego, a te zależą od aktywności jonów w roztworze. Miejsce potencjometrii w pomiarach składu chemicznego obrazuje następujące wyszczególnienie [11]:

Pomiary składu chemicznego:

- (1) metody spektroskopowe,
- (2) metody chromatograficzne,
- (3) metody optyczne,
- (4) metody radiometryczne,
- (5) **metody elektrochemiczne**,
 - (a) konduktometria,
 - (b) amperometria,
 - (c) woltamperometria,
 - (d) polarografia,
 - (e) oscylometria,
 - (f) **potencjometria**.

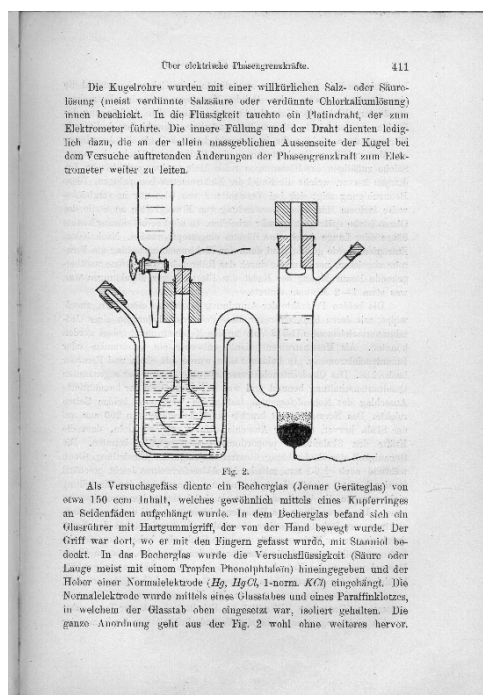
Biorąc pod uwagę opisaną już istotność wykonywania pomiarów składu chemicznego ważne jest, aby uzmysłowić sobie, że za pomocą metody potencjometrycznej można określić twardość lub zanieczyszczenia wody użytkowej, kwasność padającego deszczu, skład substancji odżywczych, krwi lub innych płynów ustrojowych. Liczba dziedzin przemysłu i nauki, które coraz częściej sięgają po tego typu pomiary wciąż rośnie, dlatego warto przedstawić, jak dziś ta sytuacja kształtuje się.

2.2. Wykorzystanie metod potencjometrycznych

Wraz z rozwojem nauki i wielu gałęzi przemysłu rosło i rośnie zapotrzebowanie na rozwój i dostęp do coraz lepszych i tańszych metod mierniczych.

Mimo iż podwaliny naukowe, z których metoda ta korzysta, zostały opracowane już w XVIII i XIX w. przez takich fizyków jak L. Galvani, A. Volta, czy W. Nernst [16], cała historia rozpoczęła się równo 100 lat temu. Wtedy to niemiecki chemik F. Haber (znany

choćby z wynalezienia Cyklonu B¹) i polski naukowiec Z. Klemensiewicz (od 1956 r. profesor Politechniki Śląskiej w Gliwicach²) skonstruowali szklaną elektrodę służącą do pomiarów pH [1], której autorski schemat przedstawia Rys. 2.1. Udowadniając w ten sposób zależność SEM od aktywności jonów wodorowych, opisaną wcześniej tzw. równaniem Nernsta, przez niemieckiego noblistę, fizyka i chemika W.H. Nernsta (odkrywcę zjawiska termomagnetycznego, znany również z pracy nad gazami bojowymi w czasach I w. ś.). Od tego czasu pomiar pH, czyli pomiar aktywności jonów wodorowych, jest najczęściej wykonywanym pomiarem potencjometrycznym w analizie takich substancji jak: woda, płyny ustrojowe, substancje odżywcze, czy inne produkty przemysłowe [16]. Z biegiem lat rozwijano badania nad opracowaniem elektrod zbudowanych z innych substancji niż szkło oraz selektywnych na coraz większą liczbę jonów. Dzisiaj ich liczba wynosi około 30 przy wyznaczaniu bezpośrednim oraz znacznie więcej przy oznaczeniach pośrednich. Dzięki temu pomiary potencjometryczne mogły zostać rozszerzone o takie działy jak analiza medyczna i badania biologiczne, czy również dalsze gałęzie przemysłu i ochrona środowiska [4].



RYSUNEK 2.1. Strona w nowatorskiej publikacji F. Habera i Z. Klemensiewicza.

Źródło: http://commons.wikimedia.org/wiki/File:Glass_Electrode.jpg dostęp: Czerwiec

2010

Oprócz wymienionego już pomiaru aktywności jonów wodoru zastosowanie innych elektrod nad którymi pracowano w XX w. pozwoliło szerzej wykorzystać potencjometrię w

¹ Źródło: http://pl.wikipedia.org/wiki/Cyklon_B, dostęp: Czerwiec 2010.

² Źródło: J. Kapłon, *Zygmunt Klemensiewicz*, http://www.cotg.pttk.pl/encyklopedia_tresc.php?id=119&szukaj=, dostęp: Czerwiec 2010.

medycynie. Używa się jej m.in. w pomiarach składu płynów ustrojowych takich jak krew, osocze, pot, ślina i mocz [1]. Dzięki monitorowaniu aktywności potasu we krwi możliwe jest w pewnym zakresie przewidywanie zawałów serca [2]. Często stosowanymi pomiarami są również badania alkoholu czy innych substancji psychoaktywnych we krwi. Zastosowanie lekkich analizatorów potencjometrycznych pozwala np. na stosowanie takich praktyk w zakładach pracy. Ważnym, jeśli nie najważniejszym czynnikiem wpływającym na istotność zastosowania tego typu pomiarów w medycynie, jest możliwość wykonywania ich w czasie rzeczywistym. Dzięki temu możliwy jest stały monitoring, np. przy stole operacyjnym, czy podczas reanimacji, gdzie tradycyjne metody laboratoryjne nie miałyby sensu. Takie pomiary są jednak często obciążone błędami i wrażliwością na zmiany współczynnika aktywności. [1]

W ochronie środowiska potencjometria jest wykorzystywana przede wszystkim w analizie wszelkich zbiorników i cieków wodnych. Do wcześniej omówionych pomiarów zasadowości czy kwasowości, dochodzą pomiary zawartości związków toksycznych rzek, jezior, mórz, opadów atmosferycznych, ścieków, wód gruntowych, jak i samej gleby [4]. Również w tej dziedzinie duże znaczenie ma zarówno łatwość dokonywania pomiarów, jak i ich ciągłość w czasie. Zwłaszcza w przypadku pomiaru amoniaku i azotanów w wodzie i ściekach [6]. Najczęściej oznaczanymi pierwiastkami i związkami są również fluorki, chlor i chlorki w wodach oraz cyjanki, siarkowodory i siarczki w ściekach [16]. Pomiary muszą być dokonywane okresowo, jednakże w porównaniu do innych metod, w tym laboratoryjnych, co kilkudziesięciosekundowe pomiary obiektów o tak dużych stałych czasowych jak np. jezioro, czy rzeka, dają wystarczającą dokładność do oceny ich najmniejszych rzeczywistych zmian. Dzięki zastosowaniu technologii GPRS można w łatwy sposób zbudować cały system monitoringu i kontroli np. zanieczyszczeń wód pitnych. Z pewnością jest to dużo tańszy i wydajniejszy sposób pomiaru od procedur laboratoryjnych, czy innych nierzadko droższych instrumentów analitycznych.

Ważną dziedziną, w której pomiar w czasie rzeczywistym jest bardzo istotny, jest przemysł. Najczęściej w nim stosowanym pomiarem potencjometrycznym jest pomiar pH. Mimo, iż w porównaniu do monitoringu cieków wodnych, możemy tutaj mówić o obiektach i całych układach regulacji o zdecydowanie mniejszych stałych czasowych, to jednak, jak wykazuje praktyka, są one na tyle duże, że czas pomiaru pozostaje bez większego znaczenia. Dużym utrudnieniem, zwłaszcza w przemyśle, jest natomiast niskie stężenie badanego składnika. Ważny jest także odpowiedni dobór elektrody w zależności od wielu czynników, takich jak: temperatura, objętość, ciśnienie, czy zakres mierzonego pH. W przemyśle pomiary potencjometryczne najczęściej wykonuje się w [11]:

(1) Produkcji:

- (a) wody, napojów,
- (b) wina i piwa,

- (c) konserw (badanie azotanów wpływających na ich korozję),
 - (d) detergentów,
 - (e) papieru,
 - (f) materiałów wybuchowych,
 - (g) nawozów sztucznych.
- (2) Oraz:
- (a) konserwacji mięsa, ryb, nabiału, soków,
 - (b) recyklingu.

Istotny w potencjometrii jest fakt, że nie dokonuje się bezpośrednio pomiaru stężenia jonów czy pierwiastka w badanej substancji, tylko aktywności jonów, która następnie poprzez współczynnik aktywności jest powiązana ze stężeniem. Ma to bardzo duże znaczenie w wielu dziedzinach nauki, np. w medycynie, gdzie ilość danego pierwiastka, a jego ilość w postaci jonowej to dwie mające inne znaczenie kwestie, a ich rozróżnienie jest bardzo ważne [2]. Chociaż czasem zaleta ta potrafi być również wadą, kiedy mimo dobrego pomiaru jonów wapnia, poznanie całkowitego stężenia tego pierwiastka za pomocą elektrod ISE nie jest już takie proste [1].

Patrząc na rozwój i możliwości, przyszłościowym kierunkiem dla potencjometrii jest zastosowanie mikroelektrod. W ostatnim czasie zastosowanie miniaturyzacji w tym dziale metrologii pozwoliło np. na pomiary gazometrii, elektrolitów, czy metabolitów, przydatne zwłaszcza w chorobach sercowo-naczyniowych. Dzięki rozwojowi genetyki, bioinformatyki i mikrotechnologii dziś możliwe są również pomiary i analizy DNA, mikroorganizmów, przeciwciał, enzymów itd. Niewielkie rozmiary pozwalają na badania poszczególnych organów i komórek [2]. Jednakże szybkość reakcji organizmów żywych wymusza wykorzystywanie elektrod o bardzo niewielkim czasie oczekiwania. To oraz wczesny etap rozwoju wyżej wymienionych dziedzin nauki sprawia, że zastosowanie miniaturyzacji w tym dziale miernictwa jest dopiero rozwijane i wiele koncepcji istnieje dopiero w fazie projektowania. Szerokie możliwości zastosowania świadczą o przyszłości i możliwej drodze rozwoju dla pomiarów z zastosowaniem elektrod jonoselektywnych [1].

Prawdopodobnie najciekawszym zastosowaniem jest wykorzystanie tzw. matrycy sensorowej, czyli zestawu wielu sensorów o zróżnicowanej selektywności (pomiar wieloskładnikowy), do budowy elektronicznego języka. Projekt ten był rozwijany chociażby na Wydziale Chemicznym Politechniki Warszawskiej [18]. Generalnie problem ten sprowadza się do zminiaturyzowanego pomiaru wieloskładnikowego oraz przyrównania otrzymanych wyników do istniejącej bazy danych substancji o określonym składzie chemicznym. Dokonuje się tego za pomocą technik rozpoznawania obrazów, często wykorzystywane w tym celu są sztuczne sieci neuronowe.

Z biegiem lat pomiary elektrochemiczne, a zwłaszcza potencjometryczne, o których była mowa, zaczęły pełnić ważną rolę w medycynie, nauce, czy przemyśle i przyczyniają

się do rozwoju tych dziedzin. Powtarzalne, szybkie, tanie, łatwe w przeprowadzeniu oraz w zautomatyzowaniu nie rzadko stanowią przewagę ich zastosowania nad tradycyjnymi metodami laboratoryjnymi. Stosując tę metodę należy liczyć się z pewnymi trudnościami i ograniczeniami. Poniżej przedstawiono zalety i wady wykorzystywania tej metody [1,2,4].

Zalety:

- (1) Prostota i szybkość pomiaru.
- (2) Stosunkowo niewielki koszt aparatury.
- (3) Możliwy pomiar w warunkach ciągłego przepływu.
- (4) Względnie niska cena i łatwość zastosowania.
- (5) Możliwość pomiaru w szerokim zakresie stężeń.
- (6) Duża odporność współczesnych elektrod.
- (7) Wysoka praktyczność przy oszacowywaniu przybliżonych stężeń.
- (8) Niewielkie rozmiary i mały pobór mocy sprawiają, że elektrody jonoselektywne mogą z powodzeniem być stosowane w układach zdalnego monitoringu.
- (9) Bezpośrednio mierzona aktywność jonów, a nie samo stężenie. Przydatne zwłaszcza w medycynie.
- (10) Przy ostrożnym użytkowaniu i częstej kalibracji możemy mówić o bardzo dobrej dokładności i powtarzalności tej metody.
- (11) Możliwość pomiaru zarówno anionów jak i kationów.
- (12) Kolor i mętność próbki nie wpływa na jakość pomiaru.
- (13) Możliwość wykorzystania w roztworach wodnych o bardzo szerokim zakresie temperatury.
- (14) Granica wykrywalności nie pogarsza się wraz ze zmniejszaniem objętości próbki. Możliwość miniaturyzacji i pomiarów in vivo.

Wady:

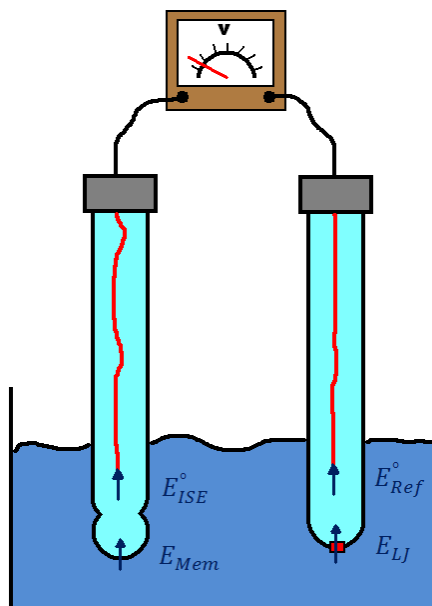
- (1) Zmienność parametrów elektrod wraz z upływem czasu – konieczność kalibracji.
- (2) Możliwość zatruwania i zanieczyszczania się elektrod pomiarowych.
- (3) Ograniczony czas życia elektrody.
- (4) Zależność wartości parametrów od temperatury.
- (5) Zależność wskazań również od aktywności jonów nie mierzonych, a znajdujących się w roztworze.
- (6) Aktywność nie wszystkich jonów można mierzyć za pomocą potencjometrii.

2.3. Stanowisko pomiarowe i sposób dokonywania pomiarów

Pokrótkie została już omówiona potencjometria jako dział metrologii. Opisano również, gdzie pomiary takie są wykorzystywane oraz jakie wady i zalety stanowią o popularności tej metody. W tej części opisano proces pomiarowy i elementy niezbędne do jego przeprowadzenia.

Najbardziej zasadniczy podział układu pomiarowego w badaniach potencjometrycznych to [16]:

- (1) ogniwo galwaniczne – pary elektrod jonoselektywnych zanurzonych w roztworze;
- (2) miernik pozwalający na pomiar siły elektromotorycznej ogniwa – najczęściej woltomierz.



RYSUNEK 2.2. Układ pomiarowy.

Symbole:

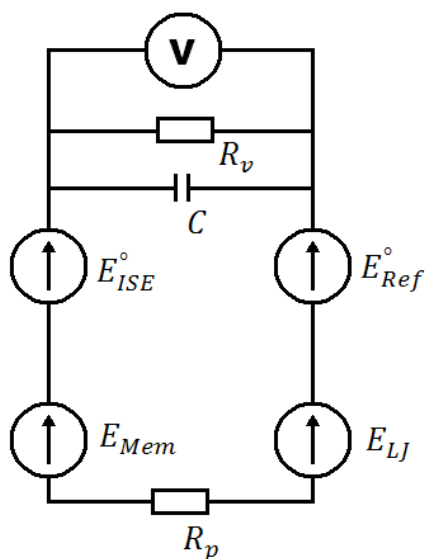
E_{Ref}° - różnica potencjału drut-roztwór wewnętrzny elektrody odniesienia;

E_{LJ} - potencjał dyfuzyjny łącznika elektrolitycznego elektrody odniesienia;

E_{ISE}° - różnica potencjału drut-roztwór wewnętrzny elektrody pomiarowej;

E_{Mem} - potencjał membranowy elektrody pomiarowej.

Z dwóch wykorzystywanych elektrod jedna jest elektrodą pomiarową, wskaźnikową, której potencjał zależny jest od stężenia oznaczanego jonu, natomiast druga jest elektrodą porównawczą o stałym potencjale w warunkach prowadzenia pomiaru [13]. Praktycznie każda elektroda jonoselektywna może być wykorzystana jako porównawcza pod warunkiem, że wykazuje stały potencjał w badanym roztworze [1]. Zastosowanie ogniwa jest



RYSUNEK 2.3. Schemat elektryczny układu pomiarowego.

Symbole:

E_{Ref}° - różnica potencjału drut-roztwór wewnętrzny elektrody odniesienia;

E_{LJ} - potencjał dyfuzyjny łącznika elektrolitycznego elektrody odniesienia;

E_{ISE}° - różnica potencjału drut-roztwór wewnętrzny elektrody pomiarowej;

E_{Mem} - potencjał membranowy elektrody pomiarowej;

R_v - rezystancja wewnętrzna woltomierza;

R_p - rezystancja roztworu;

C - pojemność przewodów układu pomiarowego.

niezbędne, bowiem potencjał elektrody pomiarowej względem roztworu nie jest dostępny dla bezpośredniego pomiaru [4]. Przewodnikiem zamykającym połączenie dwóch elektrod jest sam roztwór. Zjawiskiem zapewniającym powstawianie potencjału jest wytwarzanie się podwójnej warstwy na granicy dwóch faz. Żeby pomiar przeprowadzić prawidłowo, bardzo ważne jest aby odbywał się on w warunkach bezprądowych, tzn. zastosowanie woltomierza o bardzo dużej oporności wewnętrznej tworzy układ otwarty.

Na Rys. 2.2 przedstawiono model układu pomiarowego, zawierający oprócz elektrody pomiarowej również elektrodę odniesienia, pomiędzy którymi generuje się siła elektromotoryczna. Wcześniej wymieniana łatwość dokonywania pomiarów potencjometrycznych polega na tym, że tylko same elektrody, a nawet tylko ich części, muszą zostać zanurzone w analizowanym roztworze. Nie istnieje konieczność pobierania samej próbki, dlatego np. przy zastosowaniu odpowiednio długich przewodów, można dokonać analizy wody w rzece, opuszczając same elektrody jonoselektywne poniżej poziomu nurtu.

Na całkowitą elektrochemiczną różnicę potencjałów składa się kilka składowych. Największe znaczenie ma jednak potencjał membranowy. Na Rys. 2.3 przedstawiono elektryczny schemat zastępczy znacznie lepiej przybliżający parametry mające wpływ na mierzoną siłę SEM. Należy pamiętać, że rezystancja wewnętrzna woltomierza powinna być jak największa, aby w układzie poza membraną nie płynęły prądy. Powiązać mierzoną różnicę potencjałów z aktywnością jonów można wykorzystując któreś z opisanych w następnym rozdziale modeli matematycznych.

2.4. Modele opisujące zależność ISE od aktywności jonów

2.4.1. Model Nernsta

Wzmiankowany już wcześniej model i równanie Nernsta, opublikowany w 1880 r. stanowi podstawową zależność w elektrochemii. Mimo, iż przeznaczony jest dla roztworów czystych, często wykorzystuje się go do analiz jonu w mieszaninach wieloskładnikowych. Wtedy należy rozróżnić jon główny, który przede wszystkim decyduje o potencjale elektrody oraz jony zakłócające. W najbardziej popularnej formie równanie opisujące zależność siły elektromotorycznej ogniwa od aktywności jonu wygląda następująco:

$$E = E^{\circ} + S \lg(a) \quad (1)$$

gdzie:

E : różnica potencjałów mierzona pomiędzy elektrodą pomiarową a elektrodą odniesienia, mV;

E° : napięcie odniesienia elektrody, mV;

S : współczynnik Nernsta, $S = \ln(10) \frac{RT}{zF}$, mV;

R : stała gazowa, $R = 8.3143, J \cdot mol \cdot K^{-1}$;

T : temperatura bezwzględna, K;

z : ładunek jonu;

F : stała Faraday'a, $F = 96484, C \cdot mol^{-1}$;

a : aktywność jonu.

Najczęściej w rzeczywistych układach pomiarowych wartości S i E° wyznacza się eksperymentalnie. W równaniu (1) model Nernsta uwzględnia tylko wpływ temperatury i aktywność jednego jonu. Rzadko jednak analizuje się tzw. roztwór czysty, czyli mieszaninę jednej substancji. Nie istnieje także idealnie selektywna elektroda, czyli taka, na potencjał której ma wpływ tylko jeden określony jon. W związku z tym często pojawia się potrzeba zastosowania bardziej skomplikowanych modeli.

2.4.2. Model Nikolskiego-Eisenmana

Model ten uwzględnia aktywność jonów zakłócających na wynik pomiaru. Istnieją bowiem jony na tyle podobne pod względem właściwości, że mimo iż elektrody powinny być selektywne tylko na jeden jon, w dużym stopniu reagują zmianą różnic potencjału na aktywność innych jonów. Równanie Nikolskiego-Eisenmana najczęściej występuje w następującej postaci:

$$E_i = E^\circ + S \lg \left(a_i + \sum_{j \neq i} K_{i,j}^{\text{Pot}} a_j^{z_i/z_j} \right) \quad (2)$$

Wykorzystano takie same oznaczenia jak w modelu Nernsta (1). Ponadto $K_{i,j}^{\text{Pot}}$ oznacza współczynnik selektywności dla jonów głównych i względem jonów zakłócających j .

Jeśli współczynnik selektywności dla wszystkich jonów zakłócających będzie wynosił zero (elektroda idealnie selektywna), równanie przyjmie postać równania Nernsta. Model Nikolskiego-Eisenmana jest zatem rozwinięciem modelu Nernsta, lub inaczej rzecz ujmując ten drugi jest szczególnym przypadkiem pierwszego.

Współcześnie oprócz wyżej wymienionych stosuje się również dużo bardziej skomplikowany model potencjału na granicy faz, czy też wymagający rozwiązań równań różniczkowych i zastosowania dyskretyzacji model Nernsta-Plancka-Poissona. Ze względu na skomplikowany charakter nie będą one jednak tematem rozważań.

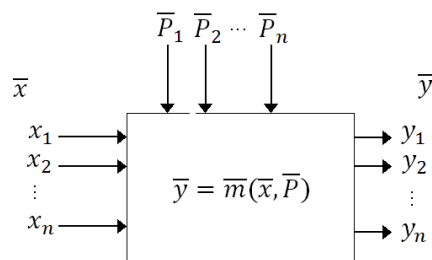
2.5. Pomiar wieloskładnikowy

Równanie Nikolskiego-Eisenmana pozwala powiązać pomiar SEM z aktywnością jonu głównego, uwzględniając przy tym aktywności wszystkich jonów zakłócających. Uwzględnienie ich wszystkich i zastosowanie z tego powodu wielu elektrod tworzy układ równań z wieloma niewiadomymi.

W pomiarach wieloskładnikowych mierzonych jest jednocześnie kilka składników próbki. Ma to uzasadnienie, kiedy wpływają one na siebie nawzajem, tak jak ma to miejsce w tej sytuacji. Zasada jest następująca. Jest dane kilka pomiarów, sygnałów z czujników x_i , tworzą one wektor $\mathbf{X}$. Dla każdej wartości wyjściowej y_i której poszukujemy, przyrównujemy funkcję modelu $m_i(\mathbf{X}, \mathbf{P})$, stanowiącą zależność wyjścia z wektorem pomiarów $\mathbf{X}$ i wektorem parametrów modelu $\mathbf{P}$. Zbiór wszystkich równań dla poszczególnych wyjść y_i utworzy układ równań, który możemy zapisać wektorowo: [11]

$$\mathbf{Y} = \mathbf{M}(\mathbf{X}, \mathbf{P}) \quad (3)$$

Gdzie wektor $\mathbf{Y}$ to zbiór wszystkich wyjść y_i , wektor $\mathbf{M}$ zbiór funkcji m_i , natomiast macierz $\mathbb{P}$ zawiera wszystkie wektory $\mathbf{P}_i$ poszczególnych funkcji modeli. Sytuację tę dobrze obrazuje ideowy Rys. 2.4.



RYSUNEK 2.4. Schemat modelu wieloskładnikowego.

W potencjometrycznym pomiarze wieloskładnikowym dokonuje się jednoczesnego pomiaru wielu wykorzystanych elektrod jonoselektywnych. Poszczególne zależności mierzonego SEM od analizowanych aktywności wyrażone równaniem Nikolskiego-Eisenmana (2) tworzą następujący układ równań:

$$\begin{cases} E_1 = E_1^0 + S_1 \cdot \lg \left(a_1 + \sum_{j \neq 1} K_{1,j}^{\text{Pot}} \cdot a_j^{z_1/z_j} + a_{0,1} \right) \\ E_2 = E_2^0 + S_2 \cdot \lg \left(a_2 + \sum_{j \neq 2} K_{2,j}^{\text{Pot}} \cdot a_j^{z_2/z_j} + a_{0,2} \right) \\ \vdots \\ E_n = E_n^0 + S_n \cdot \lg \left(a_n + \sum_{j \neq n} K_{n,j}^{\text{Pot}} \cdot a_j^{z_n/z_j} + a_{0,n} \right) \end{cases} \quad (4)$$

Znaczenie symboli jest identyczne jak w pojedynczym równaniu Nikolskiego-Eisenmana (2) uwzględniono jednak dodatkowo granicę wykrywalności jako $a_{0,i}$. Dla n elektrod tworzy się układ równań o n liczbie niewiadomych w postaci aktywności jonów a_i . Warunkiem koniecznym wyznaczenia aktywności jonów w pomiarze wieloskładnikowym jest zastosowanie elektrod w liczbie conajmniej takiej samej jak liczba analizowanych jonów. Porównując układ równań (4) z Rys. 2.4 można określić poszczególne aktywności jonów a_i jako elementy wektora niewiadomych $\mathbf{X}$, wartości mierzone E_i jako elementy wektora wielkości mierzonych $\mathbf{Y}$. Wszystkie pozostałe wartości, które powinny być znane przed pomiarem, np.: z dokonanej wcześniej kalibracji, można uznać za elementy macierzy parametrów $\mathbb{P}$.

2.6. Błędy oraz statystyczna ocena wyników

2.6.1. Błędy w analizach chemicznych

Opracowywanie wyników w pomiarach potencjometrycznych, jak we wszystkich innych metodach metrologicznych, wiąże się z uwzględnieniem obciążeń wieloma możliwymi

błędami. Ze względu na znaczenie i charakter ich występowania dzielą się one na trzy rodzaje [7]:

- (1) *Błędy przypadkowe* - praktycznie obarczone nimi są wszystkie pomiary. Mają niewielką wartość oraz znikomy wpływ na uśredniony wynik pomiaru. Przyczyna ich powstawania nie jest do końca znana, wiąże się ją najczęściej z zakłóceniami.
- (2) *Błędy systematyczne* - mają zupełnie inny charakter. Są stałe i deformują wynik zawsze w podobny sposób. Przyczyna ich występowania jest jednoznaczna i łatwa do wskazania. Najczęściej bierze się w tym przypadku pod uwagę niesprawność przyrządów, złe ich wykorzystanie lub nieskuteczność samej metody pomiarowej.
- (3) *Błędy grube* - zupełnie inny rodzaj błędów. Najczęściej powstają w wyniku tzw. czynnika ludzkiego w dokonywaniu pomiaru lub samych obliczeń. Uwzględnia się w nich również niewłaściwe sposoby pobierania i przechowywania badanej próbki. Jak można się domysleć błąd gruby może znacząco wpłynąć na poprawność wyniku końcowego.

Inny rodzaj podziału błędów odbywa się ze względu na sposób ich obliczania. W tym przypadku wyróżniamy:

- (1) *błędy bezwzględne*,
- (2) *błędy względne*.

Błąd bezwzględny jest różnicą pomiędzy rzeczywistą wartością μ , a zmierzoną x . Dlatego oblicza się go następująco:

$$d = x - \mu \quad (5)$$

Wyraża on tylko różnicę pomiędzy określonymi wartościami, które powinny mieć taką samą jednostkę. Błąd bezwzględny d przyjmuje również tę jednostkę i nic nie mówi o swojej istotności. Na przykład pomiar 101 mV przy rzeczywistej wartości 100 mV, a pomiar 2 mV przy rzeczywistym wyniku 1 mV da taką samą wartość błędu d . Dlatego często stosuje się relatywizującą modyfikację [7].

Błąd względny e jest modyfikacją błędu bezwzględnego. Jest on relacją błędu bezwzględnego do wartości rzeczywistej μ . Jako, że jednostki skracają się, istnieje w tym przypadku możliwość porównywania pomiarów wyrażonych w różnych jednostkach [7]. Błąd względny wyrażamy następująco:

$$e = \frac{|\mu - x|}{\mu} \quad (6)$$

Warto zauważyć, że do powyższego wyznaczania błędów konieczna jest znajomość rzeczywistej wartości którą mierzymy. Taka sytuacja rzadko kiedy ma miejsce, więc najczęściej są to pojęcia teoretyczne. Można je jednak wykorzystać np. podczas badania samej metody pomiarowej, kiedy dysponuje się próbką wzorcową (znana jest wtedy wartość μ).

2.6.2. Statystyczna ocena poprawności wyników

2.6.2.1. Rozkład normalny błędów pomiarowych. Wszystkie pomiary są obarczone pewnymi odchyleniami. Rozkład tych odchyłeń, jako wartości większych i mniejszych od wartości oczekiwanej, w przypadku wystąpienia tylko błędów przypadkowych, często przyjmuje w przybliżeniu kształt rozkładu normalnego (Gaussa). Charakteryzują go dwa parametry: wartość oczekiwana μ oraz odchylenie standardowe σ .

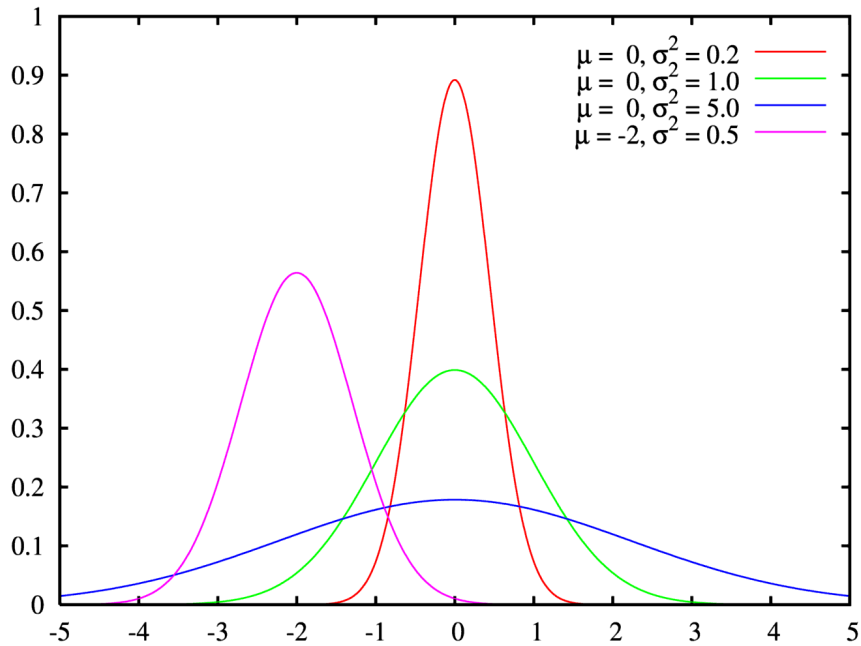
Dla nieskończonej liczby pomiarów ich wartość średnia $\bar{x}$ jest równa wartości oczekiwanej. Dla mniejszej liczby pomiarów jest to już tylko oszacowanie, które wraz ze zmniejszaniem liczby próbek obciążone jest coraz większymi błędami - często stosuje się wtedy medianę, jako wartość oczekiwaną. Do wyznaczania odchylenia standardowego również najczęściej bierze się wartość średnią pomiarów [7]. Wtedy wzór na estymowaną wartość odchylenia przyjmuje następującą postać:

$$s = \sqrt{\frac{1}{n-1} \sum (\bar{x} - x_i)^2} \quad (7)$$

gdzie n to liczba pomiarów, a x_i to kolejne ich wartości.

Na Rys. 2.5 przedstawiono funkcję gęstości prawdopodobieństwa rozkładu normalnego (w zależności od średniej μ i wariancji σ^2). Linia zielona odpowiada standardowemu rozkładowi Gaussa, odchylenie standardowe (również wariancja) jest wtedy równe 1. Wykres ten określa zależność liczebności pojawienia się błędu od jego wielkości bezwzględnej. Umożliwia także określenie prawdopodobieństwa przekroczenia danej wielkości błędu [7]. Przykładowo dla wykresu zielonego prawdopodobieństwo wystąpienia błędu większego niż 2 jest dużo mniejsze niż dla rozkładu danego wykresem niebieskim. W obu przypadkach największą licznosc mają pomiary $x_i \approx \mu$, jednakże dla drugiego możemy mówić o mniejszej dokładności pomiaru. Zatem im większe odchylenie standardowe tym większy rozrzut wyników - czyli parametr ten jest miarą dokładności metod analitycznych [7]. Zmiana parametru μ (wykres różowy) oznacza inną mierzoną próbkę o innej wartości średniej. Wartość oczekiwana jest zatem miarą rzetelności metody analitycznej [7].

2.6.2.2. Niepewność pomiaru. Każdy pomiar obarczony jest jakimś błędem. Aby podać kompletny wynik, powinien on zawierać oprócz wielkości mierzonej związaną z tą wartością niepewność pomiaru. Charakteryzuje ona rozrzut wartości, które są przypisane wielkości mierzonej [8].



RYSUNEK 2.5. Krzywe Gaussa w zależności od parametrów.

Źródło: http://commons.wikimedia.org/wiki/File:Normal_distribution_pdf.png dostęp:

Sierpień 2010

Wielkość mierzona, nazywaną również *wielkością wyjściową* Y , określaną przez dokonanie pomiaru oraz wyznaczenie modelu opisującego zależność pomiędzy wielkością mierzoną, a wielkościami wejściowymi [8]:

$$Y = f(X_1, X_2, \dots, X_n) \quad (8)$$

gdzie:

Y : wielkość mierzona;

X_i : zbiór wielkości wejściowych;

f : funkcja określająca zależność wielkości mierzonej od wielkości wejściowych.

Funkcja f opisuje zarówno metodę pomiarową, jak i metodę obliczeniową. Jako zbiór wartości wejściowych X_i uwzględnia się zatem zarówno wielkości, których oszacowania dane są bezpośrednio z pomiaru z uwzględnieniem wszelkich koniecznych poprawek jak i wielkości, których oszacowania i związane z nimi niepewności wynikają ze źródeł zewnętrznych takich jak: wzorce, dane odniesienia.

Estymatę wielkości mierzonej Y oznaczamy jako y i wyznaczmy na podstawie równania (8) podstawiając pod wartości wejściowe X_i ich estymaty x_i . Tak otrzymana zależność wygląda następująco:

$$y = f(x_1, x_2, \dots, x_n) \quad (9)$$

Zakładamy przy tym, że wykorzystane estymaty zostały skorygowane o wszelkie wielkości oddziałujące na model pomiaru, również te uznane za losowe (szumy). W przeciwnym

wypadku konieczne poprawki powinny zostać uwzględnione w modelu jako dodatkowe wejścia [8].

Niepewność standardowa pomiaru oznaczana jako $u_c(y)$ jest odchyleniem standardowym wielkości mierzonej Y . Określa się ją na podstawie estymat x_i wielkości wejściowych X_i oraz na podstawie związanych z nimi niepewnościami standardowymi $u_c(x_i)$. Przy podawaniu wyników podaje się jednak najczęściej niepewność rozszerzoną. Można wyróżnić dwie zasadnicze metody obliczania niepewności standardowej [11]:

- (1) *Szacowanie niepewności metodą typu A* - metoda stosowana dla serii pomiarów wykonanych w tych samych warunkach. Dla losowych błędów pomiarowych zakłada się rozkład normalny wyników pomiarów i wtedy miarą rozrzutu jest odchylenie standardowe.
- (2) *Szacowanie niepewności metodą typu B* - metoda analizy na podstawie badań eksperymentalnych układu pomiarowego oraz rozległej wiedzy o zachodzących zjawiskach. Wykorzystywana wszędzie tam gdzie statystyczna metoda typu A nie może być zastosowana. Metoda ta wymaga dogłębnej znajomości procesu pomiarowego i uwzględnienia wszystkich wpływających na dokładność czynników.

Uwzględnienie obu powyższych metod pozwala na wyznaczenie złożonej niepewności standardowej jako pierwiastka sumy kwadratów cząstkowych niepewności standardowych szacowanych metodą typu A oraz typu B:

$$u_c = \sqrt{u_A^2 + u_B^2} \quad (10)$$

Niepewność rozszerzona pomiaru oznaczana jako U uzyskuje się, mnożąc niepewność standardową $u_c(y)$ przez współczynnik rozszerzenia k . Najczęściej stosuje się $k = 2$ co odpowiada poziomowi ufności równemu około 95%.

$$U = k \cdot u_c(y) \quad (11)$$

Reasumując powyższe można stwierdzić, że wyniki pomiarów mogą być zapisywane jako $y \pm U$ przy uwzględnieniu, że najlepszą estymacją wartości mierzonej Y jest y . Przedział $< y - U, y + U >$ zawiera z wysokim prawdopodobieństwem wartość, którą można przypisać do wielkości mierzonej. Można to zapisać również jako następującą nierówność:

$$y - U \leq Y \leq y + U \quad (12)$$

Systematyczne błędy, które znane są przed i w czasie pomiaru powinny być uwzględnione już w etapie estymacji wartości mierzonej. Pozostałe szacuje się metodą typu B.

Najczęściej uwzględniając wiele niezależnych czynników o różnych niepewnościach dokonuje się ich składania wykorzystując prawo przenoszenia niepewności. Wtedy złożoną niepewność standardową możemy obliczyć następująco:

$$u_c = \sqrt{\sum_{i=1}^n \left(\frac{\partial f}{\partial x_i} \right)^2 u^2(x_i)} \quad (13)$$

gdzie:

- $u(x_i)$: niepewność standardowa wielkości wejściowej x_i ;
- $\frac{\partial f}{\partial x_i}$: pochodna funkcji (9) względem wielkości wejściowej x_i ;
- u_c : złożona niepewność standardowa szacowana metodą typu A lub B.

Powyższa zależność jest jednak prawdziwa tylko wtedy kiedy poszczególne wielkości wejściowe nie są skorelowane ze sobą. W przeciwnym wypadku musimy uwzględnić kowariancję wartości pomiędzy którymi występuje zależność. Wtedy powyższe równanie przyjmuje następującą postać:

$$u_c(x_1 \dots x_n) = \sqrt{\sum_{i=1}^n \left(\frac{\partial f}{\partial x_i} \right)^2 u^2(x_i) + 2 \sum_{i=1}^{n-1} \sum_{j=i+1}^n \frac{\partial f}{\partial x_i} \frac{\partial f}{\partial x_j} cov^2(x_i, x_j)} \quad (14)$$

gdzie:

- $cov(x_i, x_j)$: kowariancja wartości x_i i x_j .

Estymatę kowariancji możemy natomiast wyrazić następująco:

$$\widehat{cov}(x_i, x_j) = \frac{1}{n-1} \sum_{k=1}^n (x_{i\ k} - \bar{x}_i)(x_{j\ k} - \bar{x}_j) \quad (15)$$

gdzie:

- $x_{i\ k}, x_{j\ k}$: pary realizacji wielkości wejściowych x_i i x_j ;
- $\bar{x}_i, \bar{x}_j$: wartości średnie wielkości x_i i x_j .

Istotą ukazania równań (14) i (15) jest podkreślenie faktu, że w przypadku wieloskładnikowego pomiaru potencjometrycznego nie wykorzystuje się elektrod idealnie selektywnych. Poszczególne wielkości są zatem ze sobą powiązane i w przypadku współczynników selektywności istnieje macierz kowariancji, której wielkość zależy od ilości analizowanych jonów i wykorzystanych elektrod. Trudność odpowiednich obliczeń w algorytmach numerycznych jest zatem przyczyną dokonanego uproszczenia, które szerzej wyjaśniono w części omawiającej działanie aplikacji.

2.6.3. Metoda Monte Carlo

Jest to metoda numeryczna wykorzystywana przy rozwiązywaniu złożonych problemów dla których podejście analityczne jest bardzo trudne i czasochłonne. W tym celu wykorzystuje się symulacje komputerowe danego zjawiska przy jednoczesnym zastosowaniu generacji liczb pseudolosowych.

Twórcą był Stanisław Ulam, polski matematyk, znany m.in. jako współtwórca bomby wodorowej³. Metoda ta początkowo rzadko stosowana, wraz z rozwojem mocy obliczeniowej komputerów zyskiwała na coraz większej popularności. Istotną zaletą jej stosowania jest brak konieczności dokonywania eksperymentów i jedynie modelowanie i symulowanie wyników z wykorzystaniem liczb pseudolosowych w miejscu losowych wpływów na dane zjawisko. Losowość tych wartości związana jest z pewnymi rozkładami, znanymi np. z badań nad procesem lub z dokonanych założeń. Symulacyjność i wykorzystanie pewnego modelu sprawia zatem, że metoda ta ma więcej wspólnego z modelowaniem i przeprowadzaniem doświadczeń niż z klasycznymi metodami rachunkowymi [9].

Często wykorzystywana jest w obliczaniu całek wielowymiarowych, rozwiązywaniu równań różniczkowych i algebraicznych, optymalizacji, algorytmach genetycznych.

Dla lepszego zobrazowania tej metody można podać poniższy przykład. Należy wyznaczyć metodą symulacyjną pole koła wpisanego wewnątrz kwadratu o boku 2. W tym celu wyznacza się w sposób pseudolosowy wiele punktów wewnątrz kwadratu. Pole koła można wyznaczyć według następującego wzoru:

$$P_{\circ} = \frac{n_{\circ}}{n} P_{\square} \quad (16)$$

gdzie:

- $P_{\circ}$: pole koła;
- $P_{\square}$: pole kwadratu;
- $n_{\circ}$: liczba punktów wewnątrz koła;
- n : liczba wszystkich punktów (liczba punktów wewnątrz kwadratu).

Promień danego koła wynosi 1, a więc wyznaczane pole będzie równe co do wartości liczbie π . Wynik przeprowadzanej symulacji będzie jednak bliski jej wartości dopiero przy bardzo dużej ilości losowanych punktów n . Wtedy rozkład punktów jest dużo bardziej równomierny. Można zatem przyjąć, że stosunek:

$$\frac{n_{\circ}}{n} = \frac{P_{\circ}}{P_{\square}} \quad (17)$$

jest rzeczywiście prawdziwy dla $n \rightarrow \infty$. Ten opisany przykład jest często wykorzystywany do wyznaczania kolejnych znaków liczby π . Wykorzystuje się ten sposób również do

³ Źródło: <http://www.zwoje-scrolls.com/zwoje16/text03.htm>, dostęp: Czerwiec 2010.

szacowania całek oznaczonych.

Metodą Monte Carlo otrzymuje się pewne przybliżenie rozwiązania. Nie można bowiem wyznaczyć nieskończonej liczby punktów. Zwiększanie liczby losowań może również nie przynieść oczekiwanych rezultatów, nie wykorzystuje się bowiem liczb idealnie losowych i na ich wpływ ma postać generatora pseudolosowego. Mimo tych wad metoda ta ma zastosowanie wszędzie tam gdzie liczy się czas uzyskania rozwiązania, aniżeli matematyczna dokładność, zwłaszcza jeśli obliczenia analityczne są trudne do przeprowadzenia.

Rozdział 3

Aplikacja internetowa

Rozdział ten opisuje wykonaną aplikację internetową. Obejmuje zarówno opis użytkowania, przygotowania od strony technicznej jak i powiązania z opisanymi zagadnieniami teoretycznymi. W rozdziale tym opisano również środowiska i narzędzia jakie zostały wykorzystane.

3.1. Wykorzystane środowiska

Celem pracy było wykonanie aplikacji działającej online, do obsługi której wystarczy zwykła przeglądarka. Wiąże się to z wykorzystaniem tzw. języka znaczników HTML. Obsługa plików, formularzy, dynamicznych zdarzeń oraz obliczenia macierzowe wymagają jednak zastosowania również innych środowisk programistycznych.

3.1.1. HTML

Rozwinięcie skrótu *HTML* w języku angielskim to **H**yper**T**ext **M**arkup **L**anguage, czyli hipertekstowy język znaczników wykorzystywany do tworzenia stron www. Nazwa wywodzi się z używanych znaczników, zwanych tagami, które nadają specyficzne znaczenie poszczególnym fragmentom dokumentu.

Przez niecałe 20 lat, kiedy HTML żywo zmieniał się i rozwijał, ich liczba oraz wykorzystanie zwiększała się. W tym miejscu należy podkreślić, że HTML jest niezależny od platformy, na której jest wyświetlany. To znaczy, że system operacyjny nie ma znaczenia, ale już różne przeglądarki interpretują go w swój własny, czasem odmienny sposób. Wszystko to prowadziło do bałaganu i konieczności uporządkowania i wprowadzania pewnych specyfikacji. Najnowszą z nich jest HTML 4.01 i właśnie według niej dokumenty strony zostały wykonane. W specyfikacji tej dokonano oddzielenia warstwy logicznej od warstwy reprezentacyjnej tworzonej strony, której zadania przejęły kaskadowe arkusze stylów (CSS). Wiązało się to z koniecznością uznania wielu znaczników za przestarzałe. Mimo to, ze względu na wsteczną kompatybilność nadal działają. Początkowo nie przynosiło to jednak pożądanych rezultatów, ponieważ znowu różne przeglądarki różnie interpretowały CSS. Obecnie sytuacja ta jest coraz lepiej unormowana [12].

Wart wzmianki jest jeszcze fakt, że nowszym projektem jest XHTML oparty na XML (jest z nim zgodny). W uproszczeniu można powiedzieć, że jest to bardziej uniwersalny język znaczników. Mimo iż miał być niekompatybilnym następcą HTML, niekompatybilność wprowadzono dopiero w specyfikacji XHTML 2. Jednakże zarzucono nad nim pracę i obecnie wraca się do korzeni opracowując HTML 5 i obok niego XHTML 5.

3.1.2. CSS

Rozwinięcie skrótu *CSS* w języku angielskim to **C**ascading **S**tyle **S**heet, czyli kaskadowy arkusz stylów wykorzystywany do tworzenia stron www, a zwłaszcza definiowania jej części wizualnej. Jego zastosowanie pozwala na zwiększenie przejrzystości kodu samego HTMLa, ale również i na zwiększenie funkcjonalności i łatwości zmiany wyglądu strony www. Jest to możliwe ponieważ w osobnym pliku stylów możemy tworzyć i definiować wygląd wielu klas oraz wprost określonych obiektów. W samym HTML zamiast osobno nadawać atrybuty każdemu obiektowi, można je przypisać do danej klasy. Dzięki temu dokonując zmian jedynie w jednym pliku ze stylami możemy zmieniać wygląd całej strony, a edycja każdej klasy zmienia wszystkie obiekty odwołującego się do niej. Można również przygotować więcej plików z różnymi definicjami tych samych klas, a wybór określonego designu pozostawić użytkownikowi. Wyjaśnia to dlaczego mówi się o arkuszach stylów. Kaskadowość natomiast polega na tym, że atrybuty każdemu obiektowi można przypisywać w wielu miejscach, a w razie powielenia największe znaczenie mają te położone „najbliżej”. Tak więc odwołania do klasy nie będą miały znaczenia jeśli w znaczniku obiektu nadaje się mu inny wygląd za pomocą lokalnego określenia *style* [12].

Mimo iż w wykonanej stronie www interfejs nie zajmuje ważnej roli, a co za tym idzie ograniczony został do niezbędnego minimum, wykorzystano CSS do definiowania wyglądu. Wynika to przede wszystkim z konieczności zachowania standardów oraz opisanej powyżej łatwości dokonywania zmian.

3.1.3. PHP

PHP Hypertext Preprocesor to obiektowy i skryptowy język programowania wykorzystywany do tworzenia dynamicznych stron internetowych. Jego użycie było niezbędne do obsługi plików, wielu obliczeń, oraz innych dynamicznych zdażeń. Skrypty tego języka wykonuje się po stronie serwera, obarczając przeglądarkę jedynie wyświetlaniem kodu wynikowego. Pierwszą mało funkcjonalną wersję opracował w 1994 roku duński informatyk Rasmus Lerdorf. Od tego czasu popularność oraz społeczność rozwijająca PHP systematycznie rośnie. Dziś ciężko wyobrazić sobie jakąkolwiek stronę, czy witrynę, która nie

wykorzystywałyaby PHP lub nowszych odpowiedników. Najnowsza wersja to PHP 5 i z niej skorzystano podczas pisania aplikacji. Aktualnie trwają prace nad wersją 6.

Język ten posiada wiele zalet, które z pewnością stanowiły o jego dynamicznym rozwoju. Nie bez znaczenia jest udostępnianie go na zasadach otwartego oprogramowania Open Source. Liczna społeczność gwarantuje również ciągły rozwój i powstawanie nowych modyfikacji. PHP jest językiem interpretowanym, a więc brak potrzeby zastosowania kompilatora zapewnia jego przenośność, czyli zgodność z wieloma systemami operacyjnymi. Minusem tego jest nieco wolniejsze działanie. Sama składnia przypomina język C, chyba obecnie najbardziej popularny język programowania. Pierwszą rzucającą się w oczy różnicą w stosunku do C jest brak potrzeby deklarowania typów zmiennych. PHP nie obsługuje również wskaźników jak i utrudnione są działania bezpośrednio na bitach. Skrypty pisane w PHP są zagnieżdżane w HTML zatem dodawanie ich jest bardzo łatwe, jeśli tylko posiada się dostęp do plików serwera [3].

Mimo możliwości wykorzystania innych środowisk do tych samych celów (np.: ASP.NET, JSP, czy MSP), zastosowanie części z nich ogranicza linuksowy system serwera. Ponadto autor zdecydował się na PHP ze względu na jego największą popularność, łatwość zastosowania wynikającą m.in. z dużego podobieństwa do języka C oraz wcześniejszej znajomości.

3.1.4. JavaScript

Pewne dynamiczne procesy, takie jak dodawanie obiektów formularza, musiały zostać wykonane po stronie klienta. Dzięki temu m.in. ograniczono niepotrzebne przesyłanie wielu danych pomiędzy klientem-serwerem i odwrotnie. W związku z tym wykorzystano do tego celu język *JavaScript*, który jest językiem obiektowym, interpretowanym, najczęściej działającym po stronie klienta. Wykorzystuje się go często do pisania skryptów, które reagują na zdarzenia wywołane przez system lub użytkownika [5]. Najczęściej są to: zmiana wyglądu dokumentu w zależności od wyboru użytkownika lub innych warunków, różnego rodzaju kalkulatory, weryfikacja danych w formularzach, tworzenie adresów URL na podstawie zmiennych itd. Ważnym faktem jest, że dzięki temu sieć i serwery mogą zostać odciążone od wielu zadań, które wykonane są po stronie klienta.

JavaScript jest zintegrowana z HTML, w związku z tym skrypty ładowane są w momencie odświeżania strony. Można je umieszczać w tym samym lub innym pliku. Sposób ich wykonania zależy od tego jak powiąże się go z HTML. Uruchomienie pewnej funkcji może być np. powiązane z przyciskiem formularza.

3.1.5. Octave

Jako, że niezbędne są obliczenia na wielowymiarowych macierzach, wykorzystano wspierające je środowisko *Octave*. Jest to program do obliczeń numerycznych jeden z darmowych odpowiedników Matlab, działający zarówno w systemach windowsowych jak i linuxowych i zarówno jak swój prekursor posiada wiele możliwości. Po doinstalowaniu nakładki graficznej, umożliwia również generować i wyświetlać wiele rodzajów wykresów. Z punktu widzenia pisanej aplikacji najbardziej istotny był jednak fakt, że oprogramowanie to może działać w trybie wiersza poleceń systemu Linuks, w którym działa serwer. Dzięki temu bez większych problemów i opóźnień, na serwerze mogą być wykonywane obliczenia i algorytmy z wykorzystaniem macierzy. Prace nad językiem i aplikacją Octave rozpoczęły się wcześniej bo już w 1988 r., pierwsza stabilna wersja wyszła 6 lat później¹.

3.2. Elementy aplikacji

W tej sekcji opisano wykonanie kolejnych części składających się na aplikację. Przedstawiono je zarówno pod kontem pisanego programu jak i funkcjonalności.

3.2.1. Wprowadzanie danych

Część interfejsu odpowiedzialną za obsługę wprowadzanych przez użytkownika danych, podzielono na dwie zasadnicze części:

- (1) ręczną - w postaci formularza;
- (2) automatyczną - jako obsługę danych przekazywanych w plikach.

3.2.1.1. Formularz. Ręcznego wprowadzenia danych możemy dokonać wybierając przycisk *Data generation* na głównej stronie aplikacji. Jak zostało opisane w rozdziale 2.4.2 dane, które są potrzebne do wyznaczenia aktywności na podstawie równania Nikolskiego-Eisenmana są następujące:

- E : różnica potencjałów mierzona pomiędzy elektrodą pomiarową, a elektrodą odniesienia, mV;
- E° : napięcie odniesienia elektrody, mV;
- S : współczynnik Nernsta, mV;
- K^{Pot} : współczynnik selektywności;
- a_0 : granica wykrywalności;
- z : ładunek jonu.

¹ Źródło: <http://www.gnu.org/software/octave/about.html>, dostęp: Czerwiec 2010.

Ponadto jak wykazano w rozdziale 2.6.2.2 należy uwzględnić niepewności wszystkich tych danych wejściowych. Dlatego formularz na dane wprowadzane przez użytkownika powinien być dwa razy większy. Jeżeli dokonuje się pomiaru wieloskładnikowego, a to jest głównym celem tej aplikacji, liczba potrzebnych danych rośnie. Jak wykazano w rozdziale 2.5 jest ona zależna zarówno od liczby jonów jak i elektronów. Formularz w HTML został dodany następującym tagiem:

```
1 <form action="dane.php" method="POST" name="formularz1">
```

atrybut *action* zawiera nazwę skryptu wywołanego po przesłaniu zawartości formularza. *Method* natomiast decyduje o sposobie przekazania danych. Rozróżnia się dwie metody: GET i POST.

Pierwszą z nich charakteryzuje fakt, że wszystkie zmienne formularza dodawane są w odnośniku po nazwie skryptu wykonującego, oddzielane pomiędzy sobą są znakiem *&*. W naszym przypadku część adresu skryptu mogłaby wyglądać następująco:

```
1 dane.php?ilosc_elektrod=3&ilosc_jonow=2&...
```

Ilość przesyłanych danych jest zatem ograniczona maksymalną długością adresu.

W związku z powyższym wykorzystano metodę POST. Jest ona dużo bardziej bezpieczna, ponieważ nie można podejrzec przesyłanych informacji w pasku adresu. Ponadto nie posiada związanej z tym wady w postaci ograniczenia wielkości przekazywanych danych, co w opisywanym przypadku przy wielowymiarowych tablicach ma największe znaczenie. Dlatego ta metoda została wykorzystana w każdym formularzu znajdującym się na opisywanej stronie.

Wnętrze formularza zostało podzielone na poszczególne fragmenty, z których każdy dotyczy innej zmiennej. Dokonano tego za pomocą znaczników `<div></div>`. Każdy taki fragment posiada dynamicznie wytworzoną, odpowiednią ilość elementów *input* typu *text*. Są to pola edycji, do których można wpisywać poszczególne wartości danych wejściowych. Początkowo jednak formularz zawiera jedynie dwa pola edycji, umożliwiające wprowadzenie liczby jonów oraz liczby elektrod. W zależności od wpisywanych ww. wartości w dynamiczny sposób dodawane są kolejne pola, umożliwiające wprowadzenie wszystkich potrzebnych danych. Za dynamiczne dodawanie elementów odpowiada funkcja napisana w języku JavaScript.

Wykorzystaną technikę najczęściej opisuje się jako DOM, czyli *Document Object Model*, co oznacza obiektowy model dokumentu. Jest to taki sposób przedstawiania dokumentu, w którym w łatwy sposób można odnajdywać, zmieniać, usuwać i dodawać poszczególne elementy. Należy zaznaczyć, że model ten jest niezależny od platformy i języka programowania. Jednakże jego standaryzacja obejmuje tylko języki Java i JavaScript, jak już nadmieniono w tworzeniu aplikacji wykorzystano ten drugi.

Pierwszym przykładem wykorzystania tego języka jest pobranie wartości z określonych obiektów. W kodzie poniżej pobierane są wartości liczb elektrod i jonów oraz przypisywane do odpowiednich zmiennych:

```
1 var iloscElektrod=document.forms.formularz1.lelektrod.value;
2 var iloscJonow=document.forms.formularz1.ljonow.value;
```

Dzięki JavaScript możliwy jest łatwy dostęp do wartości atrybutu *value* obiektu *lelektrod*, który znajduje się w obiekcie *formularz1* należącym do typu *forms* wewnątrz strony, która oznaczana jest jako *document*. Ten przykład obrazuje jak można odwoływać się do poszczególnych węzłów strony i ich zagnieżdżeń. W powyższym przykładzie odczytano zatem pola formularza i przypisano ich wartości do zmiennych *iloscElektrod* i *iloscJonow*. Należy podkreślić, że metoda ta pozwala nam dokonać tego przed wysłaniem metodą POST.

Kolejnym krokiem jest wygenerowanie na podstawie wartości powyższych zmiennych odpowiedniego kształtu formularza. Napisano do tego odpowiednią funkcję. Za inicjowanie kolejnych elementów odpowiada polecenie *document.createElement()*, a za ustawianie odpowiednich jego parametrów *setAttribute()*. Przykład zastosowania obrazuje poniższy listing:

```
1 var a = document.createElement('input');
2   a.setAttribute('type', 'text');
3   a.setAttribute('name', name_parametr);
4   a.setAttribute('size', '5');
```

W pierwszej linii wprowadzana jest zmienna *a* i przypisuje się jej nowo kreowany element *input*. W następnych liniach ustawiane są atrybuty: typ danych (*type*) jako *text*, nazwa (*name*) jako *name_parametr* oraz rozmiar (*size*) jako 5, czyli mieszczący 5 znaków. Wartość *name_parametr* nie znajduje się wewnątrz apostrofów ponieważ, jest zmienną przekazywaną przez funkcję i dzięki niej w różnym miejscu formularza mogą powstawać elementy *input* o różnych nazwach.

Po zainicjowaniu elementu należy go jeszcze dodać do dokumentu. Wykorzystano w tym celu polecenie *appendChild()*. Działa ono w ten sposób, że element podany jako argument zostaje dołączony jako dziecko na koniec elementu, w którym funkcja ta została wywołana. Aby wywołać to polecenie w odpowiednim miejscu, wykorzystano jego wyszukiwanie po identyfikatorze. Służy do tego polecenie *getElementById()*. Całość dużo lepiej obrazuje poniższy przykład:

```
1 document.getElementById(ID_parametr).appendChild(a);
```

W obiekcie o identyfikatorze *ID_parametr* dodajemy wcześniej utworzony element przypisany do zmiennej *a*. Jako, że *ID_parametr* jest zmienną przekazywaną przez funkcję, w zależności od jej wartości możemy dodawać raz utworzony element *a* w wielu miejscach formularza.

Powyżej przedstawiono podstawowe elementy wykorzystywane przy tworzeniu nowych obiektów. Obok nich napisano i wykorzystano wiele innych funkcji, które pełnią takie zadania jak: obliczanie odpowiedniej ilości potrzebnych elementów, przypisywanie im odpowiednich nazw, identyfikatorów i wartości domyślnych, usuwanie niepotrzebnych elementów. Listing całego, napisanego w tym celu skryptu można znaleźć w dodatku A.

Aby zobaczyć kiedy dokonywana jest zmiana ilości elementów należy na chwilę powrócić do HTML. Poniżej przedstawiono początek naszego opisanego już formularza:

```
1 <form action="dane.php" method="POST" name="formularz1">
2 number of Electrodes: <input type="text" name="lelektrod" size="8" maxlength="2" value="1"
   ↳onKeyUp="if(isNaN(this.value)) {this.value = '1'};dodajPole();"><br/>
3 number of Ions: <input type="text" name="ljonow" size="8" maxlength="2" value="1"
   ↳onKeyUp="if(isNaN(this.value)) {this.value = '1'};dodajPole();"><br/>
```

W dwóch elementach *input*, które początkowo jako jedyne znajdują się w formularzu, znajduje się atrybut *onKeyUp*. Definiuje on reakcję jaka zostanie wykonana po wpisaniu czegokolwiek w określone pole. Jedną z nich jest właśnie wywołanie funkcji *dodajPole()*, która jest odpowiedzialna za wcześniej opisywane tworzenie nowych elementów. Wcześniej jednak wykonywany jest następujący kod:

```
1 onKeyUp="if(isNaN(this.value)){this.value='1'}"
```

Dzięki niemu za każdym razem kiedy użytkownik dokona edycji pól odpowiedzialnych za ilość elektrod i ilość jonów, sprawdzane jest czy wprowadzone przez niego dane to liczby. W przeciwnym wypadku wpisywana jest wartość taka sama jak domyślna czyli 1. Napisano również inne zabezpieczenia już wewnątrz samej funkcji *dodajPole()*. Ograniczono w nich maksymalną liczbę elektrod i jonów do 10. Po przekroczeniu tej wartości, czyli po wpisaniu większej liczby przez użytkownika zostaje ona zamieniona na maksymalnie możliwą, czyli 10. Z rozdziału 2.5 wynika również, że liczba jonów nie może być większa od liczby elektrod. To zabezpieczono w analogiczny sposób i użytkownik wpisując liczbę jonów większą od wprowadzonej liczby elektronów uzyska co najwyżej jej równowartość.

Przy edycji dwóch pól rozmiar i ilość elementów natychmiast zostaje zmieniona (jeśli to konieczne). Tworzy to interakcję z użytkownikiem, który w natychmiastowy sposób zostaje poinformowany o tym jakich i w jakiej liczbie danych potrzebuje, aby obliczyć aktywności w modelu o określonej przez niego liczbie elektrod i jonów.

Ostatnim istotnym elementem opisywanego formularza jest przycisk potwierdzający. Najczęściej wykorzystuje się go do wywołania funkcji *submit()*. W danym przypadku postąpiono jednak inaczej:

```
1 <input type="button" value="OK" onclick="sprawdzanieGeneracja()"><br/>
```

Atrybut *onclick* przycisku, czyli po wciśnięciu, wywołuje funkcję *sprawdzanieGeneracja()*. Jest to funkcja napisana w JavaScript, która ma za zadanie sprawdzenie poprawności wprowadzonych danych. Następuje to w momencie w którym użytkownik dokonał potwierdzenia, ale zanim zawartość formularza zostanie przesłana. Jeżeli, któreś pole zawiera nieodpowiednią wartość, poniżej formularza w opisywany już wcześniej dynamiczny sposób pojawia się stosowny komunikat. W przeciwnym razie, jeśli wszystkie pola zawierają poprawne dane, wykonywana jest metoda *submit()*, i zawartość formularza trafia do podanego w nim skryptu PHP, czyli w tym przypadku do *dane.php*.

Na Rys. 3.1 pokazano początkowy wygląd formularza. Mimo, że kod HTML początkowo zawiera tylko dwa elementy *input* formularz zawiera już pewne pola do których można wprowadzać dane. Wynika to z tego, że pierwsze wywołanie funkcji *dodajPole()*, odpowiadającej za generację pól formularza, odbywa się automatycznie zaraz po załadowaniu sekcji *<body>*. Odbywa się to bez interakcji z użytkownikiem, przy wartościach domyślnych dla ilości elektrod i jonów równych 1. Związku z tym początkowa wielkość formularza zawsze jest taka sama i łatwo ją zmienić wybierając inne wartości domyślne dla pól *lelektrod* i *ljonow*. Na Rys. 3.1 przedstawiono również, że ładunki jonów *charges* możemy ustalić za pomocą listy określonego wyboru *select*. Wynika to z faktu, że ładunki przyjmują wartości całkowite ze zbioru liczb $(-3, -2, -1, +1, +2, +3)$ i zastosowanie ograniczonego pola wyboru zapobiega ewentualnym, dodatkowym pomyłkom użytkownika.

Na Rys. 3.2 pokazano zmianę ilości elementów *input* w odpowiedzi na wprowadzone dane przez użytkownika. Ustawienie liczby elektrod jako 3 sprawia, że do wypełnienia takich zmiennych jak pomiar SEM (*E*), napięcie odniesienia (*E0*), stała Nernsta (*S*), granica wykrywalności (*a0*) oraz ich niepewności (*u(E)*, *u(E0)*, *u(S)*, *u(a0)*) potrzeba wektorów 3×1 wymiarowych. Ponadto liczba 2 jonów wymusza podania macierzy 3×2 jako współczynniki selektywności (*K*) i ich niepewności (*u(K)*) oraz 2 ładunków (*z1,z2*). Dokonano również próby wysłania zawartości formularza, wywołało to wcześniej omówioną funkcję *sprawdzanieGeneracja()*, która jednak zwróciła błąd. Informuje o tym wyraźny komunikat sugerujący zmianę określonej zmiennej. W tym przypadku jest to współczynnik selektywności $K_{1,1}^{Pot}$ dotyczący głównego jonu danej elektrody i aby była ona na niego selektywna musi wynosić 1.

number of Electrodes:

number of Ions:

E: u(E): E0: u(E0): S: u(S): K: u(K): a0: u(a0):

80	0.1	100	0.7	59	0.4	1	0	5e-16	2.7e-18
----	-----	-----	-----	----	-----	---	---	-------	---------

Charges:

z1 :

This may take several minutes.
 You can define maximum solving time, but it can affect an uncertain of the result.
 For accurate result and no time limit put "0".

max. time: min

RYSUNEK 3.1. Początkowy wygląd formularza.

number of Electrodes:

number of Ions:

E: u(E): E0: u(E0): S: u(S): K: u(K): a0: u(a0):

80	0.1	100	0.7	59	0.4	2	0.0001	0	1e-15	5e-16	2.7e-18
80	0.1	100	0.7	59	0.4	0.0001	1	1e-15	0	5e-16	2.7e-18
80	0.1	100	0.7	59	0.4	1	0.0001	0	1e-15	5e-16	2.7e-18

Charges:

z1 :

z2 :

value of this parameters should be equal 1: Kpot 1,1;

This may take several minutes.
 You can define maximum solving time, but it can affect an uncertain of the result.
 For accurate result and no time limit put "0".

max. time: min

RYSUNEK 3.2. Przykładowy wygląd formularza.

3.2.1.2. Upload plików. Zamiast ręcznego generowania danych możemy przesłać je za pomocą plików wybierając *Data upload* na głównej stronie aplikacji. Jest to możliwe bowiem część wprowadzanych wielkości może pochodzić z aplikacji *SiCaPSA* służącej do wyznaczania współczynników z kalibracji. Można zatem przysłać posiadane wielkości wprost

z plików znajdujących się na twardym dysku użytkownika. Należało jednak dostosować ich strukturę i format odczytu do tej uzyskanej na wyjściu aplikacji *SiCaPSA*. Dokonano koniecznego podziału na trzy odrębne pliki:

- (1) mierzone różnice potencjałów, ich niepewności oraz ładunki poszczególnych jonów ($E, u(E), Z$);
- (2) pozostałe parametry (E°, S, K^{Pot}, a_0);
- (3) niepewności tych parametrów ($u(E^\circ), u(S), u(K^{Pot}), u(a_0)$).

Jak można wywnioskować pierwszy plik dotyczy bezpośrednio pomiaru i niezbędnej wiedzy o badanym roztworze. Pozostałe dwa wprowadzają parametry uzyskane podczas kalibracji. Pliki powinny zawierać tylko same dane oddzielone od siebie znakami pustymi, np. spacją. Sugerowanym formatem, jest format tekstowy *.txt*, jednakże nie jest on niezbędny. Wymiary macierzy wewnątrz plików jak i wielkość ich samych różni się w zależności od stosowanego modelu pomiarowego, które charakteryzuje liczba elektrod i liczba jonów. Te dwie wartości nie muszą być zatem podawane, ich odczytanie następuje na podstawie pierwszego przesłanego pliku. Dokładniej rzecz ujmując po ilości wprowadzonych pomiarów SEM, aplikacja rozpoznaje ilość zastosowanych elektrod. Natomiast po ilości podanych ładunków ilość badanych jonów. Na podstawie tych dwóch wielkości oceniana jest również poprawność przesłanych następnie pozostałych plików zawierających parametry i ich niepewności. Szczegóły wyglądu poszczególnych plików użytkownik może odnaleźć w pomocy aplikacji znajdującej się pod zakładką *Help*.

W celu utworzenia interfejsu umożliwiającego użytkownikowi przesłanie plików, posłużono się po raz kolejny formularzem, którego część wygląda następująco:

```
1 <form id="maszForm" enctype="multipart/form-data" action="uploadedE.php" method="post">
2 <input type="hidden" name="MAX_FILE_SIZE" value="1000000"/>
3 Upload the file with <b>Voltages</b>: <br />
4 <input name="pomiarEfile" type="file" />
5 <br/>
6 <input type="submit" value="upload"/>
```

W tagu formularza wykorzystano atrybut, którego w przypadku ręcznego wprowadzania danych nie opisywano - był on pozostawiony domyślnie. *Enctype* definiuje sposób kodowania danych, dla przesłania pliku konieczne jest ustawienie jego wartości jako *multipart/form-data*. W kolejnej linii znajduje się element *input*, który ogranicza maksymalny rozmiar przesyłanego pliku. Świadczy o tym jego nazwa *name* oraz wartość *value* wyrażona w bajtach. W tym przypadku ograniczono zatem wielkość do 1 MB. Typ *hidden* oznacza po prostu, że znacznik ten jest ukryty. Kolejny element typu *file* to po prostu miejsce na podanie ścieżki przesyłanego pliku, któremu przyznano roboczą nazwę *pomiarEfile*. Ostatnim pokazanym tutaj elementem jest przycisk służący do przekazania zawartości

formularza. W tym celu wystarczyło utworzyć element typu *submit*. Jest to trochę inny sposób wywołania zdarzenia niż opisywany w rozdziale 3.2.1.1 w opisie formularza do ręcznego wprowadzenia danych. Wynika to z faktu, że tym razem nie da się sprawdzić poprawności przekazywanych wielkości przed ich przesłaniem w postaci plików. Nie można bowiem badać zawartości plików na twardym dysku użytkownika.

Powyższy przykład formularza został zatem wykorzystany dla każdego przesyłanego pliku osobno. Nie dokonano zestawienia tego w jednym miejscu strony. Zamiast tego użytkownik systematycznie jest proszony o kolejne wybieranie plików, jeśli wcześniej przesłane są poprawne pod względem struktury oraz są ze sobą zgodne pod względem opisywanego modelu. Wydaje się, że takie postępowanie jest czytelniejsze przede wszystkim dlatego, że natychmiast informuje użytkownika o wadliwości przesłanych danych i prosi o korektę. Dopiero po wprowadzeniu poprawnego pliku istnieje możliwość przesłania kolejnego.

Nie można zbadać struktury danych przed ich przesłaniem. Zamiast tworzenia odpowiedniej funkcji w JavaScript wykorzystano polecenia PHP i Octave, na dwóch różnych poziomach zabezpieczeń.

Jako pierwsze na poziomie PHP wykorzystano funkcję *is_uploaded_file()*. Zwraca ona wartość *true* jeżeli plik o nazwie, która jest parametrem tej funkcji został poprawnie uploadowany. Jeżeli nie został odebrany od użytkownika zwraca wartość *false*. Zabezpieczenie to jest również istotne, aby złośliwy użytkownik w wyniku źle napisanego skryptu nie miał dostępu do wszystkich plików, do których prawo odczytu posiada użytkownik na jakim pracuje serwer www. Jeżeli przesłanie pliku nastąpi bez problemu następuje jego przeniesienie w docelowe położenie. Wykorzystaną w tym przypadku funkcją jest *move_uploaded_file()*. Jeżeli wszystko przebiegnie prawidłowo zwraca ona wartość *true*, w przeciwnym razie zwraca *false*. W przypadku zwrócenia wartości *false* przez którąkolwiek z powyższych funkcji, następuje przerwanie wykonywania dalszego skryptu, użytkownik informowany jest o wystąpieniu błędu, po czym po raz kolejny proszony jest o przesłanie prawidłowego pliku.

Jeżeli udało się prawidłowo umieścić plik na serwerze, kolejnym zabezpieczeniem jest sprawdzenie jego zawartości odpowiednim skryptem napisanym w środowisku Octave. Poprzez jego użycie weryfikuje się, czy zapisane dane są macierzami oraz czy posiadają odpowiednie rozmiary. Skrypt sprawdzający zawartość pierwszego przysyłanego pliku, w przypadku jego poprawności, jak zostało już nadmienione, na jego podstawie wyznacza liczbę elektrod i jonów użytych w pomiarze. Wartości te wykorzystywane są następnie do badania struktury i rozmiarów macierzy przesyłanych w kolejnych plikach. Na każdym etapie użytkownik informowany jest o rezultatach. Jeżeli przesyłane dane są prawidłowe, zostaje zwrócona podstawowa wartość je charakteryzująca (np. ilość elektrod, jonów, parametrów), pojawia się komunikat potwierdzający pozytywny wynik sprawdzania oraz użytkownik proszony jest o wybranie kolejnego do przesłania pliku. W przypadku kiedy

ostatni plik zostanie przekazany prawidłowo, po zaakceptowaniu aplikacja przechodzi do następnego etapu. Jeżeli jednak dane okażą się nieprawidłowe, podobnie jak w przypadku zabezpieczeń PHP, użytkownik zostaje poinformowany o błędzie oraz poproszony o wybranie prawidłowego pliku.

Powyżej opisano dwa sposoby w jakie można wprowadzić wartości niezbędne do wyznaczenia aktywności i ich niepewności. Oba w różny sposób prowadzą do tego samego rezultatu - utworzenia na serwerze plików z danymi, po jednym dla każdego wektora i macierzy. W przypadku ręcznego wprowadzenia danych następuje to po przesłaniu formularza, kiedy uruchamiany jest skrypt zapisujący wartości pól do odpowiednik plików. W drugim przypadku podział danych odbywa się w skryptach Octave, po sprawdzeniu z pozytywnym rezultatem zawartości przesyłanych przez użytkownika danych. Dzięki temu na serwerze powstają pliki *txt*, po jednym dla każdego rodzaju parametrów, do których istnieją sposoby bardzo łatwego dostępu i odpowiedniego odczytu zarówno z poziomu PHP jak i Octave.

3.2.2. Komunikacja i sesje

Poza przekazywaniem danych typowym dla większości stron www, takim jak wysyłanie żądań lub wczytywanie zawartości stron, w przypadku danej aplikacji możemy wyróżnić dwa istotne poziomy komunikacji. Pierwszy z nich, wcześniej prawie w całości opisany, dotyczy obliczeń i przekazywania danych pomiarowych klient-serwer-klient. Dzięki PHP dokonać można zapisu odpowiednich wartości w plikach serwera. Następnie po dokonaniu obliczeń wyniki są pobierane i wyświetlane również z wykorzystaniem skryptów napisanych w tym języku.

Sytuacją rzadziej wykorzystywaną we wszelkiego rodzaju kalkulatorach i aplikacjach jest użycie środowiska służącego stricte do obliczeń numerycznych i nie zaliczającego się do szeroko rozumianych technologii internetowych. W opisywanej aplikacji tak wykorzystywanym programem jest Octave, którego zastosowanie wymaga odpowiedniego przekazywania danych pomiędzy tym środowiskiem a skryptami PHP. Jako, że ten ostatni jest językiem wykonywanym po stronie serwera, nie tyle zostanie opisana rzeczywista komunikacja pomiędzy różnymi nośnikami danych, jak uruchamianie skryptów i odbieranie ich wyników z Octave przez PHP.

W związku z tym, że serwer działa na linuxie oraz Octave działa w trybie wiersza poleceń wykonanie tego jest dość proste. Aby zastosować polecenia systemowe w takiej samej formie jak przy wpisywaniu w konsoli wystarczy użyć w PHP funkcji *system()*. Dzięki niej możemy właśnie uruchomić program zewnętrzny i przekazać do niego parametry. Funkcja ta zwraca również ostatnią linię wynikową zastosowanej komendy, jeśli ona oczywiście wystąpi. Dla lepszego zobrazowania warto przeanalizować następujący przykład:

```
1 system("octave -qf /var/www/poten/glowny.m ".$path)
```

Octave jest poleceniem uruchamiającym ten program. Flaga *-qf* zapobiega m.in. przed wyświetlaniem wszelkich komunikatów startowych, o błędach itd. Jest to istotne przy użyciu funkcji *system*, która zwraca nam ostatnią wyświetlaną linię. Następnie podana jest ścieżka uruchamianego skryptu, w tym przypadku jest to *glowny.m*. Jako ostatnie możemy podawać argumenty. W tym przypadku przesłano jeden w postaci zmiennej PHP o nazwie *\$path*. Niesie ona ze sobą informację o położeniu katalogu do którego wcześniej zapisane zostały dane wejściowe oraz do którego chcemy zapisywać wyniki uruchamianego skryptu. Po stronie kodu napisanego w Octave zmienna ta odbierana jest w następujący sposób:

```
1 sciezka=argv(){1,1};
```

Zmiennej *sciezka* jest zatem przypisana wartość zmiennej *\$path* i możemy ją wykorzystywać wewnątrz pliku *glowny.m*. Plik ten zawiera najważniejsze obliczenia po stronie Octave i cały jego listing został zamieszczony w dodatku A na końcu dokumentu. Po wykonaniu jego kodu funkcja *system* zwraca jego wynik. Nie jest to jednak funkcjonalne rozwiązanie przy większej ilości danych, dlatego można przekazać je również za pomocą plików.

Zostało powiedziane, że dane muszą być zapisywane w określonej lokacji, której adres powinien być znany zarówno po stronie PHP i Octave, aby można było dowolnie zapisywać i odczytywać zawartość plików w tym katalogu. Jest to niezbędne do tego by z aplikacji mogło korzystać więcej użytkowników jednocześnie, a ich dane nie kolidowały ze sobą. W PHP do tego celu wykorzystuje się tzw. *sesje*.

W potocznym rozumieniu sesja to czas, w którym użytkownik przegląda daną stronę. W PHP jest to obiekt, który może posiadać wiele zmiennych charakteryzujących użytkownika, zapisywanych w tablicy *\$_SESSION*. Najważniejszą z nich jest *PHPSESSID*, która zawiera unikatowy ciąg znaków różny dla każdej sesji. Przesyłany jest ona do użytkownika przy wykorzystaniu ciasteczek i dzięki niej każda podstrona może rozróżnić osoby korzystające z aplikacji. Jako, że dane w postaci plików każdego użytkownika zapisywane są w odrębnym katalogu, możemy go od razu poprzez nazwę powiązać z sesją. Dzięki temu każdy folder będzie miał automatycznie generowaną nazwę, taką samą jak ID danej sesji. Poniżej zestawiono najistotniejsze funkcje i polecenia, które zostały wykorzystane i które bezpośrednio lub pośrednio związane są z sesją:

```
1 session_start();
2 session_id();
3 $path='/tmp/potencjometria/'. session_id() .'/';
4 system ("mkdir /tmp/potencjometria/");
5 system ("mkdir ".$path);
```

```
6 system ("rm -R ".$path);  
7 session_destroy();
```

Polecenie *session_start()* służy do rozpoczęcia sesji lub odtworzenia stanu już istniejącej jeżeli wcześniej nastąpiło rozpoczęcie. Musi ona znajdować się na początku każdego pliku. *Session_id()* podana bez argumentu zwraca unikatowy klucz, który określa sesję. W 3 linii kodu następuje utworzenie zmiennej *\$path* która zawiera ścieżkę do katalogu o tej samej nazwie co ID sesji i znajduje się wewnątrz katalogu na pliki tymczasowe *tmp*. W kolejnej linii poleceniem systemowym stworzymy w istniejącym katalogu na pliki tymczasowe podkatalog *potencjometria*, jeżeli wcześniej nie został już utworzony. Następnie wewnątrz stworzymy kolejny podkatalog dla danego użytkownika-sesji o nazwie takiej samej jak nazwa klucza sesji. W lini 6 następuje usunięcie całego katalogu, polecenie to jest wykorzystywane po wygenerowaniu przez naszą aplikację odpowiedzi, kiedy dane przesłane przez użytkownika nie będą już potrzebne. Jest to istotne aby nie obciążać serwera zbędnymi plikami, których gromadzenie nie jest celem napisanej strony. Po usunięciu plików i katalogu użytkownika można również usunąć bieżącą sesję za pomocą *session_destroy()*.

3.2.3. Wyznaczanie aktywności

Aby oszacować aktywność jonów niezbędne jest rozwiązanie układu równań nieliniowych (rozdział 2.5). W Octave można tego dokonać wykorzystując napisaną pierwotnie przez J.W. Eatona funkcję *fsolve()*, która implementuje metodę będącą modyfikacją metody Newtona². Wyznacza ona pierwiastki z równań przekształconych tak, aby po jednej ze stron posiadały zero. Jest to metoda lokalna, która do poprawnego działania potrzebuje początkowego wektora wartości, w pobliżu których następuje wyznaczanie rozwiązania. Przykład wywołania tej funkcji wygląda następująco:

```
1 A=fsolve("model",start);
```

Funkcję wywołuje się z conajmniej dwoma parametrami. W tym przypadku, pierwszym z nich jest „*model*”, który odwołuje się do układu równań zapisanego w pliku *model.m*. Drugim jest wektor *start* zawierający wartości początkowe, w pobliżu których wyznaczane jest rozwiązanie. Rozwiązania układu równań przypisywane są wektorowi *A*. Wymiary wektora *start* i *A* muszą być takie same i zawierać tyle elementów, ile układ ma niewiadomych.

² Źródło: <http://velveeta.che.wisc.edu/cgi-bin/hgwebdir.cgi/octave/file/tip/scripts/optimization/fsolve.m?rev=HEAD&content-type=text/plain>, dostęp: Wrzesień 2010.

Aby wykorzystać funkcję *fsolve()* należy odpowiednio przebudować układ równań (4). W tym celu dokonano odpowiednich przekształceń tak, aby zgrupować wszystkie wyrazy wolne oraz całość przyrównać do zera:

$$\begin{cases} a_1 + \sum_{j \neq 1} K_{1,j}^{\text{Pot}} \cdot a_j^{z_1/z_j} + a_{0,1} - 10^{\left(\frac{E_1 - E_1^0}{S_1}\right)} = 0 \\ a_2 + \sum_{j \neq 2} K_{2,j}^{\text{Pot}} \cdot a_j^{z_2/z_j} + a_{0,2} - 10^{\left(\frac{E_2 - E_2^0}{S_2}\right)} = 0 \\ \vdots \\ a_n + \sum_{j \neq n} K_{n,j}^{\text{Pot}} \cdot a_j^{z_n/z_j} + a_{0,n} - 10^{\left(\frac{E_n - E_n^0}{S_n}\right)} = 0 \end{cases} \quad (18)$$

Tak zmieniony układ równań można łatwo zapisać iteracyjnie, czego dokonano w funkcji *model*, której najważniejsza część wygląda następująco:

```
1 function z=model(x)
2 % tworzenie układu równań
3 E=zeros(lelektrod,1);
4 for i=1:lelektrod
5     for j=1:ljonow
6         E(i)=E(i)+K(i,j)*x(j)^(Z(i));
7         if (j==ljonow)
8             E(i)=E(i)-Y(i);
9         end
10    end
11 end
12 z=E;
13 endfunction
```

W linii 6 w zależności od ilości jonów tworzy się sumę $\sum_{j \neq i} K_{j,i}^{\text{Pot}} \cdot a_j^{z_i/z_j}$. Wyrazy wolne w postaci $a_{0,i} - 10^{\left(\frac{E_i - E_i^0}{S_i}\right)}$ dodawane są jako ostatnie w linii 8, gdzie odejmuje się element wektora *Y*, który wcześniej wyliczany jest następująco:

```
1 Y=10.^((E-E0)./S)-A0
```

Oprócz modelu istotny jest również dobór wartości początkowych w wektorze *start*. Nie jest to jednak trudne, znany jest bowiem przedział w którym poszukujemy rozwiązania. W skali logarytmicznej z jednej strony ogranicza go granica wykrywalności a_0 , czyli najmniejsza mierzona ilość substancji (aktywność) jaka z przyjętym poziomem ufności może być odróżniona od jej braku [11]. Z drugiej strony w tak przyjętej skali rozwiązanie nie może przekroczyć wartości 1. Podczas wielu obliczeń aktywności zostają od razu

prawidłowo oszacowane, po określeniu punktów startowych jako wektora granic wykrywalności. W przeciwnym wypadku, po odpowiednim zwiększeniu wartości wektora *start*, dokonuje się kolejne wywołanie funkcji *fsolve()* ze zmienionymi wartościami startowymi.

Ocenę poprawności dokonuje się na podstawie wyznaczonej różnicy pomiędzy rzeczywistym pomiarem SEM a symulowanym przy użyciu oszacowanych aktywności. W tym celu wylicza się symulacyjne wartości *E* jako *EE*, podając jako jedno z wejść wektor nowo uzyskanych aktywności *A*:

```
1 EE=model2(E0,S,KK,A0,A,lljonow,ZZ);
```

Funkcja *model2* podobnie jak *model* tworzy iteracyjnie układ równań. Różnica polega na tym, że wyjściem jest nowo wyliczana wartość różnicy potencjałów:

```
1 function w=model2(E0,S,KK,A0,A,lljonow,ZZ)
2
3 suma=zeros(lelektrod,1);
4 ##układ równań
5 #suma K*A
6 for i=1:lelektrod
7     for j=1:lljonow
8         suma(i)=suma(i)+KK(i,j)*A(j)^(ZZ(i));
9     end
10 end
11
12 #wyznaczane E
13 w=E0+S.*log10(suma+A0);
14 endfunction;
```

Następnie wyznacza się różnicę pomiędzy rzeczywiście zmierzoną różnicą potencjału a symulowaną (na podstawie wektora obliczonych aktywności *A*):

```
1 deltaE=EE-E;
2 if (abs(deltaE(i,1))>(E(i)/1000))
```

Linia 2 zawiera warunek jaki musi być spełniony, aby po zmianie wartości początkowych dokonać kolejnego wyliczenia aktywności z użyciem funkcji *fsolve*. Mimo iż jako kryterium przyjmuje się $\Delta E > 0.1\%E$, jest on wystarczający na otrzymanie wyników z dużo większą dokładnością. Jest to możliwe ponieważ funkcja *fsolve* gdy zwraca inne wyniki niż oczekiwane, otrzymany na ich podstawie wektor *EE* różni się od *E* dużo bardziej niż o 0.1%. Nawet jeśli funkcja zainicjowana stosunkowo blisko poprawnych rozwiązań od razu zwraca prawidłowe wartości, pewne kryterium porównawcze jest jednak potrzebne.

Wyliczona SEM różni się bowiem w niewielkim stopniu od rzeczywistej, wprowadzonej przez użytkownika błędami numerycznymi.

3.2.4. Wyznaczanie niepewności

Głównym celem aplikacji jest szacowanie niepewności. Jak opisano w rozdziale 2.6.2.2, dokonanie tego w sposób analityczny jest czynnością długotrwałą i trudną. Podejście numeryczne również nie jest pozbawione pewnych niedogodności. Wartość niepewności podaje się z dokładnością do dwóch cyf znaczących [8], więc pewne niewielkie uproszczenia i niedokładności nie mają wpływu. Dużą symplifikacją w toku pisania aplikacji było jednak nie uwzględnienie kowariancji współczynników selektywności (14)(15). W związku z tym wpływ ich niepewności jest uwzględniony tylko częściowo.

Do szacowania niepewności wykorzystano, opisaną w rozdziale 2.6.3, metodę Monte Carlo w następujący sposób. Wykonuje się dużą liczbę testów polegających na obliczeniu aktywności w podobny sposób jak w poprzednim podrozdziale. Dane z których buduje się model obarcza się pewnymi błędami. Do ich kreowania wykorzystuje się zmienną losową o rozkładzie normalnym oraz odchyleniu standardowym równym niepewnościom wprowadzonym przez użytkownika. Po wykonaniu wielu testów, uwzględnia się tak otrzymane rozrzuty i na ich podstawie wylicza się odchylenie standardowe, które odpowiada niepewności standardowej wyjścia. Przy odpowiednio dużej liczbie wykonanych cykli uzyskana wartość niepewności nie odbiega znacząco od wartości uzyskanych analitycznie [17].

Wszystkie te obliczenia dokonywane są w Octave. Poniżej opisano jeden cykl testowy, który wielokrotnie jest powtarzany. W pierwszym kroku dokonywane jest zakłócenie danych wejściowych pewnymi błędami:

```
1 nE=E+randn(1)*uE;  
2 nEO=EO+randn(1)*uEO;  
3 nS=S+randn(1)*uS;  
4 K=K+randn(1)*uK;  
5 nAO=AO+randn(1)*uAO;
```

Funkcja *randn* generuje w pseudolosowy sposób macierz, której elementy mają rozkład normalny o zerowej wartości oczekiwanej i jednostkowym odchyleniu standardowym. Aby zmienić wartość oczekiwaną, należy do wygenerowanych liczb dodać odpowiednią wartość. Natomiast aby zmienić odchylenie standardowe, należy je pomnożyć przez pożądaną liczbę. Argumenty tej funkcji oznaczają rozmiar generowanej macierzy. W przypadku jedynki generowana jest macierz 1×1 , czyli skalar. Zapis $E+\text{randn}(1)*uE$ zwraca więc jedną wartość z rozkładu o wartości oczekiwanej E i odchyleniu standardowym uE . Po wyznaczeniu nowych parametrów modelu następuje na ich podstawie wyliczenie nowej, obciążonej błędem aktywności:

```

1 Y=10.^(nE-nEO)./nS)-nA0;
2 nA=fsolve("model",A);

```

Jak w poprzednim podrozdziale następuje pogrupowanie wyrazów wolnych w postaci wektora Y oraz wykorzystanie funkcji *fsolve*. Jako argumenty podawany jest układ równań, czyli *model* na podstawie którego następują obliczenia oraz wyznaczone wcześniej aktywności A jako punkty startowe. Nie zmieniają się one w każdym cyklu testowym bowiem niewielkie odchyłki parametrów wejściowych nie zmieniają radykalnie nowych wartości aktywności nA , więc z dużym prawdopodobieństwem należy ich szukać w pobliżu A .

Kolejnym krokiem jest obliczenie uzyskanej odchyłki i zapamiętanie jej wartości:

```

1 Macierz(:,test)=nA-A;

```

Parametr *test* jest zmienną inkrementowaną z każdym cyklem testowym. W ten sposób zmienna *Macierz* stopniowo wypełnia się kolejno obliczanymi wektorami odchyłek, które dodawane są jako kolejne kolumny.

Po dokonaniu kilkutysięcy powtórek powyższego cyklu, można na podstawie zgromadzonych danych wyznaczyć odchylenie standardowe, a co za tym idzie oszacować niepewność. Wprowadzono jednak możliwość szybszego przerywania powtarzanych testów. Oprócz liczby powtórzeń zbudowano warunek stopu w oparciu o jakość otrzymywanych wyników. Niepewność aktywności jest jednak szacowywana i nie istnieje możliwość rzetelnej weryfikacji jej wartości, tak jak miało to miejsce w przypadku samej aktywności. Dlatego porównuje się nie tyle jej wartość co zmienność wraz ze wzrostem liczby kolejno szacowanych odchyłek. Zrealizowano to w następujący sposób:

```

1 warunek=mod(test,100);
2
3 if (warunek==1)
4     uA_new=(sqrt(meansq(Macierz')))' ;
5     uA_new=FormatNiep(uA_new);
6
7     if (uA_new==uA_old)
8         suma=suma+1;
9     else
10        suma=0;
11    end
12    uA_old=uA_new;
13    czas=toc;
14 end

```

W 1 linii zmienna *warunek* przyjmuje wartość reszty z dzielenia inkrementowanej zmiennej *test* i 100. Wynika z tego że warunek z linii 3 jest spełniany co 100 cykli testowych. Wtedy w linii 4 funkcją *meansq*, wyliczana jest średnia sumy kwadratów kolejnych odchylek, a funkcją *sqr*t całość zostaje pierwiastkowana. Na podstawie wzoru 7, uzyskuje się w ten sposób odchylenia standardowe dla każdej aktywności.

Następnie tak otrzymane wartości podaje się jako argument dla funkcji *FormatNiep()*, która zwraca je ograniczone do trzech znaczących cyfr. W wyniku jej działania np. wartość 0,01345602 zostaje zamieniona na 0,0134. Dzięki temu działaniu w linii 7, gdzie porównuje się aktualnie oszacowane niepewności i te pochodzące z poprzedniego zadziałania pętli *if*, znaczenie mają tylko najbardziej istotne części tych liczb. Przyjmuje się podawać niepewność ograniczoną tylko do dwóch znaczących cyfr, w związku z tym takie działanie nie niesie ze sobą żadnych pogarszających ostateczny wynik uproszczeń.

Jeżeli 100 kolejnych cykli nie wpłynęło na żadne zmiany 3 znaczących cyfr szacowanych niepewności (linia 7), następuje inkrementacja zmiennej *suma* (linia 8). W przeciwnym wypadku jest ona zerowana (linia 10). Intuicyjna interpretacja tego jest taka, że co 100 losowań (cykli) sprawdzamy czy wyliczane odchylenia znacznie się różnią. Jeśli nie, pewny wskaźnik rośnie, jeśli tak, jest całkowicie zerowany. Jak wynika z rozdziału 2.6.3 wraz ze wzrostem liczby losowanych wartości, rośnie rzetelność opartych na nich równań. Podobnie i w tym przypadku, wyznaczane co 100 cykli odchylenia początkowo charakteryzuje duży rozrzut. Testowanie nie jest jednak zatrzymywane tuż po pierwszej zgodności odchyleń, tylko po kilku następujących po sobie powtórzeniach (inkrementacja zmiennej *suma*). Zabezpiecza to przed przypadkową, jednokrotną równością.

Zastosowanie ograniczenia ilości wykonanych cykli testowych jest niestety konieczne ze względu na stosunkowo długie działanie funkcji *fsolve*. Przy większej ilości równań trwa ono nawet kilka sekund, co przy kilkuset lub kilkutysiącach powtórzeń ma już ogromne znaczenie.

Innym warunkiem zakończenia symulacji jest czas jej trwania. W Octave odmierzać czas można dzięki dwóm poleceniom. Komenda *tic* uruchamia proces odmierzenia czasu, natomiast *toc* zwraca w sekundach czas, który upłynął od rozpoczęcia pomiaru. Tę ostatnią można wywoływać wielokrotnie, a przykład jej wykorzystania można odnaleźć w poprzednim listingu. Co 100 cykli testowych, kiedy wyznaczane jest odchylenie, dokonuje się również przyrównanie do zmiennej *czas* odmierzanego od początku działania algorytmu czasu.

Na zatrzymanie cyklu testowego (czyli pętli obciążania zmiennych błędami, wyznaczania obciążonej aktywności, wyliczania i zapamiętywanie odchylek) mają więc wpływ 3 warunki. Uwzględniane są w następujący sposób:

```
1 if (suma>=2 || czas>=time || test>=5000)
2   losuj=false;
```

```

3 else
4   losuj=true;
5 end

```

Po spełnieniu któregoś z trzech warunków następuje zatrzymanie pętli cyklu, warunek jej trwania *losuj* przyjmuje wartość *false*. Ograniczająca czas zmienna *time* jest wprowadzana przez użytkownika w chwili wypełniania formularza. Jeśli pozostanie domyślna (czyli będzie wynosiła 0), zostanie zmieniona na nieskończoność i czas nie będzie wpływał na zatrzymywanie algorytmu. Wprowadzana przez użytkownika wartość w minutach, jest po wczytywaniu zamieniana na sekundy, w których wyrażona jest również zmienna *czas*.

Ograniczenia wskaźnika *suma*, czy liczby iteracji kolejnych cykli *test* zostały wprowadzone bezpośrednio. Podyktowane to zostało trudnością w zrozumieniu przez potencjalnego użytkownika z jakiego rodzaju ograniczeniami ma do czynienia, chociaż wpływają one bezpośrednio na jakość otrzymanego rozwiązania. Bez zagłębiania się w algorytm i zasadę jego działania, wielkości te nie są tak intuicyjne jak czas oczekiwania na wynik.

Po wyliczeniu aktywności, oszacowaniu jej niepewności standardowej, uwzględnieniu współczynnika rozszerzenia $k = 2$ i obliczeniu niepewności rozszerzonej, rozwiązania te zostają zapisane.

3.2.5. Wyświetlanie wyników

Po dokonaniu obliczeń, plik Octave *główny.m* kończy swoje działanie, a z poziomu PHP dołączony zostaje skrypt *raport.php*, który wyświetla rezultaty działania programu. Parametry wejściowe zostają bezpośrednio wczytane z plików wcześniej zapisanych na dysku serwera, natomiast wyniki wyświetlane są przez uruchomioną funkcję Octave:

```

1 system("octave -qf /var/www/poten/wynik.m ".$path);

```

Skrypt *wynik.m* wczytuje wcześniej zapisane wartości aktywności i niepewności oraz informację o warunku zakończenia pętli testowej w pliku *główny.m*. Dzięki temu, możliwa jest ocena poprawności otrzymanych wyników oraz ich odpowiednie przeskalowanie.

Zgodnie z zaleceniami wartość liczbową niepewności podaje się z najwyżej dwiema cyframi znaczącymi. Wartość wyniku, którego dotyczy niepewność, zaokrągla się tak aby ostatnia znacząca cyfra była na takim samym miejscu, co ostatnia znacząca cyfra niepewności rozszerzonej [8]. Wykorzystano w tym celu właściwości logarytmu dziesiętnego, który po dopełnieniu do liczby naturalnej przyjmuje wartość równą liczbie ilości miejsc po przecinku, po których znajduje się pierwsza cyfra znacząca. Zastosowanie tej wielkości pozawala na odpowiednie przeskalowanie wyników zgodnie z zaleceniami. Dokonuje się

to wewnątrz funkcji *FormatResult()*, która jest wywoływana po wczytaniu danych przez *wynik.m*:

```
1 [a,b]=size(A);
2
3 for i=1:a
4     FormatResult(A(i),uA(i),0,RZ);
5     printf("<br/><br/>");
6 end
```

Dzieje się to tyle razy, ile użyto jonów w modelu. Wartość tę odczytuje się bezpośrednio jako rozmiar wektora A , który zawiera obliczone wcześniej aktywności. Oprócz aktywności i niepewności rozszerzonej (uA) przekazywana jest również zmienna RZ . Zawiera ona informacje na podstawie, których dokonuje się oceny wyników. Uwzględnia tylko opisaną wcześniej zmienną $suma$, lecz istnieje możliwość przekazywania innych wskaźników, takich jak czas, czy liczba iteracji. Jako, że użytkownik ma tylko niewielki wpływ na kryteria stopu, ewentualne wykorzystanie tych wielkości pozostawiono w gestii programisty.

Funkcja *FormatResult()* posiada wielokrotnie powtarzający się fragment:

```
1 if (digits == 1)
2     printf("%10.1f", Value);
3     testA(Value);
4     printf(" +/- %9.1f", Uncert);
5     testNiep(Value,Uncert,RZ);
6
7 endif
```

Oprócz wyświetlania w odpowiednio sformatowany sposób samej aktywności ($Value$) i niepewności rozszerzonej ($Uncert$), zwracane są wartości przez funkcje *testA()* oraz *testNiep()*. Wyświetlają one jeden z trzech możliwych komunikatów: *ok*, *attention* lub *false*, w zależności od tego czy wynik podany jest z dobrym przybliżeniem, pewną niedoskonałością lub nosi znamiona błędne. Rys. 3.3 zawiera przykładowy rezultat aplikacji. Komentarze dotyczące niepewności informują o ryzyku jej pewnej niedokładności. W takiej sytuacji najprawdopodobniej użytkownik zbytnio skrócił czas oczekiwania na wynik.

3.3. Testy

Aby lepiej zobrazować dobrane kryteria oraz ocenić działanie aplikacji, poniżej przedstawiono kilka przykładów jej wywołania. Najczęściej wykorzystano model składający się z trzech elektrod (zwany dalej „modelem domyślnym”), które charakteryzują następujące

results:

Activites:

Ion. No.1 : 0.46 OK +/- 0.07 ATTENTION

Ion. No.2 : 0.079 OK +/- 0.010 ATTENTION

Ion. No.3 : 0.0092 OK +/- 0.0008 ATTENTION

RYSUNEK 3.3. Przykładowo wyświetlone wyniki.

parametry:

	E	$u(E)$	E°	$u(E^\circ)$	S	$u(S)$	K^{Pot}	$u(K^{Pot})$				a_0	$u(a_0)$	
El. nr.1	80	0.1	100	0.7	59	0.4	1	1e-5	1e-5	0	1e-15	1e-15	5e-16	2.7e-18
El. nr.2	80	0.1	100	0.7	59	0.4	1e-5	1	1e-5	1e-15	0	1e-15	5e-16	2.7e-18
El. nr.3	80	0.1	100	0.7	59	0.4	1e-5	1e-5	1	1e-15	1e-15	0	5e-16	2.7e-18

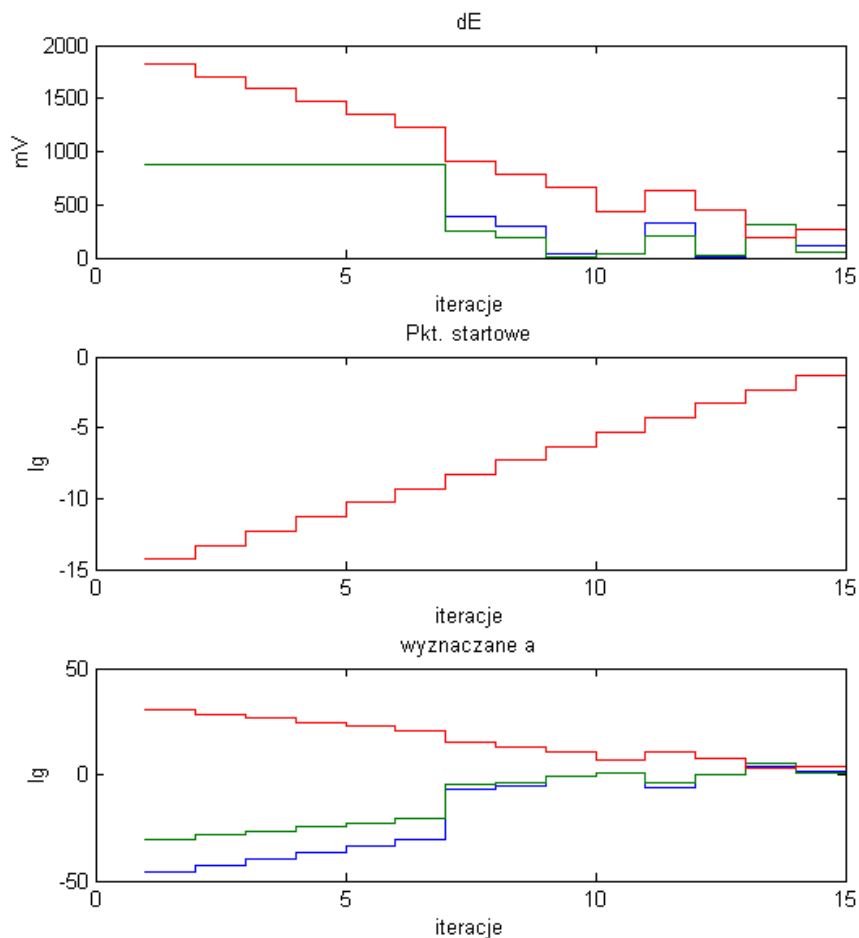
Elektrodom przypisano identyczne parametry i wyniki pomiaru. Dobrano je tak, aby wyznaczana aktywność była duża (ok. 0.1 - w zależności od zastosowanych ładunków), a granica wykrywalności niska (mała wartość to duży zakres pomiaru i odwrotnie). Dzięki temu można dobrze przetestować metodę doboru punktów początkowych dla funkcji $f_{solve}()$. Zastosowana metoda może skutkować stopniowym przeszukiwaniem szerokiego przedziału (od bardzo niskiej granicy wykrywalności do dużej aktywności). W rzeczywistości granica wykrywalności dla typowych elektrod jest często o kilka rzędów wyższa (mniejszy zakres) [11].

Testy wykonano na serwerze z procesorem Core 2 duo, 2.66 GHz.

3.3.1. Testowanie wyznaczania aktywności

Jako dane wejściowe algorytmu wykorzystano model domyślny z ładunkami: $z_1 = 3$, $z_2 = 2$, $z_3 = -2$. Rys. 3.4 przedstawia wartości ΔE (w programie: $EE - E$), a_0 ($A0$), a (A) w kolejnych iteracjach (patrz rozdział 3.2.3).

Można zauważyć, że metoda poszukiwania poprawnej aktywności jest zbieżna i ΔE zdąża do wartości bardzo bliskiej zeru. Można również potwierdzić to, co zostało napisane w rozdziale 3.2.3 - kryterium poprawności wyniku, przyjęty jako $|\Delta E| > 0.1\%E$ jest wystarczający. Przykładowo w ostatniej błędnej iteracji (14) zasymulowany pomiar SEM dla elektrody selektywnej na jon o $z_1 = -2$ (kolor czerwony wykresu) wynosił ok. 260. Następna pętla zmniejszyła tę wartość do ok. $6e - 6$. W przypadku elektrody selektywnej na jon o $z_2 = 2$ (kolor niebieski) jest to spadek z błędu wynoszącego 55 do $3e - 10$. Na tym przykładzie widać słabość doboru punktów startowych, polegający na stopniowym zwiększaniu ich o rząd. Zmiana metody, np. na połowienia przedziałów, pozwoliłaby skrócić

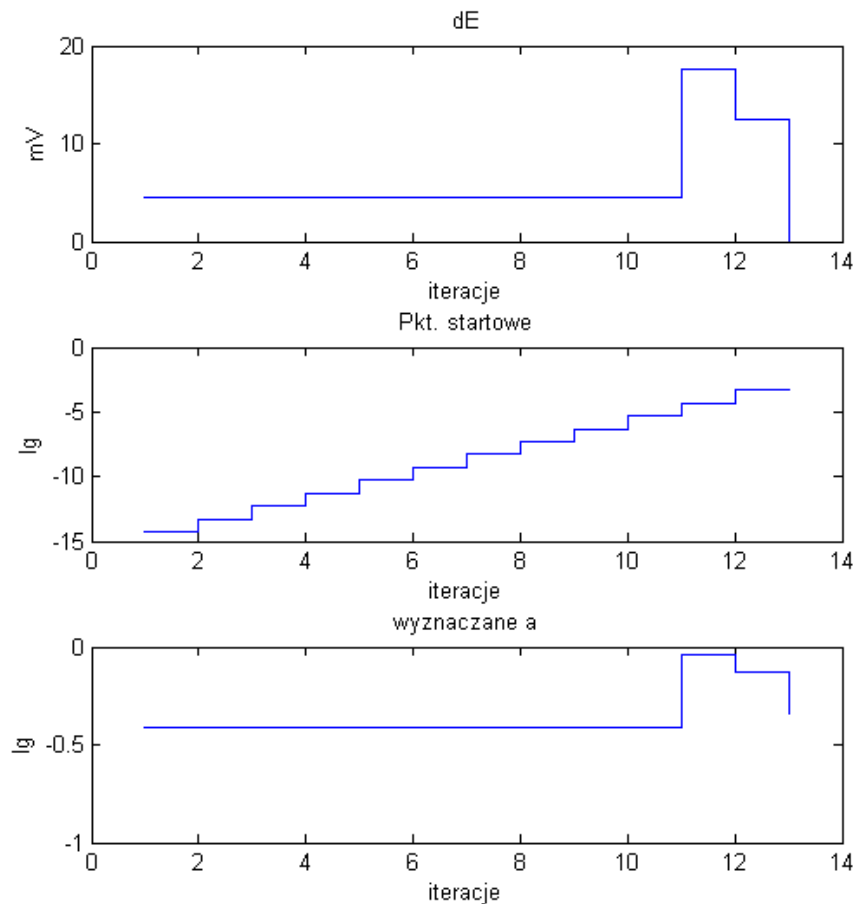


RYSUNEK 3.4. Przykładowe obliczenia aktywności 1.

ten proces o kilka iteracji i w ten sposób zaoszczędzić czas nawet o ok. 1 sekundę. Z punktu widzenia funkcjonalności nie jest to jednak dużo.

Na Rys. 3.5 przedstawiono odpowiedź algorytmu na model zbudowany z jednej elektrody modelu domyślnego o wsp. selektywności równym 1 i jego niepewności równej 0. Analizowano jon o ładunku $z_1 = 3$.

Na powyższym wykresie widać odwrotną sytuację niż we wcześniejszym przykładzie. Różnica w rzeczywistym pomiarze a symulowanym rośnie, po czym gwałtownie maleje w ostatniej znajdującej rozwiązanie iteracji. W tym przypadku kryterium opierające się na dwojeniu przedziału, „poszukiwałoby” rozwiązania blisko początku przedziału, gdzie znajdują się mniejsze różnice pomiędzy E i EE . ΔE w pierwszych kilku iteracjach wynosiła ok. 4.5, a obliczana wtedy aktywność ok. 0.73. W ostatniej iteracji po której zadziało kryterium stopu ΔE wynosiła $4.7e-9$, a obliczona niepewność ok. 0.77. Niewielkie błędy w podanej a symulowanej SEM w małym stopniu odbijają się na wyliczanej aktywności.



RYSUNEK 3.5. Przykładowe obliczenia aktywności 1.

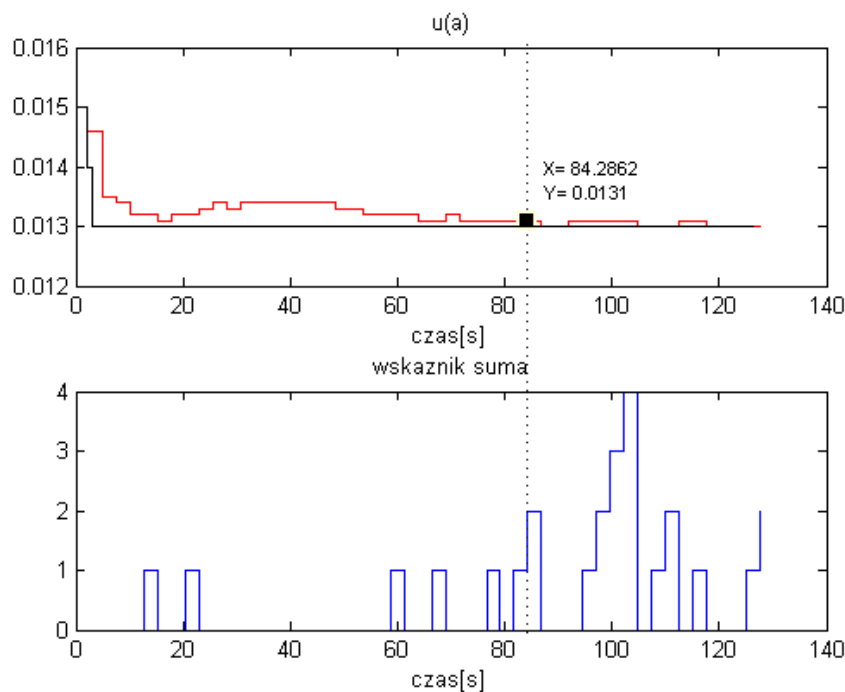
Zastosowane kryterium pozwala na uzyskanie bardzo dobrej dokładności. Metodyczne przeszukiwanie przedziału krok po kroku jest najprostsze i najpewniejsze mimo, iż czasem można byłoby zastosować szybsze. Czasem występujące błędy na 6 i dalszych cyfrach znaczących symulowanej SEM, odpowiadają błędom na 8 i dalszych cyfrach znaczących wyznaczonej aktywności i nie są wyświetlane przy prezentacji wyników. Zmiana kryterium tylko w niektórych przypadkach pozwoliłaby zyskać pewien ułamek sekundy.

3.3.2. Testowanie wyznaczania niepewności

Poniżej przedstawiono kilka przykładów zachowania algorytmu przy oszacowywaniu niepewności. Na pierwszych wykresach ukazano kolorem czerwonym trzy znaczące cyfry, a kolorem czarnym dwie znaczące cyfry, szacowanej niepewności dla aktywności jonu o ładunku z_1 . Kolorem niebieskim wykreślono wartości zmiennej *suma*, na podstawie której opracowano kryterium stopu co opisano w rozdziale 3.2.4. Obliczenia wykonywano

5000 razy, co jest ilością wystarczającą dla bardzo dobrej dokładności, przy zastosowaniu modelu domyślnego (chyba, że podano inaczej) [17].

Na Rys. 3.6 przedstawiono odpowiedzi algorytmu dla następujących ładunków: $z_1 = 3$, $z_2 = 2$, $z_3 = -1$.



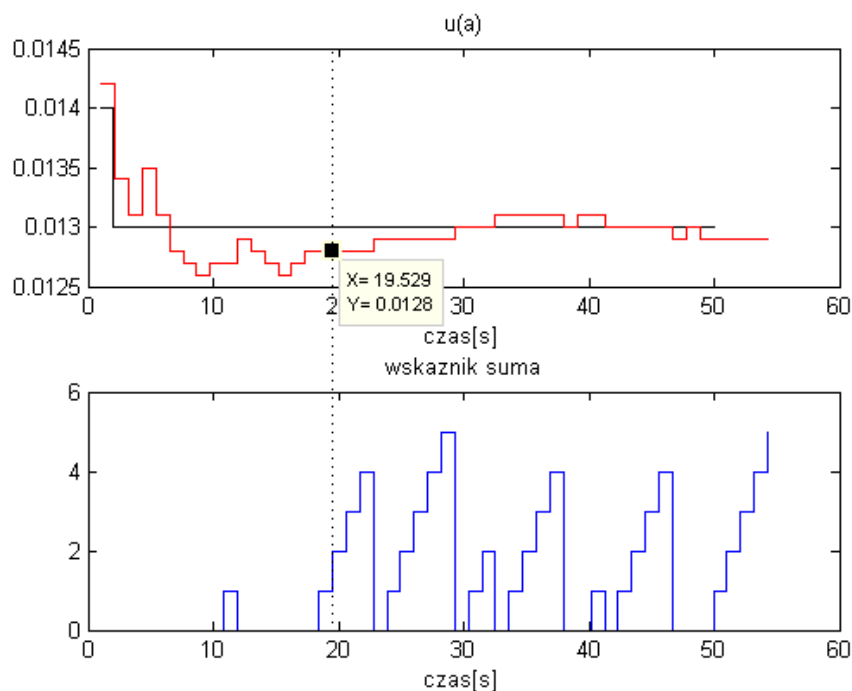
RYSUNEK 3.6. Przykładowe szacowania niepewności 1.

Jak widać metoda Monte Carlo uzyskała poprawną dokładność dla dwóch znaczących cyfr już po kilku sekundach symulacji. Wybrane kryterium stopu zatrzymuje ją jednak dopiero po 84 s., co i tak jest skróceniem o 45 s. czasu trwania 5000 iteracji. W tym przykładzie również widać, że metoda ta nawet po 2 s. podałaby bardzo dobre przybliżenie do dwóch miejsc znaczących. Zastosowane kryterium pozwoliło skrócić czas o 35 s.

Na Rys. 3.6 przedstawiono odpowiedzi algorytmu dla następujących ładunków: $z_1 = 1$, $z_2 = 2$, $z_3 = 1$.

Ukazane dwa przykłady charakteryzują się bardzo szybko szacowaną niepewnością z dokładnością do 2 cyfr i niewielkimi zmianami na miejscu 3 cyfry. Dzięki temu wskaźnik *suma* zatrzymuje pętlę w bezpiecznych chwilach, po których nie następują już gwałtowne zmiany. Dzieje się tak ponieważ zastosowano model ze stosunkowo niedużymi niepewnościami. Należy zauważyć ponadto, że ręczne ograniczenie czasu działania przez użytkownika do 10 s. również dałoby bardzo dobrą dokładność.

Na Rys. 3.8 przedstawiono odpowiedzi algorytmu dla modelu o ładunkach $z_1 = 1$, $z_2 = 2$, $z_3 = 1$ i następujących parametrach:

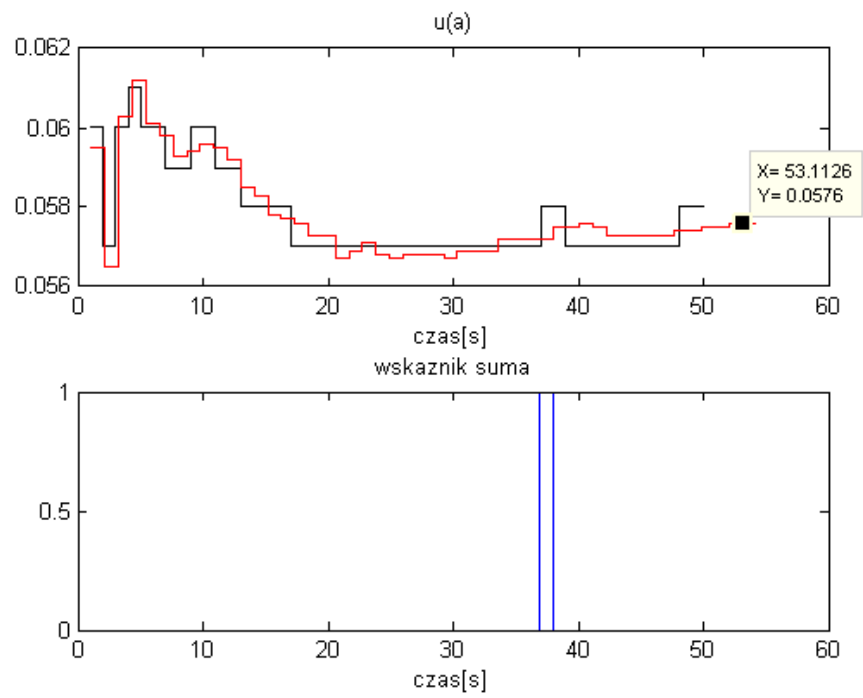


RYSUNEK 3.7. Przykładowe szacowania niepewności 2.

	E	$u(E)$	E°	$u(E^\circ)$	S	$u(S)$	K^{Pot}	$u(K^{Pot})$				a_0	$u(a_0)$	
El. nr.1	80	1	100	3	59	1.5	1	1e-8	1e-8	0	1e-15	1e-15	5e-16	2.7e-18
El. nr.2	80	1.5	100	2	59	2	1e-8	1	1e-8	1e-15	0	1e-15	5e-16	2.7e-18
El. nr.3	80	0.8	100	1	59	2.5	1e-8	1e-8	1	1e-15	1e-15	0	5e-16	2.7e-18

Model ten charakteryzuje wprowadzana większa niepewność parametrów, co w prostej linii przyczynia się do losowania liczb z rozkładów o większym odchyleniu standardowym. Jak widać na załączonym rysunku kryterium oparte na zmiennej *suma* nie zadziałało mimo, iż przy dokładności do 2 cyfr następuje pewna stabilizacja w czasie 18-37 s. Przy większej dokładności wyliczenia charakteryzują się jednak zbyt małą stabilnością, dlatego algorytm wykonał wszystkie 5000 powtórzeń w czasie 55 s. W tym przypadku nawet rozsądne skrócenie czasu przez użytkownika, może spowodować wyświetlenie niedokładnego wyniku.

Można wywnioskować, że przy bardzo dużych niepewnościach modelu, 5000 iteracji może się okazać niewystarczające i pojawi się wtedy pewne prawdopodobieństwo niedokładnego oszacowania. Zwiększanie liczby powtórzeń, a co za tym idzie wydłużanie obliczeń i tak trwających prawie minutę, dla zniwelowania możliwego błędu ± 1 na drugiej znaczącej cyfrze, byłoby zbyt nieopłacalne ze względu na funkcjonalność. Jak wykazano, zastosowane kryterium stopu oparte o wskaźnik *suma* nie zwiększa ryzyka niedokładnego wyniku, a niejednokrotnie zmniejsza czas oczekiwania o ponad połowę.



RYSUNEK 3.8. Przykładowe szacowania niepewności 3.

Rozdział 4

Zakończenie

4.1. Podsumowanie i wnioski

W powyższej pracy omówione zostały podstawowe zagadnienia związane z potencjometrią. Opisano także zasadę i sposób działania aplikacji. Jak wykazano, w wielu dziedzinach nauki wykorzystuje się tego typu pomiary. Wielofunkcyjne urządzenia laboratoryjne, czy prostsze pH-metry przekształcają mierzone SEM na odpowiedni wynik wyświetlany wraz z niepewnością. Wyznaczanie niepewności metodą analityczną jest trudne i czasochłonne. Dlatego numeryczny odpowiednik tego procesu, bazujący w dużej mierze na statystyce, mimo pewnych uproszczeń może okazać się bardzo praktyczny. W pomiarach wieloskładnikowych nawet obarczone niewielkimi błędami wyniki mogą być bardzo porządane ze względu na stosunkowo dużą szybkość ich otrzymywania. Dając w ten sposób łatwy pogląd na skład badanego roztworu w analizie pomiarów wstępnych, czy terenowych gdzie często nie jest wymagana bardzo duża dokładność. Szybki wgląd na otrzymaną niepewność może również okazać się przydatny przy podjęciu decyzji o powtórzeniu pomiarów, procesie kalibracji, czy zastosowaniu innej aparatury lub metody pomiarowej.

W części testowej opisano również otrzymywane wyniki i ich ocenę. Wykazano poprawnie otrzymywane aktywności oraz szacowane niepewności, które obarczone mogą być pewnymi niedokładnościami. Czynnikiem, którego wpływu na otrzymywane rezultaty nie przebadano, jest nieuwzględnienie korelacji współczynników selektywności. Rezygnacja z brania ich pod uwagę wynikała z założeń początkowych. Niedokładności wynikające z zastosowania metody Monte Carlo można przeanalizować na podstawie przytoczonej literatury [9, 17]. Przy wykorzystywaniu dwóch znaczących cyfr niepewności (trzech dla poprawnego zaokrąglania) oraz odpowiednio dużej liczby testów, metoda ta pozwala uzyskać bardzo dobre rezultaty. Zastosowane jej ograniczenie było podyktowane uwzględnieniem funkcjonalności. Jak wykazano jednak, nie wpływa ono na zmianę wyniku, a skraca czas w określonych przypadkach nawet o ponad połowę. Ponadto, uwzględniono możliwość skrócenia czasu obliczeń przez użytkownika, przydatne zwłaszcza przy modelu zbudowanego z wielu elektrod i dużych niepewności ich parametrów. Można zatem przyjąć, że obok uproszczonego uwzględnienia niepewności współczynników selektywności, czas oczekiwania na wynik jest drugą niedogodnością aplikacji.

W tekście wykazano również pozytywną realizację przyjętych wytycznych dotyczących wprowadzania danych. Zbudowano prosty i przejrzysty interfejs, pozwalający na ręczne

wpisywanie wartości. Umożliwiono także wykorzystanie plików pochodzących z podobnej aplikacji, wyznaczającej parametry elektrod. Napisano również podstawowe funkcje sprawdzające poprawność wrowadzonych danych.

Wykorzystanie wielu środowisk programistycznych pozwoliło autorowi na lepsze ich poznanie i nabycie nowych umiejętności. Zastosowanie JavaScript przyczyniło się do zrozumienia tworzenia stron www w oparciu o obiektowy model dokumentu oraz wymusiło na autorze, nauczenia się tego języka od podstaw. Zostały również poznane lub przypomniane zjawiska i procesy związane z pomiarami potencjometrycznymi oraz ich statystyczną oceną.

4.2. Propozycje dalszego rozwoju pracy

Opisaną aplikację można znaleźć pod następującym adresem: <http://science.wiora.pl/~herok/poten> Oznaczono ją wersją 0.1, którą pod wieloma względami można ulepszyć bądź rozwinąć. Wynika to zarówno z początkowo przyjętych założeń, jak i uproszczeń wprowadzonych w toku tworzenia.

Podstawowy kierunek rozwoju powinien uwzględniać zarówno poprawność wyników jak i szybkość ich otrzymywania. Na większą rzetelność powinno wpłynąć uwzględnienie macierzy kowariancji współczynników selektywności. Przykładowy algorytm oraz propozycję powiązania dwóch losowych wartości poprzez współczynnik korelacji można odnaleźć w przytaczanym źródle [10].

Uproszczenia wynikające z zastosowania metody Monte Carlo można uwzględnić w różny sposób. O jej niedokładności możemy mówić w przypadku niewielkiej liczby testów. Te ogranicza czas rozwiązywania układów równań nieliniowych zmodyfikowaną metodą Newtona, za pomocą funkcji *fsolve()*. Należałoby zaimplementować inne algorytmy i zbadać ich czasy działania. W toku tworzenia aplikacji przebadano m.in. metodę Brenta (zaimplementowaną w funkcji *fzero()*). Jest ona połączeniem metody Newtona i bisekcji oraz w domyśle ma łączyć szybkość zbieżności tej pierwszej z globalną zbieżnością tej drugiej. Nie jest jednak przystosowana do działania na układzie równań. Metody globalne mogą również zwracać więcej niż jedno rozwiązanie, z których trzeba byłoby wybrać to odpowiednie. Jest to również utrudnienie, dlatego przy znajomości przedziału w którym szukamy rozwiązania, metoda lokalna wydaje się jednak skuteczniejsza. Wykazały to również testy, w których odpowiednie aktywności z dobrą dokładnością uzyskiwano już w pierwszych iteracjach. Testowano również działanie metody potocznie nazywanej metodą najmniejszych kwadratów (MNK), również z uwzględnieniem tw. Frischa-Waugh-Lavella. Jednakże przy różnych ładunkach jonów, ten sposób nie pozwala na poprawne obliczenie wyników, a otrzymywane wyniki są błędne. Metoda ta w szybki sposób pozwala uzyskać

oszacowania poszukiwanych parametrów, opiera się bowiem tylko na kilku działaniach na macierzach. Można zatem dokonać pewnej hybrydyzacji i w przypadku jednakowych ładunków wszystkich jonów, kiedy możliwe jest przekształcenie układu w liniowy, zastosować MNK. Pozwoliłoby to w tych szczególnych sytuacjach znacznie przyspieszyć obliczenia.

Powyższe rozwiązania w częściowy sposób mogą przyczynić się do przyspieszenia algorytmu, a co za tym idzie również zastosowania większej liczby iteracji metody Monte Carlo i zwiększenia jej dokładności. Dużo lepszym lecz bardziej pracochłonnym sposobem jest rezygnacja ze środowiska Octave i przepisanie algorytmu na język C. Nie zawiera on sam w sobie funkcji wejść-wyjść i jest kompilowany i łatwo przekładany na kod maszynowy. Jego wykorzystanie znacząco wpłynęłoby na przyspieszenie obliczeń. W tym celu należałoby stworzyć klasę działań na macierzach lub wykorzystać jakieś gotowe biblioteki jeśli takowe istnieją. Należy pamiętać, że przykładowo w przypadku trzech równań, zastosowanie C pozwoliłoby zmniejszyć czas z rzędu sekund do rzędu milisekund oraz zniwelowałoby prawdopodobnie wszelkie możliwe niedokładności poprzez zastosowanie wielu tysięcy iteracji. Poniesiony nakład pracy podniósłby funkcjonalność algorytmu przede wszystkim z powodu przyspieszenia jej działania. Najprawdopodobniej jest to najlepszy kierunek rozwoju aplikacji.

Warto byłoby również, wykonać testy na rzeczywistych pomiarach oraz porównać je z przeprowadzonymi obliczeniami analitycznymi.

Istnieją również inne mniej istotne czynniki, które na wiele sposobów można poprawić, a które zauważone zostały podczas pisania programu lub częściej po testowaniu końcowej wersji. W związku z tym ich uwzględnienie najczęściej ograniczył czas.

Jak już nadmieniono przy wyznaczaniu aktywności funkcja *fsolve()* nie zawsze od razu zwraca poprawne rozwiązanie. W tym celu zmieniane są stopniowo punkty startowe przed ponownym jej wykorzystaniem. Proces ten można bardziej zoptymalizować i wartości początkowe wyznaczać poprzez dwojenie znanego przedziału. W ten sposób rozwiązanie szukane byłoby zawsze w połowie przedziału. Metoda ta wydaje się skuteczniejsza z punktu pisania algorytmów przeszukujących, należałoby jednak przetestować jej współdziałanie z funkcją *fsolve()*, jak wykazano bowiem czasem jej wykorzystanie nie wniosłoby żadnej poprawy. Ta optymalizacja pozwoliłaby na zaoszczędzenie czasu rzędu kilkuset ms. W porównaniu do długości trwania całego algorytmu jest to bardzo niewiele.

Kolejnym udogodnieniem byłoby umożliwienie użytkownikowi ograniczenie algorytmu również w inny sposób niż tylko poprzez czas jej trwania. Bez znajomości sposobu działania algorytmu czas nie jest wymiernym wskaźnikiem chociaż bardzo praktycznym. Dlatego mogłaby istnieć również opcja podawania akceptowalnej precyzji. Z zasady działania metody Monte Carlo wynika bowiem, że przy akceptacji pewnych niedokładności rezultaty

uzyskuje się bardzo szybko. Można powiedzieć, że dokładność tej metody rośnie inercyjnie. Te inne kryteria stopu należałoby również uwzględnić przy końcowej ocenie rzetelności wyników.

Istnieje pewna zależność pomiędzy, złożonością modelu (ilość niewiadomych, niepewność parametrów), czasem i dokładnością działania metody Monte Carlo. Na tej podstawie istnieje teoretyczna możliwość wyliczenia z dużym uproszczeniem np. dokładności jaką uzyska użytkownik po wprowadzeniu ograniczenia czasowego/iteracyjnego lub odwrotnie. Można również oszacować czas potrzebny na dokonanie odpowiedniej liczby iteracji dla bardzo dobrej dokładności. Działanie te musiałyby być wykonane przed wysłaniem formularza, czyli zostać napisane w JavaScript.

Biorąc pod uwagę interfejs, w żaden sposób nie narzuca on jednostki posiadającym je parametrom (E, E°, S) . Wyznaczenie przedziału, w którym poszukiwane są wartości początkowe, przy wyliczaniu aktywności oraz jej końcowa ocena wymusza jednak zastosowanie przez użytkownika skali logarytmicznej. W celu podniesienia funkcjonalności należałoby uzależnić brana pod uwagę skalę od wprowadzonej granicy wykrywalności (liniowo lub logarytmicznie). Przy wprowadzaniu danych można również wprowadzić dodatkowe zabezpieczenie np. przed wpisywaniem liter lub zablokować po prostu pewne znaki ASCII.

Niektóre funkcje pisane jako pierwsze można zoptymalizować pod względem działania i czytelności zapisu. Należy podkreślić, że niektórych języków autor uczył się wraz z pisaniem aplikacji i nabyte umiejętności przy ponownym programowaniu pozwoliłyby uzyskać dużo lepszy kod. Wprowadzenie większej ilości komentarzy również mogłoby wpłynąć na lepszy rozwój aplikacji w przyszłości.

Dodatek A

Kod źródłowy niektórych plików

. functions.js

```
1 // F usuwanie dzieci obiektów
2 function removeChildren(obj)
3 {
4   if(typeof(obj)=='string'){
5     obj = document.getElementById(obj);
6   }
7   while(obj.firstChild!=null){
8     obj.removeChild(obj.firstChild);
9   }
10  return false;
11 }
12
13 // F okreslajaca wartosci domyslne
14 function addvalue(ID_parametr,endline,nr_lini,iloscJonow)
15 {
16   var nazwa=0;
17   switch(ID_parametr)
18   {
19     case 'E':
20       nazwa=80;
21       break;
22     case 'uE':
23       nazwa=0.1;
24       break;
25     case 'EO':
26       nazwa=100;
27       break;
28     case 'uEO':
29       nazwa=0.7;
30       break;
31     case 'S':
32       nazwa=59;
33       break;
34     case 'uS':
35       nazwa=0.4;
36       break;
```

```

37         case 'A0':
38             nazwa=5e-16;
39             break;
40         case 'uA0':
41             nazwa=2.7e-18;
42             break;
43
44         case 'K':
45             if ((iloscJonow-nr_lini>0 && endlne==iloscJonow-nr_lini) ||
↳(iloscJonow-nr_lini<=0 && endlne==iloscJonow))
46             {
47                 nazwa=1;
48             }
49             else
50             {
51                 nazwa=1e-4;
52             }
53             break;
54         case 'uK':
55             if ((iloscJonow-nr_lini>0 && endlne==iloscJonow-nr_lini) ||
↳(iloscJonow-nr_lini<=0 && endlne==iloscJonow))
56             {
57                 nazwa=0;
58             }
59             else
60             {
61                 nazwa=1e-15;
62             }
63             break;
64         default:
65             nazwa=null;
66     }
67     return nazwa;
68 }
69
70
71 // F dodawanie dzieci
72 function addChildren(name_parametr,ID_parametr,endlne,nr_lini,iloscJonow)
73 {
74     var nazwa=0;
75     var a = document.createElement('input');           //inputy
76     a.setAttribute('type', 'text');
77     a.setAttribute('name', name_parametr);
78     a.setAttribute('size', '5');
79     a.setAttribute('value', addvalue(ID_parametr,endlne,nr_lini,iloscJonow));

```

```

80 var b = document.createElement('br');           //znak nowej lini
81 var c = document.createElement('p');           //dodawanie nazwy
82     nazwa=ID_parametr;
83     switch (nazwa)
84     {
85         case 'uE':
86             nazwa='u(E)';
87             break;
88         case 'uEO':
89             nazwa='u(E0)';
90             break;
91         case 'uS':
92             nazwa='u(S)';
93             break;
94         case 'uK':
95             nazwa='u(K)';
96             break;
97         case 'A0':
98             nazwa='a0';
99             break;
100        case 'uA0':
101            nazwa='u(a0)';
102            break;
103    }
104    c.innerHTML = nazwa+' : ';
105
106 var f = document.createTextNode('z'+nr_linii+' ');           //dodawanie nazwy dla z
107 var e = document.createElement('select');           //obiekty select
108     e.setAttribute('type', 'text');
109     e.setAttribute('name',name_parametr);
110
111     var opcja;
112
113         opcja = document.createElement('option');
114         opcja.appendChild(document.createTextNode('+3'));
115         opcja.setAttribute('value','3');
116         e.appendChild(opcja);
117         opcja = document.createElement('option');
118         opcja.appendChild(document.createTextNode('+2'));
119         opcja.setAttribute('value','2');
120         e.appendChild(opcja);
121         opcja = document.createElement('option');
122         opcja.appendChild(document.createTextNode('+1'));
123         opcja.setAttribute('value','1');
124         opcja.selected=true;

```

```

125         e.appendChild(opcja);
126         opcja = document.createElement('option');
127         opcja.appendChild(document.createTextNode('-1'));
128         opcja.setAttribute('value', '-1');
129         e.appendChild(opcja);
130         opcja = document.createElement('option');
131         opcja.appendChild(document.createTextNode('-2'));
132         opcja.setAttribute('value', '-2');
133         e.appendChild(opcja);
134         opcja = document.createElement('option');
135         opcja.appendChild(document.createTextNode('-3'));
136         opcja.setAttribute('value', '-3');
137         e.appendChild(opcja);
138
139
140         if ((nr_lini==0 && ID_parametr!='K') && (nr_lini==0 && ID_parametr!='uK') ||
        ➡((ID_parametr!='K' || ID_parametr!='uK') && nr_lini==0 && endlime==iloscJonow)) &&
        ➡ID_parametr!='Z') // || ID_parametr!='uK'))//endlime==iloscJonow
141         {      document.getElementById(ID_parametr).appendChild(c)      //dodanie tytułu
        ➡kolumny
142         if (name_parametr!=null && ID_parametr!='Z')
143         {      document.getElementById(ID_parametr).appendChild(a)      //dodanie
        ➡elementu input
144         //else
145         //{      document.getElementById(ID_parametr).appendChild(d)      //dodanie liczby
        ➡porzadkowej elektrod
146         if (ID_parametr=='Z')
147         {      document.getElementById(ID_parametr).appendChild(f);      //dodanie
        ➡ list rozwijanych dla Z
148                 document.getElementById(ID_parametr).appendChild(e);
149                 document.getElementById(ID_parametr).appendChild(b);
150         }
151         if (endlime==1)
152         {      document.getElementById(ID_parametr).appendChild(b)      //dodanie nowej
        ➡lini
153
154         return false;
155 }
156
157 // F czyszczenie, uzupełnianie, sprawdzanie po wpisaniu liczby elektrod i jonow
158 function dodajPole()
159 {
160
161 var iloscElektrod=document.forms.formularz1.lelektrod.value;
162 var iloscJonow=document.forms.formularz1.ljonow.value;

```

```

163
164 if (iloscElektrod>10) //liczba elektrod nie wieksza niz 10
165     {
166         document.forms.formularz1.lelektrod.value=10;
167         iloscElektrod=10;
168     }
169 if (iloscJonow > iloscElektrod) //liczba jonow nie wieksza niz liczba elektrod
170     {
171         document.forms.formularz1.ljonow.value = document.forms.formularz1.lelektrod.value;
172         iloscJonow=iloscElektrod;
173     }
174
175
176
177 //wywołania czyszczenia divow
178 removeChildren('E');
179 removeChildren('uE');
180 removeChildren('EO');
181 removeChildren('uEO');
182 removeChildren('S');
183 removeChildren('uS');
184 removeChildren('K');
185 removeChildren('uK');
186 removeChildren('AO');
187 removeChildren('uAO');
188 removeChildren('Z');
189
190 //uzupełnianie elementow input
191 for (var i=0;i<iloscElektrod;i++)
192     {
193         //addChildren(null, 'licz_El', 1, i);
194         addChildren('E□', 'E', 1, i, iloscJonow);
195         addChildren('uE□', 'uE', 1, i, iloscJonow);
196         addChildren('EO□', 'EO', 1, i, iloscJonow);
197         addChildren('uEO□', 'uEO', 1, i, iloscJonow);
198         addChildren('S□', 'S', 1, i, iloscJonow);
199         addChildren('uS□', 'uS', 1, i, iloscJonow);
200         for (j=0;j<iloscJonow;j++)
201             {
202                 endlne=iloscJonow-j;
203                 addChildren('K□', 'K', endlne, i, iloscJonow);
204             }
205         for (var j=0;j<iloscJonow;j++)
206             {
207                 endlne=iloscJonow-j;

```

```

208         addChildren('uK[]','uK',endline,i,iloscJonow);
209     }
210     addChildren('A0[]','A0',1,i,iloscJonow);
211     addChildren('uA0[]','uA0',1,i,iloscJonow);
212 }
213
214 for (var z=0;z<iloscJonow;z++)
215 {
216     addChildren('Z[]','Z',0,z+1,iloscJonow);
217 }
218 }
219 return false;
220 }
221
222 // F sprawdzanie wartosci przed proba wyslania formularza
223 function sprawdzanieGeneracja()
224 {
225     var alarm=false;
226     var iloscElektrod=document.forms.formularz1.lelektrod.value;
227     var iloscJonow=document.forms.formularz1.ljonow.value;
228     var pole=1+6*iloscElektrod;
229     var tekst = document.createElement('p');
230
231     tekst.appendChild(document.createTextNode('value of this parameters should be equal 1: '));
232
233
234     for(var i=1;i<=iloscElektrod;i++)
235     {
236         for(var j=1;j<=iloscJonow;j++)
237         {
238             var wartosc=pole+((j-1)*iloscJonow+i)
239             if (i==j && document.forms.formularz1.elements[wartosc].value!=1)
240             {
241                 removeChildren('komunikaty');
242                 tekst.appendChild(document.createTextNode('Kpot '+i+', '+j+', '));
243                 document.getElementById('komunikaty').appendChild(tekst);
244
245                 alarm=true;
246             }
247         }
248     }
249     if (alarm!=true)
250     {
251         document.formularz1.submit();
252     }

```

253 }

. glowny.m

```
1 clear all
2 sciezka=argv(){1,1};
3 aktualna=pwd();
4
5 [zE,zEO,zS,zA0,KK,ZZ,ll elektrod,ll jonow,zuE,zuEO,zuS,zuA0,uK,time]=wczytywanie(sciezka);
6
7 %% transponowanie macierzy K i uK do wyswietlenia w raporcie
8 trK=KK';
9 utrK=uK';
10
11 cd (sciezka);
12 save -ascii trK.txt trK;
13 save -ascii utrK.txt utrK;
14 cd (aktualna);
15
16 #####% zmiana macierzy
17 if (llelektrod>lljonow)
18     for (i=1:lljonow)
19         E(i,1)=zE(i);
20         uE(i,1)=zuE(i);
21         EO(i,1)=zEO(i);
22         uEO(i,1)=zuEO(i);
23         S(i,1)=zS(i);
24         uS(i,1)=zuS(i);
25         AO(i,1)=zA0(i);
26         uAO(i,1)=zuA0(i);
27     end
28     for i=lljonow+1:llelektrod
29         for j=1:lljonow
30             if (KK(i,j)==1)
31                 wybor=j;
32                 break;
33             end
34         end
35         E(wybor,1)=E(wybor,1)+zE(i,1);
36         uE(wybor,1)=uE(wybor,1)+zuE(i,1);
37         S(wybor,1)=S(wybor,1)+zS(i,1);
38         uS(wybor,1)=uS(wybor,1)+zuS(i,1);
39         EO(wybor,1)=EO(wybor,1)+zEO(i,1);
40         uEO(wybor,1)=uEO(wybor,1)+zuEO(i,1);
41         AO(wybor,1)=AO(wybor,1)+zA0(i,1);
```

```

42         uA0(wybor,1)=uA0(wybor,1)+zuA0(i,1);
43     end
44 else
45         E=zE;
46         uE=zuE;
47         EO=zEO;
48         uEO=zuEO;
49         S=zS;
50         uS=zuS;
51         AO=zAO;
52         uAO=zuAO;
53 end
54
55 if (time==0)
56     time=inf;
57 elseif
58     time=time*60;
59 end
60
61
62 YY=10.^( (E-EO)./S)-AO;  #!/lewa strona rownania
63
64 global Y=YY;
65 global K=KK;
66 global Z=ZZ;
67 global lelektrod=llelektrod;
68 global ljonow=lljonow;
69
70 #!/pierwsze punkty startowe
71 for i=1:ljonow
72     start(i,1)=AO(i);
73 end
74
75 koniec=0;
76 t=1;
77 tic;
78 while (t==1)
79
80     koniec=koniec+1;
81     #!/wyznaczanie aktywnosci
82     A=fsolve("model",start);
83
84     #!/wyznaczanie E i dE na podstawie obliczonych A
85     EE=model2(EO,S,KK,AO,A,lljonow,ZZ);
86

```

```

87      ##%wyznaczanie delty E
88      for i=1:ljonow
89          deltaE(i,1)=EE(i)-E(i);
90      end
91
92      ##%sprawdzanie-warunki
93      a=0;
94      e=0;
95      for i=1:ljonow
96          if (A(i)<0)
97              a=1;
98          end
99          if (abs(deltaE(i,1))>(E(i)/1000)) ##%powtarzaj jezeli dE jest wieksze niz 0.1E
100              e=1;
101          end
102      end
103
104      ##%sprawdzanie-decyzje
105      t=0;
106      if (a==true || e==true)
107          start=start.*10;
108          t=1;
109
110      end
111      if (start>1) ##%jezeli wszystkie elementy start sa wieksze od 1
112          t=0;
113      end
114
115      ##%do celow testowych
116      testA(koniec,:)= [koniec,start',t,deltaE',A'];
117  end
118  A=A.^Z; ##%uwzglednienie ladunkow jonow w mianownikach poteg aktywnosci
119
120  #####%wyznaczanie niepewnosci#####
121  tic;
122  losuj=1;
123  test=1;
124  uA_old=0;
125  suma=0;
126  iteracja=0;
127  czas=0;
128  while (losuj==1)
129      nE=E+randn(1)*uE;
130      nEO=EO+randn(1)*uEO;
131      nS=S+randn(1)*uS;

```

```

132     K=KK+randn(1)*uK;
133     nA0=A0+randn(1)*uA0;
134
135     Y=10.^( (nE-nEO) ./nS)-nA0;
136
137     nA=fsolve("model",A);
138     nA=nA.^Z;
139     Macierz(:,test)=nA-A;
140     warunek=mod(test,100);
141
142     if (warunek==99)
143         iteracja=iteracja+1;
144         uA_new=(sqrt(meansq(Macierz')))' ;
145         uAtest=uA_new;
146         [uA_new,nieptest]=FormatNiep(uA_new);
147
148         czas=toc;
149
150         if (uA_new==uA_old)
151             suma=suma+1;
152         else
153             suma=0;
154         end
155         uA_old=uA_new;
156
157         testUA(iteracja,:)= [iteracja,uAtest',uA_new',nieptest',suma,czas] ;
158     end
159
160     if (suma>=100 || czas>=time || test>=5000)      #!/suma - ilosc takich samych powtorzen
161         ➡obliczanych niepewnosci
162         losuj=0;
163     else
164         losuj=1;
165     end
166
167     test=test+1;
168 end
169
170 if (suma >=2)
171     rzetelnosc=1;
172 else
173     rzetelnosc=0;
174 end
175

```

```
176
177 raport=[czas, test, suma]';
178
179 uA=(sqrt(meansq(Macierz')))' ;
180 uA=uA.*2;
181
182 K=KK;
183 PdeltaE=deltaE./E;      #!/delta E w procentach
184
185 cd (sciezka)
186 save -ascii A.txt A;
187 save -ascii uA.txt uA;
188 save -ascii rzetelnosc.txt rzetelnosc;
189 save -ascii raport.txt raport;
190
191 #!/do celow testowych
192 cd (aktualna);
193 save -ascii testuA.txt testUA;
```

Bibliografia

- [1] Z. Brzózka, W. Wróblewski. *Sensory chemiczne*. Oficyna Wydawnicza Politechniki Warszawskiej, Warszawa 1999.
- [2] K. Camman. *Zastosowanie elektrod jonoselektywnych*. WNT, Warszawa 1977.
- [3] E. Castro. *HTML, XHTML i CSS : książka w kolorze*. Helion, Gliwice 2008.
- [4] A. Cygański. *Podstawy metod elektroanalitycznych*. WNT, Warszawa 1999.
- [5] A. Danesh. *JavaScript*. Helion, Gliwice 1997.
- [6] J. Dojlido, J. Zerbe. *Instrumentalne metody badania wody i ścieków*. Arkady, Warszawa 1997.
- [7] K. Eckschlager. *Błędy w analizie chemicznej*. PWN, Warszawa 1974.
- [8] Główny Urząd Miar. *EA-4/02, Wyrażanie niepewności pomiaru przy wzorcowaniu*, 2001.
- [9] D. W. Heermann. *Podstawy symulacji komputerowych w fizyce*. WNT, Warszawa 1997.
- [10] M. Kopacz. Model skorelowanych procesów losowych – propozycja algorytmu wymuszania korelacji dwóch niezależnych zmiennych losowych. *IV Krakowska Konferencja Młodych Uczonych*, Kraków 2009.
- [11] A. Kozyra, A. Wiora, J. Wiora. *Szacowanie niepewności w pomiarach potencjometrycznych. Właściwości statyczne i dynamiczne elektrod jonoselektywnych w wieloskładnikowych pomiarach potencjometrycznych*. Wydawnictwo Pracowni Komputerowej Jacka Skalmierskiego, Katowice 2007.
- [12] R. Lerdorf, K. Tatroe, P. MacIntyre. *PHP5 : programowanie*. Helion, Gliwice 2007.
- [13] J. Minczewski, Z. Marczenko. *Chemia analityczna. Analiza instrumentalna. Tom 3*. PWN, Warszawa 1978.
- [14] J. Piotrowski. *Pomiary. Czujniki i metody pomiarowe wybranych wielkości fizycznych i składu chemicznego*. WNT, Warszawa.
- [15] U. Później. *Słownik chemii analitycznej*. WNT, Warszawa 1984.
- [16] W. Szczepaniak. *Metody instrumentalne w analizie chemicznej*. PWN, Warszawa 2005.
- [17] J. Wiora, Wiora A. Porównanie wyników analitycznego i symulacyjnego szacowania niepewności na przykładzie pomiaru stężenia jonów metodą potencjometryczną.
- [18] W.A. Wróblewski, P. Ciosek. *Elektroniczny nos i elektroniczny język - systemy do automatycznego rozpoznawania próbek złożonych*. Rozdział w pracy zbiorowej, Nowoczesne techniki analityczne, 2006.