

**WYŻSZA SZKOŁA INFORMATYKI
WYDZIAŁ ZAMIEJSCOWY WE WŁOCŁAWKU
KIERUNEK INFORMATYKA**

**PRACA DYPLOMOWA
INŻYNIERSKA**

**Tytuł pracy: Komputerowe metody rozwiązywania równań nieliniowych
z jedną niewiadomą**

Imię i Nazwisko: Rafał Leśniewski

Studia: informatyka

Specjalność: programowanie i bazy danych

Nr albumu: 17489

Promotor: prof. dr hab. inż Krzysztof Dems

Rok akademicki 2011

Spis treści

1. Wprowadzenie.....	3
2. Metody rozwiązywania równań nieliniowych.....	5
2.1. Przeszukiwanie przedziału.....	8
2.1.1. Algorytm działania metody.....	9
2.1.2. Schemat Blokowy.....	10
2.2. Metoda Regula falsi.....	11
2.2.1. Algorytm działania metody.....	13
2.2.2. Schemat Blokowy.....	14
2.3. Metoda Newtona.....	15
2.3.1. Algorytm działania metody.....	17
2.3.2. Schemat Blokowy.....	18
3. Java i narzędzia programistyczne.....	19
3.1. Język programowania Java.....	20
3.2. Zintegrowane środowisko programistyczne NetBeans IDE.....	21
4. Analiza wyrażeń matematycznych.....	23
4.1. JEP parser	24
4.2. DJEP procesor różniczkowania.....	25
5. Bloki działania aplikacji.....	26
6. Budowa i zastosowanie aplikacji.....	29
6.1. Przygotowanie procesu wyszukiwania metodą przeszukiwania przedziału.....	35
6.2. Przygotowanie procesu wyszukiwania metodą Regula falsi.....	37
6.3. Przygotowanie procesu wyszukiwania metodą Newtona.....	39
7. Testy i porównania.....	41
7.1. Badanie pierwszego równania.....	43
7.2. Badanie drugiego równania.....	47
7.3. Badanie trzeciego równania.....	51
8. Podsumowanie.....	55

1. Wprowadzenie

Dynamiczny rozwój elektroniki a co za tym idzie również informatyki, otworzył nowe możliwości praktycznego wykorzystania metod numerycznych. Ograniczenia konwencjonalnych metod liczenia nie są już przeszkodą w realizacji nieograniczonych możliwości badawczych istniejących metod numerycznych. Nigdy wcześniej nie istniały narzędzia wspierające badania o tak dużych możliwościach jak dziś. Obliczenia, które tradycyjnymi technikami wymagają wielu rachunków i czasu, dzięki wykorzystaniu komputerów, są wykonane błyskawicznie, ryzyko błędów w rachunkach zostało wyeliminowane.

Celem pracy jest przedstawienie możliwości, jakie dają nowoczesne technologie informatyczne, dla wykorzystania metod numerycznych przez stworzenie aplikacji, która będzie je stosować.

Aplikacja będzie charakteryzować się przede wszystkim:

- dokładnością obliczeń,
- szybkością obliczeń,
- łatwością obsługi,
- nieograniczonym dostępem.

Praca podzielona jest na rozdziały merytoryczne, każdy z nich opisuje inne zagadnienia związane z realizacją tematu niniejszej pracy. Pierwszy rozdział zawiera informacje dotyczące zagadnienia równań nieliniowych, ich definicje jak również dokładny opis użytych metod numerycznych oraz schematy działania algorytmów. Narzędzia programistyczne wykorzystane w procesie tworzenia aplikacji zostaną ujęte i opisane w drugim rozdziale niniejszej pracy. Opis wykorzystanego parsera wyrażeń matematycznych znajdzie się w rozdziale trzecim. Czwarta część poświęcona będzie opisowi etapów działania aplikacji oraz jej budowie. W ostatnim rozdziale scharakteryzowana zostanie obsługa aplikacji i przykłady jej praktycznego zastosowania.

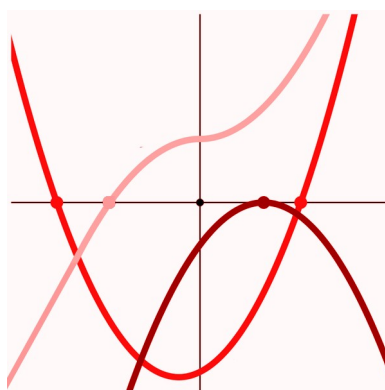
2. Metody rozwiązywania równań nieliniowych

Równaniem nieliniowym nazywamy równanie w postaci:

$$f(x)=0$$

Rozwiązywanie równań nieliniowych jest zatem wyszukiwaniem pierwiastków rzeczywistych funkcji nieliniowej, czyli punktów, dla których równanie przyjmuje wartość 0¹.

Funkcje, których wykresy nie są liniami prostymi, nazywamy funkcjami nieliniowymi (Rys. 1). By odszukać pierwiastki funkcji nieliniowych wykorzystywane są metody numeryczne, czyli metody rozwiązywania problemów matematycznych za pomocą operacji na liczbach.



Rys. 1 Wykresy funkcji nieliniowych

Nie istnieje jedna, uniwersalna metoda rozwiązywania równań. Wybór metody następuje na podstawie analizy funkcji oraz warunków, które musi spełniać funkcja (przedziały rozpatrywania funkcji, warunki brzegowe). W niniejszej pracy zostaną opisane i zanalizowane trzy metody, tzn:

- metoda przeszukiwania przedziału,
- metoda *Regula falsi*,
- metoda *Newtona*.

Metody rozwiązywania równań nieliniowych dzielimy ze względu na dokładność wyniku :

a) metody poszukiwania rozwiązań przybliżonych

- metoda przeszukiwania przedziału

b) metody uściślania pierwiastków

- metoda *Regula falsi*
- metoda *Newtona*

¹ K. Dems, *Podstawy metod numerycznych*, wykład czwarty, s. 55-57, Z. Fortuna, B. Macukow, J. Wąsowski, *Metody numeryczne*, Warszawa 1992, s. 115

Ze względu na wykorzystanie wartości funkcji i wartości jej pierwszej pochodnej metody uściślenia pierwiastków dzielimy na dwie podgrupy:

- metody rzędu zerowego – metoda regula falsi, która umożliwia znalezienie pierwiastków rzeczywistych funkcji korzystając z jej wartości,
- metody rzędu pierwszego – metoda Newtona, która umożliwia znalezienie pierwiastków rzeczywistych funkcji korzystając z jej wartości oraz jej pierwszej pochodnej. Tylko znajomość przybliżonej wartości pierwiastka pozwala obliczyć dokładną jego wartość.

2.1. Przeszukiwanie przedziału

Jest to metoda poszukiwania rozwiązań przybliżonych². Oparta jest ona na iteracyjnym skanowaniu przedziału ze stałym krokiem w poszukiwaniu pierwiastków rzeczywistych równania funkcji nieliniowej.

Co to jest pierwiastek rzeczywisty funkcji nieliniowej?

Jeśli dana jest funkcja $f(x)$, to pierwiastkiem równania $f(x)=0$ jest każda liczba spełniająca to równanie, tzn.

Dla $x = \xi$:	$f(\xi)=0$
-----------------	------------

Wzór iteracyjny

Przybliżone rozwiązania pierwiastka rzeczywistego uzyskujemy ze wzoru:

$$x^{(k+1)} = F_k(x^{(k)}, x^{(k-1)}, x^{(k-2)}, \dots, x^{(k-l)})$$

gdzie: F_k - funkcja iteracyjna stacjonarna lub niestacjonarna

Etapy wyznaczania pierwiastków

- lokalizacja pierwiastka – określenie przedziału (a,b) zawierającego jeden pierwiastek rzeczywisty,
- uściślanie wartości pierwiastka – uściślanie wartości pierwiastka z żadaną dokładnością.

Lokalizacja pierwiastka (podstawa: Twierdzenie Bolzano – Cauchy'ego)

Jeśli ciągła funkcja $f(x)$ ma różne znaki na krańcach przedziału $\langle a,b \rangle$, tzn. $f(a)*f(b)<0$, to w tym przedziale istnieje co najmniej jedna liczba spełniająca równanie $f(x)=0$, tzn. że istnieje liczba $\xi \in \langle a,b \rangle$, taka, że $f(\xi)=0$ (ξ - pierwiastek rzeczywisty). Ponadto jeśli istnieje w przedziale otwartym (a,b) pochodna $df(x)/dx$ i jest ona stałego znaku w całym przedziale, tzn. $df(x)/dx>0$ dla $a<x<b$ lub $df(x)/dx<0$ dla $a<x<b$ to ξ jest jedynym pierwiastkiem równania $f(x)=0$ w przedziale (a,b) .

Kryterium lokalizacji pierwiastka: jeśli $f(\xi^{(k)})*f(\xi^{(k+1)})<0$ to rozwiązanie znajduje się w przedziale $\langle \xi^{(k)}, \xi^{(k+1)} \rangle$

Kryteria umożliwiające zastosowanie metody:

- funkcja musi być ciągła w przedziale, w którym ją rozpatrujemy,
- w przedziale musi znajdować się co najmniej jeden pierwiastek funkcji.

² K. Doms, dz. cyt., s. 58.

2.1.1. Algorytm działania metody

Dane wejściowe

- przedział $\langle a, b \rangle$ gdzie $a < b$
- krok przeszukiwań h

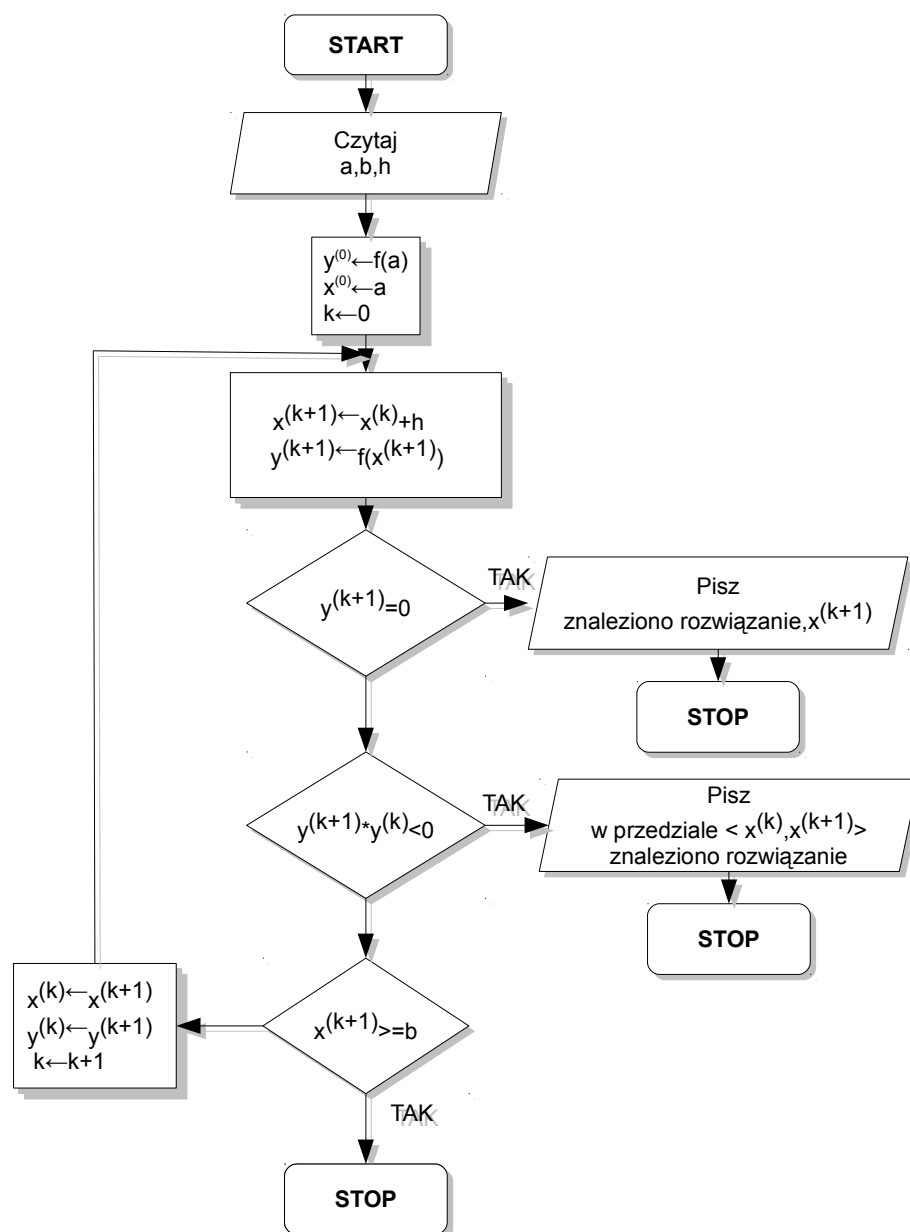
Kroki k-tej iteracji

1. $x^{(0)} = a, y^{(0)} = f(x^{(0)}), k = 0$
2. $x^{(k+1)} = x^{(k)} + h, y^{(k+1)} = f(x^{(k+1)})$
3. Jeśli $f(x^{(k+1)}) = 0$ – znaleziono rozwiązanie, $x^{(k+1)}$, KONIEC
4. Jeżeli
 $y^{(k+1)} * y^{(k)} < 0$
znaleziono rozwiązanie w przedziale $\langle x^{(k)}, x^{(k+1)} \rangle$, KONIEC
inaczej idź do kroku 4
5. Jeżeli $x^{(k+1)} \geq b$, KONIEC
inaczej
 $x^{(k)} = x^{(k+1)}, y^{(k)} = y^{(k+1)}, k = k + 1$, idź do kroku 2

Uwagi:

- metoda ta nie odnajduje pierwiastków, dla których funkcja osiąga minimum lub maksimum,
- zbyt duży krok, może sprawić, że część pierwiastków zostanie pominięta, wówczas algorytm zakończy działanie gdy osiągnie krawędź przedziału poszukiwań, tzn $x^{(k+1)} \geq b$.

2.1.2. Schemat Blokowy



2.2. Metoda Regula falsi

Nazwa metody wywodzi się z języka łacińskiego, słowo „regula” oznacza „prosta”, natomiast „falsi” fałsz. W wolnym tłumaczeniu można dokonać przekładu nazwy na „fałszywa prosta”. Metoda opiera się o założenie, że funkcja nieliniowa w odpowiednio małym przedziale przypomina funkcję liniową.

Jest to jedna z metod uściślenia pierwiastków³. Działanie metody polega na wyznaczaniu prostej (siecznej, Rys. 2) przecinającej funkcję $f(x)$ w punktach utworzonych przez krańce przedziału $\langle a, b \rangle$, tzn. $(a, f(a))$ i $(b, f(b))$. Jeśli sieczna przecina OX, wyznacza przybliżenie pierwiastka rzeczywistego funkcji nieliniowej.

Równanie prostej $y = Ax + B$ przechodzącej przez punkty $(a^{(k)}, f(a^{(k)}))$ i $(b^{(k)}, f(b^{(k)}))$

$$(b^{(k)} - a^{(k)})(y^{(k)} - f(a^{(k)})) = (f(b^{(k)}) - f(a^{(k)}))(x^{(k)} - a^{(k)})$$

$$y^{(k)} - f(a^{(k)}) = \frac{(f(b^{(k)}) - f(a^{(k)}))(x^{(k)} - a^{(k)})}{(b^{(k)} - a^{(k)})}$$

$$y^{(k)} = f(a^{(k)}) + \frac{f(b^{(k)}) - f(a^{(k)})}{b^{(k)} - a^{(k)}}(x^{(k)} - a^{(k)})$$

Jeśli prosta w przedziale (a, b) zastępuje funkcję $f(x)$ to jej miejsce zerowe jest przybliżeniem pierwiastka, tzn. $y^{(k)} = 0$ dla $x^{(k)} = \xi_p^{(k)}$

$$\xi_p^{(k)} = a^{(k)} - f(a^{(k)}) \frac{b^{(k)} - a^{(k)}}{f(b^{(k)}) - f(a^{(k)})}$$

przybliżenie pierwiastka

$$\xi_p^{(k)} \rightarrow f(\xi_p^{(k)})$$

Warunek uzyskania dokładnego wyniku, zatrzymania algorytmu (ϵ_x – żądana dokładność):

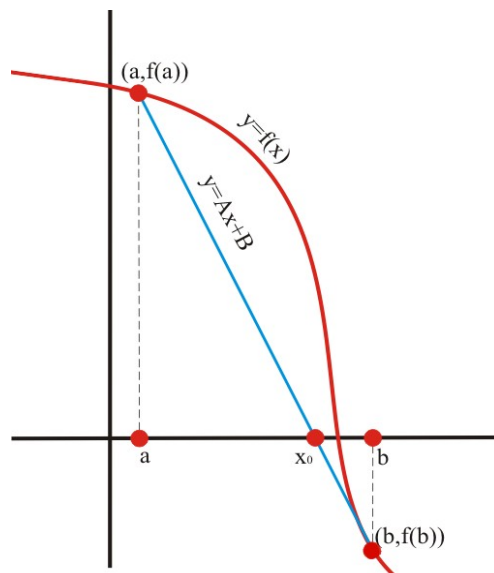
$$|\xi_p^{(k-1)} - \xi_p^{(k)}| < \epsilon_x$$

lub $f(\xi_p^{(k)}) = 0$ - znaleziono dokładne rozwiązanie

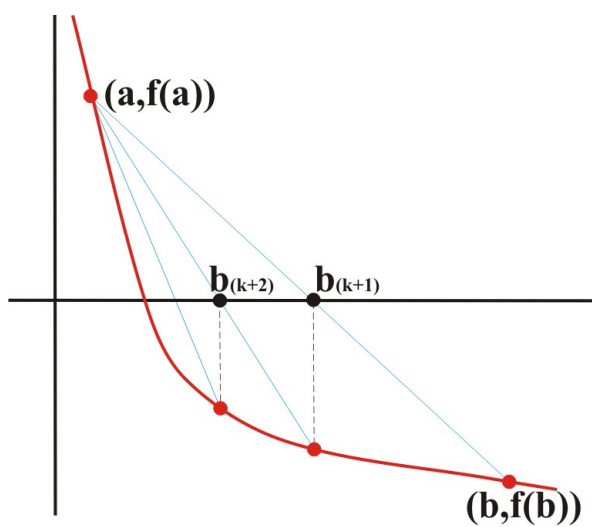
Warunki, jakie musi spełniać funkcja, by metoda mogła być wykorzystana:

- funkcja musi być ciągła w przedziale, w którym ją rozpatrujemy,
- w przedziale musi znajdować się pierwiastek.

³ K. Dems, *dz. cyt.*, s. 62-64, Z. Fortuna, B. Macukow, J. Wąsowski, *dz. cyt.*, s. 121-125



Rys. 2. Użycie metody Regula falsi



Rys. 3. Etapy uściśniania pierwiastka

2.2.1. Algorytm działania metody

Dane wejściowe

- przedział a, b , gdzie $a < b$
- k_{\max} – maksymalna ilość kroków
- Ex – dokładność wyniku

Zamiast rozwiązywać $f(\xi)=0$ dla $\xi \in (a, b)$, rozwiązujemy $A\xi_p+B=0$ (równanie prostej) dla $\xi_p \in (a, b)$.

Kroki k-tej iteracji

1. Dane początkowe

$$\{a^{(k)}, f(a^{(k)})\}, \{b^{(k)}, f(b^{(k)})\}, \xi_p \in (a^{(k)}, b^{(k)})$$

2. Przez końce przedziału $\{a^{(k)}, f(a^{(k)})\}$ i $\{b^{(k)}, f(b^{(k)})\}$ prowadzimy prostą $y=Ax+B$

$$y^{(k)} = f(a^{(k)}) + \frac{f(b^{(k)}) - f(a^{(k)})}{b^{(k)} - a^{(k)}}(x^{(k)} - a^{(k)})$$

Miejsce zerowe prostej jest przybliżeniem pierwiastka

$$y^{(k)}=0 \text{ dla } x^{(k)} = \xi_p^{(k)}$$

$$\xi_p^{(k)} = a^{(k)} - f(a^{(k)}) \frac{b^{(k)} - a^{(k)}}{f(b^{(k)}) - f(a^{(k)})}$$

$\xi_p^{(k)}$ – przybliżenie pierwiastka

3. Jeśli $f(\xi_p^{(k)}) = 0$ – znaleziono dokładny pierwiastek
4. Jeśli $|\xi_p^{(k)} - \xi_p^{(k-1)}| < Ex$ - znaleziono wynik z szukaną dokładnością, $\xi_p^{(k-1)}$, KONIEC
5. Jeśli $f(a^{(k)}) * f(\xi_p^{(k)}) > 0$ – to znaczy, że pierwiastek leży w przedziale $(\xi_p^{(k)}, b^{(k)})$, nowy przedział w iteracji ($k=k+1$):

$$a^{(k+1)} = \xi_p^{(k)}$$

$$b^{(k+1)} = b^{(k)}$$

inaczej $f(a^{(k)}) * f(\xi_p^{(k)}) < 0$ – to znaczy, że pierwiastek leży w przedziale $(a^{(k)}, \xi_p^{(k)})$, nowy przedział w iteracji ($k=k+1$):

$$a^{(k+1)} = a^{(k)}$$

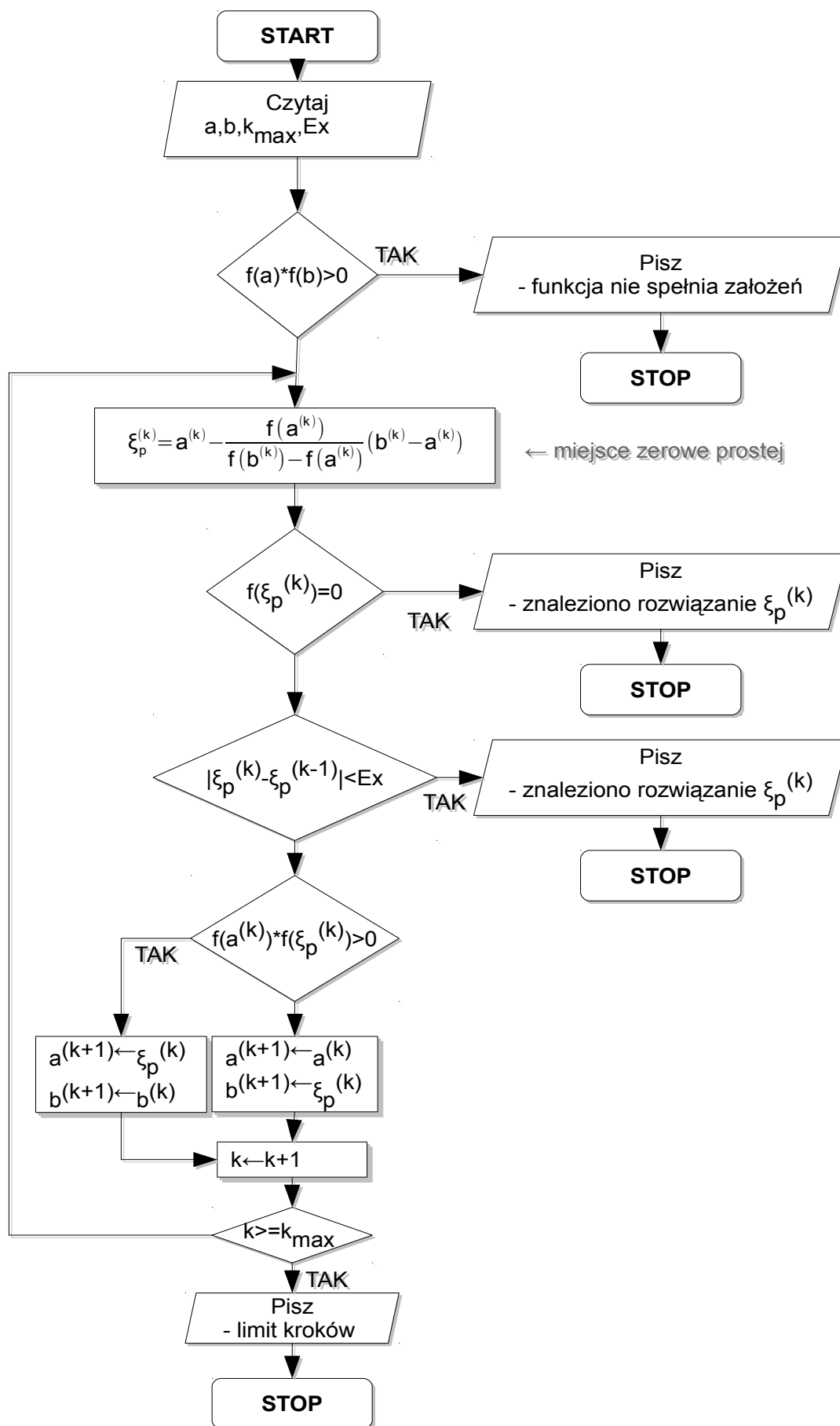
$$b^{(k+1)} = \xi_p^{(k)}$$

6. Jeśli $k \geq k_{\max}$ – przekroczono maksymalną ilość kroków, KONIEC
inaczej idź do kroku 2

Uwagi:

- zawsze zbieżna dla funkcji ciągłych
- metoda ta nie odnajduje pierwiastków, dla których funkcja osiąga minimum lub maksimum

2.2.2. Schemat Blokowy



2.3. Metoda Newtona

Jest to jedna z metod uściślenia pierwiastków⁴, w działaniu korzysta z wartości funkcji oraz jej pierwszej pochodnej. Do zastosowania metody wymagana jest znajomość przybliżonej wartości pierwiastka rzeczywistego funkcji oraz wyliczenie jej pierwszej pochodnej. Metoda jest szybko zbieżna.

W metodzie wykorzystywane są wartość funkcji $f(x^{(k)})$ oraz jej pierwszej pochodnej $\frac{df(x^{(k)})}{dx}$. Działanie metody polega na wyznaczaniu stycznej do funkcji $f(x^{(k)})$, która przecina OX wyznaczając $x^{(k+1)}$, przybliżenie pierwiastka rzeczywistego funkcji. Jeśli przybliżenie wartości pierwiastka nie jest dostatecznie dokładne, posłuży do uściślenia rozwiązania w następnej iteracji. Z punktu $(x^{(k+1)}, f(x^{(k+1)}))$ zostanie poprowadzona następna styczna.

Jeśli dane jest k-te przybliżenie pierwiastka $\xi = x^{(k)}$

$$\text{to } \xi = x^{(k)} + h^{(k)}$$

Rozwijamy w punkcie ξ funkcję $f(x)$ w szereg Taylora,

$$f(\xi) = f(x^{(k)} + h^{(k)}) = f(x^{(k)}) + \frac{df(x^{(k)})}{dx} h^{(k)} + \dots = 0$$

stad otrzymujemy równanie stycznej do funkcji

$$f(x^{(k)}) + \frac{df(x^{(k)})}{dx} h^{(k)} = 0 \rightarrow h^{(k)} = -\frac{df(x^{(k)})}{dx}$$

Iteracyjny wzór

$$x^{(k+1)} = x^{(k)} + h^{(k)} = x^{(k)} - \frac{f(x^{(k)})}{\frac{df(x^{(k)})}{dx}} \quad k=0,1,2,\dots,n$$

Warunek uzyskania dokładnego wyniku, zatrzymania algorytmu (ϵ_x – żądana dokładność):

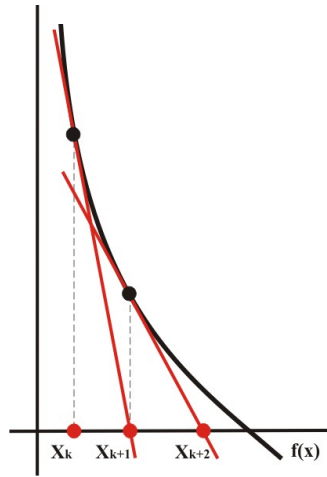
$$|x^{(k+1)} - x^{(k)}| < \epsilon_x$$

lub $f(x^{(k+1)}) = 0$ - znaleziono dokładne rozwiązanie

Aby zastosować tę metodę funkcja musi spełniać następujące warunki:

- w przedziale między punktem startowym a szukanym pierwiastkiem rzeczywistym funkcja musi posiadać niezerową pierwszą pochodną,
- funkcja musi być ciągła,
- nie mogą występować jej gwałtowne skoki,
- punkt startowy $x^{(k)}$ musi znajdować się odpowiednio blisko rozwiązania.

⁴ K. Dems, *dz. cyt.*, s. 62-64, Z. Fortuna, B. Macukow, J. Wąsowski, *dz. cyt.*, s. 121-125



Rys. 4 Ilustracja działania metody Newtona, trzy pierwsze kroki

2.3.1. Algorytm działania metody

Dane wejściowe

- a – punkt startowy (przybliżona wartość pierwiastka)
- k_{\max} – maksymalna ilość kroków
- E_x – dokładność wyniku

Kroki k-tej iteracji

1. $k=0$

$$x^{(k)} = a$$

2. Jeśli $\frac{df(x^{(k)})}{dx} = 0$ - pisz "Zły punkt startowy", KONIEC

3. Oblicz

$$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{\frac{df(x^{(k)})}{dx}}$$

4. Jeśli $f(x^{(k+1)}) = 0$ - znaleziono dokładny pierwiastek, $x^{(k+1)}$, KONIEC

5. Jeśli $|x^{(k+1)} - x^{(k)}| < E_x$

TAK – znaleziono wynik z szukaną dokładnością, $x^{(k+1)}$, KONIEC

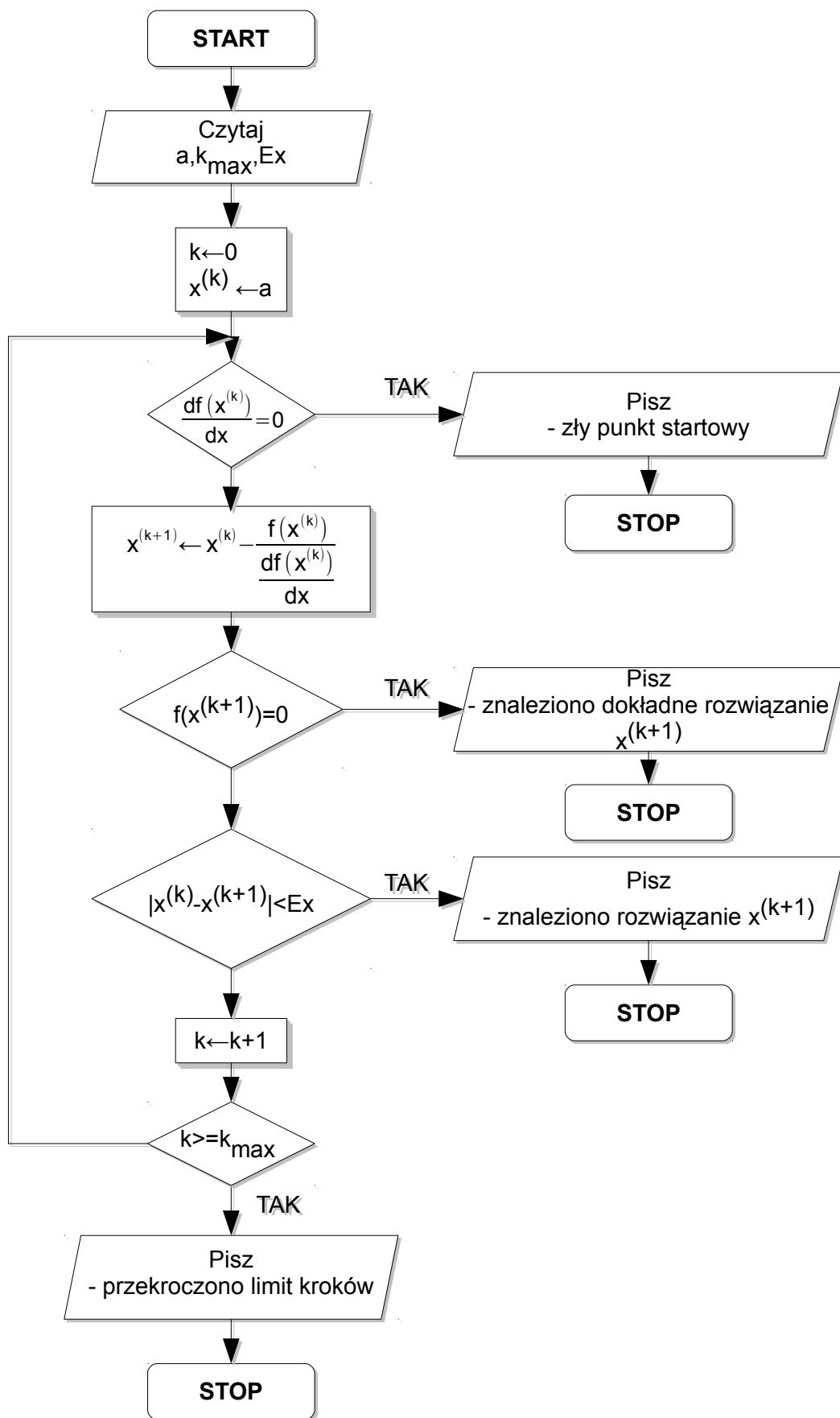
inaczej

$k=k+1$, idź do kroku 2

6. Jeśli $k \geq k_{\max}$ – przekroczono maksymalną ilość kroków, KONIEC

inaczej idź do kroku 2

2.3.2. Schemat Blokowy



3. Java i narzędzia programistyczne

3.1. Język programowania Java

Język Java jest to bardzo popularny i ceniony język programowania do tworzenia oprogramowania. Nowoczesna architektura, silne ukierunkowanie na obiektowość, bogate narzędzia wspomagające, to największe zalety języka Java. Kod programu można łatwo uruchomić na wielu różnych platformach, co przy tworzeniu aplikacji ma wielkie znaczenie bo z założenia ma ona być łatwo dostępna i niezawodna.

Przed przystąpieniem do tworzenia aplikacji w Javie należy zaopatrzyć się w niezbędne oprogramowanie, czyli JDK i JRE.

- Java Development Kit(JDK), to zestaw bibliotek i narzędzi (np. kompilator) służących do tworzenia oprogramowania w języku Java.
- Java Runtime Environment(JRE), to środowisko uruchomieniowe programu napisanego w języku Java, tzn. wirtualnej maszyny, podstawowych klas.

Kod aplikacji jest kompilowany do kodu bajtowego, który jest następnie wykonywany przez środowisko uruchomieniowe, wirtualna maszyna. Działanie takiej aplikacji jest wolniejsze od programów skompilowanych do kodu maszynowego gdyż wymaga pośrednictwa interpretera.

Aplikacja służąca do rozwiązywania równań nieliniowych składa się z obiektów, które zawierają metody i dane. Metody wykonują działania na danych, które dostarczono do obiektu np. przedział poszukiwań, kroki, równanie.

Lista obiektów odpowiedzialnych za przygotowanie procesu rozwiązywania równań nieliniowych oraz samo rozwiązywanie równań to: SilnikPrzeszukiwaniePrzedzialu, SilnikRegulaFalsi, SilnikMNewtona, StatyczneMetody.

Każda z metod rozwiązywania równań jest zaimplementowana w osobnym obiekcie, każdy obiekt zawierają trzy metody tworzące proces rozwiązywania równań nieliniowych.

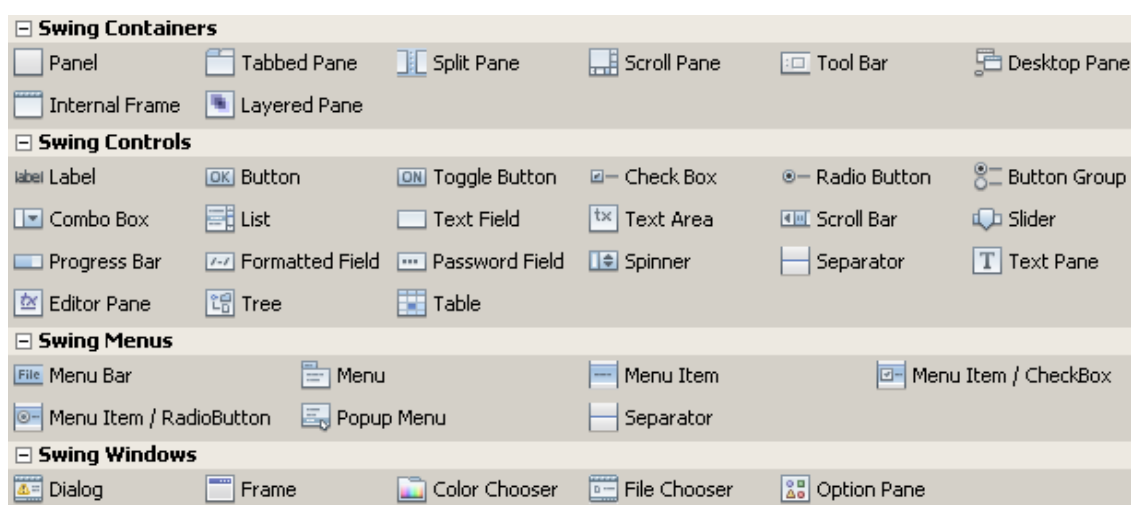
1. MkStyle() - tworzy szkielet dokumentu HTML dla wyników i kroków obliczeń. Jest uruchamiana przed rozpoczęciem wyszukiwania pierwiastka rzeczywistego funkcji.
2. Run() - uruchamia proces rozwiązywania równania.
3. Serializer() - metoda uruchamiana po zakończeniu procesu rozwiązywania równania nieliniowego. Przepisuje wektor kroków oraz wektor rozwiązań w dokument wcześniej przygotowany przez metodę MkStyle().

Czwartym obiektem to StatyczneMetody, nie zawiera on żadnych danych wejściowych, nie rozwiązuje równań żadną metodą. Zawiera natomiast statyczne metody wykorzystywane w innym obiektach, np. funkcje do liczenia równania, pochodnych, długość odcinka.

3.2. Zintegrowane środowisko programistyczne NetBeans IDE

Jest to zestaw narzędzi do szybkiego tworzenia interfejsów aplikacji oraz do programowania. Najciekawszą funkcją środowiska jest możliwość projektowania okna aplikacji oraz wszystkich jego elementów baz znajomości frameworków, bibliotek API do tworzenia interfejsów. Wszystkie elementy okna są tworzone przez kliknięcia myszką, przeciąganie elementów na płaszczyznę projektowanego okna. Bogaty zestaw elementów daje prawie nieograniczone możliwości tworzenia zaawansowanych interfejsów aplikacji.

Dodatkowo środowisko posiada wbudowany edytor wspierający proces programowania zintegrowany z kompilatorem.



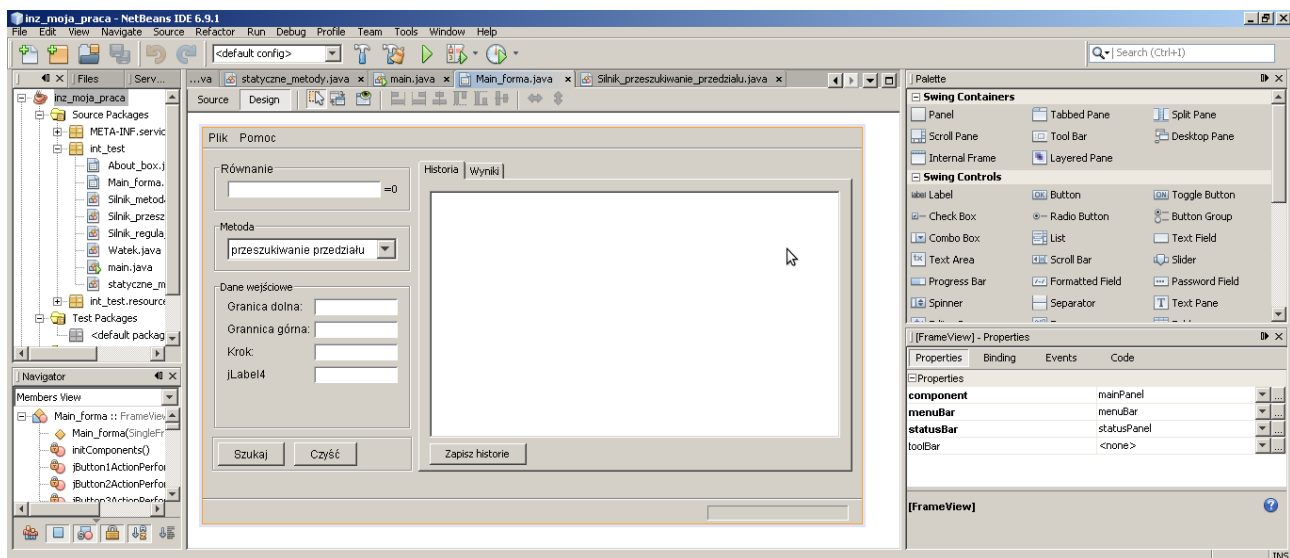
Rys. 5. Lista elementów do budowania okien

Historia aplikacji sięga drugiej połowy lat dziewięćdziesiątych dwudziestego wieku. Wówczas grupa czeskich programistów utworzyła projekt o nazwie Xelfix⁵. Początkowo była to prosta aplikacja wspomagająca prace projektowe przy tworzeniu interfejsów w aplikacjach Java, z założenia miała posiadać podobne narzędzia do Delphi, czyli wsparcie graficznego tworzenia interfejsów.

Środowisko stworzono do pracy z językiem Java, jednak później dodano moduły umożliwiające pracę z innymi językami, aktualnie wspiera większość istniejących języków programowania, np. C++, Pascal, Pearl, Python. Istnieją funkcjonalności wspierające pracę grupową np. SVN, CSV, GIT a nawet tworzenie stron internetowych w językach znaczników HTML, xHTML, XML.

NetBeans i Eclipse (konkurencyjny projekt), są to aktualnie dwa największe i najlepsze aplikacje Java wspierające proces tworzenia oprogramowania.

⁵ A Brief History of NetBeans, [Dostęp 23 kwietnia 2011]. Dostępny w Internecie: <http://netbeans.org/about/history.html>



Rys. 6. Okno projektu w Netbeans

4. Analiza wyrażeń matematycznych

4.1. JEP parser

W aplikacji zastosowano parser wyrażeń matematycznych do weryfikacji i obliczania równań matematycznych. Z wielu dostępnych narzędzi wybrano JEP⁶ - nowoczesny, oparty na wolnej licencji GNU zestaw bibliotek służący do weryfikowania i rozwiązywania równań. Pozwala on na wprowadzanie dowolnego ciągu znaków, który następnie zostanie poddany analizie. Jeśli równanie zostanie rozpoznane, można wyliczyć jego wartość. Większość typowych funkcji matematycznych i stałych wbudowanych w parser pozwala na właściwe rozpoznanie równania, co w konsekwencji umożliwia jego rozwiązanie. Parser obsługuje następujące funkcje: $\sin()$, $\cos()$, $\tan()$, $\arcsin()$, $\arccos()$, $\arctan()$, $\cot()$. Dzięki temu działanie aplikacji nie ograniczają się tylko to prostych równań, możliwe jest również rozwiązywanie równań zawierających m.in. funkcje trygonometryczne.

Poniższy kod przedstawia funkcję *liczRow()* stworzoną na potrzeby aplikacji, której zadaniem jest weryfikacja równania oraz obliczenie jej wartości. Jeśli proces analizy i rozwiązania równania zakończy się sukcesem, funkcja zwraca wynik w innym wypadku zostanie wywołany wyjątek.

```
public class statyczne {
    static double liczRow(String rownanie, double x)
    {
        JEP parser = null;
        parser = new JEP(); // tworzenie nowego obiektu
        parser.addStandardFunctions(); // dołączenie typowych funkcji
        parser.addStandardConstants(); // dołączenie typowych stałych
        parser.addVariable("x", x); // wprowadzenie zmiennej
        parser.parseExpression(rownanie); // analiza/wyliczenie równania
        double exit = parser.getValue(); // odczyt wartości równania
        if(!parser.hasError()) return exit; // zwraca wynik jeśli, lub
        throw new IllegalStateException(); // wywołuje wyjątek jeśli
    } // wystąpił błąd
}

try {
    double wynik = statyczne.liczRow("x^2+3*x-2",2); // liczenie wartości równania
    System.out.println(wynik); // wyświetlanie wyniku, 8
} catch(ParseException e) { // przechwytywanie wyjątku
    System.out.println(e.getMessage()); // wyświetlanie błędu
}
```

Rys.7. Kod aplikacji - rozwiązywanie równania z wykorzystaniem parsera JEP

⁶ Jep Documentation, [Dostęp: 21 kwietnia 2011]. Dostępny w Internecie:
<http://www.singularsys.com/jep/doc/html/index.html>.

4.2. DJEP⁷ procesor różniczkowania

Do rozwiązania równania nieliniowego metodą *Newtona* wykorzystywane są wartości funkcji oraz pochodnej. Obliczenie pochodnej z funkcji wymusiło zatem zastosowanie w aplikacji procesora różniczkowania – DJEP. Jest to rozszerzenie modułu JEP, dodaje możliwość różniczkowania równań. Narzędzie najpierw analizuje równanie a następnie zwraca wzór pochodnej z którego można wyliczyć jej wartość.

Poniższy kod prezentuje przykład wykorzystania DJEP w aplikacji. Funkcja analizuje równanie, jeśli nie wystąpią błędy, zwraca pochodną.

```
public class statyczne {
    static String pochodna(String str) {
        DJep j = new DJep();
        j.addStandardConstants();
        j.addStandardFunctions();
        j.addComplex();
        j.setAllowUndeclared(true);
        j.setAllowAssignment(true);
        j.setImplicitMul(true);
        try {
            j.addDiffRule(new MacroDiffRules(j,"sin","cos(x)"));
            j.addDiffRule(new MacroDiffRules(j,"cos","-sin(x)"));
            j.addDiffRule(new MacroDiffRules(j,"tan",
                "1/((cos(x))^2)"));
            j.addDiffRule(new MacroDiffRules(j,"sqrt",
                "1/(2 (sqrt(x)))"));
            j.addFunction("exp", new Exp());
            j.addDiffRule(new MacroDiffRules(j,"exp", "exp(x)"));
            Node node = j.parse(str);
            Node diff = j.differentiate(node, "x");
            Node simp = j.simplify(diff);
            return j.toString(simp);
        }
        catch (ParseException e) {
            System.out.println(e.getMessage());
        }
    }
}

String p = statyczne.pochodna("x^2+3*x-2");
System.out.println(p);
```

// tworzenie nowego obiektu
/* dołączenie typowych
funkcji i stałych,
przygotowanie modułu do
działania */

// określamy zachowanie
DJEP przy rozpoznaniu
funkcji trygonometrycznej

// analiza równania
// określanie niewiadomej
// upraszczanie równania
// zwracamy równanie

// przechwytywanie wyjątków
// wyświetlanie komunikatu
o błędzie

// przykład zastosowania
funkcji

Rys. 8. Kod aplikacji - różniczkowanie DJEP

⁷ DJep - Java Expression Parser (JEP), [Dostęp: 21 kwietnia 2011]. Dostępny w Internecie:
<http://www.singsurf.org/djep/index.php>

5. Bloki działania aplikacji

Działanie aplikacji podzielone jest na trzy bloki:

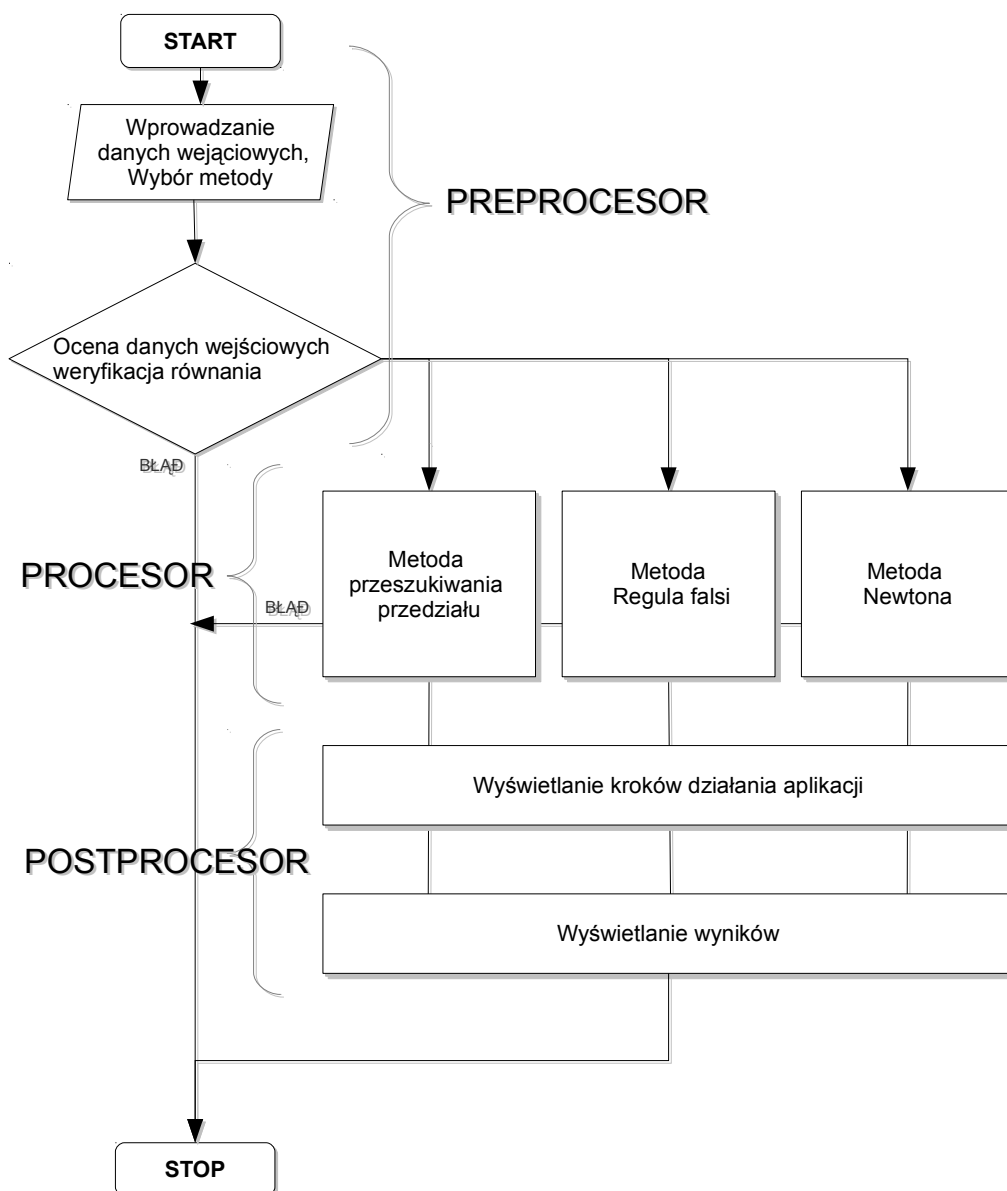
- pre-procesor,
- procesor,
- post-procesor.

Poszczególne bloki zawierają klasy tworzące cały proces rozwiązywania równań nieliniowych. Wszystkie bloki są ze sobą powiązane i należą do głównej klasy aplikacji `Main_form`, wewnątrz której są uruchamiane.

Interfejs użytkownika został zbudowany z użyciem biblioteki Swing. Wszystkie elementy głównego okna (przyciski, ramki panele, labela) tworzą klasę `Main_form`, która jest nadrzędną dla wszystkich pozostałych.

Blok pre-procesora, jest inicjowany po wprowadzeniu danych w wymagane pola formularza, wybraniu metody obliczeń i kliknięciu przycisku „Szukaj”.

Pre-procesor w czasie pracy weryfikuje zgodność typu wprowadzonych danych z wybraną metodą, sprawdza czy wprowadzona funkcja spełnia wymagania metody, np. czy jest ciągła w przedziale, czy zawiera pierwiastek rzeczywisty. Jeśli dane są nieprawidłowe, moduł odpowiada za wskazanie nieprawidłowych danych. Źle wypełnione elementy formularza są zaznaczane czerwonym kolorem, wyświetlany jest komunikat o błędzie, jeśli niezbędne są dodatkowe wyjaśnienia. Praca aplikacji jest przerywana. Natomiast jeśli wszystkie etapy weryfikacji danych są zgodne z założeniami metody, program przechodzi do etapu obliczeń - uruchamiane są metody procesora. Procesor używa wprowadzone dane jako wartości początkowe obliczeń, przystępuje do obliczeń. Wszystkie kroki pracy procesora są zapisywane w pamięci, jak również wyniki, błędy i wyjątki. Elementem procesora jest również klasa zawierająca statyczne funkcje używane we wszystkich metodach obliczeń. Odpowiada ona za liczenie długości wektora, wartości funkcji, pochodnej. Zakończenie pracy procesora wyzwala metody post-procesora, który odpowiada za wyświetlenie etapów obliczeń, wyników i błędów. Moduł ten odpowiada również za odblokowanie elementów okna, przywrócenie ich do stanu pierwotnego.



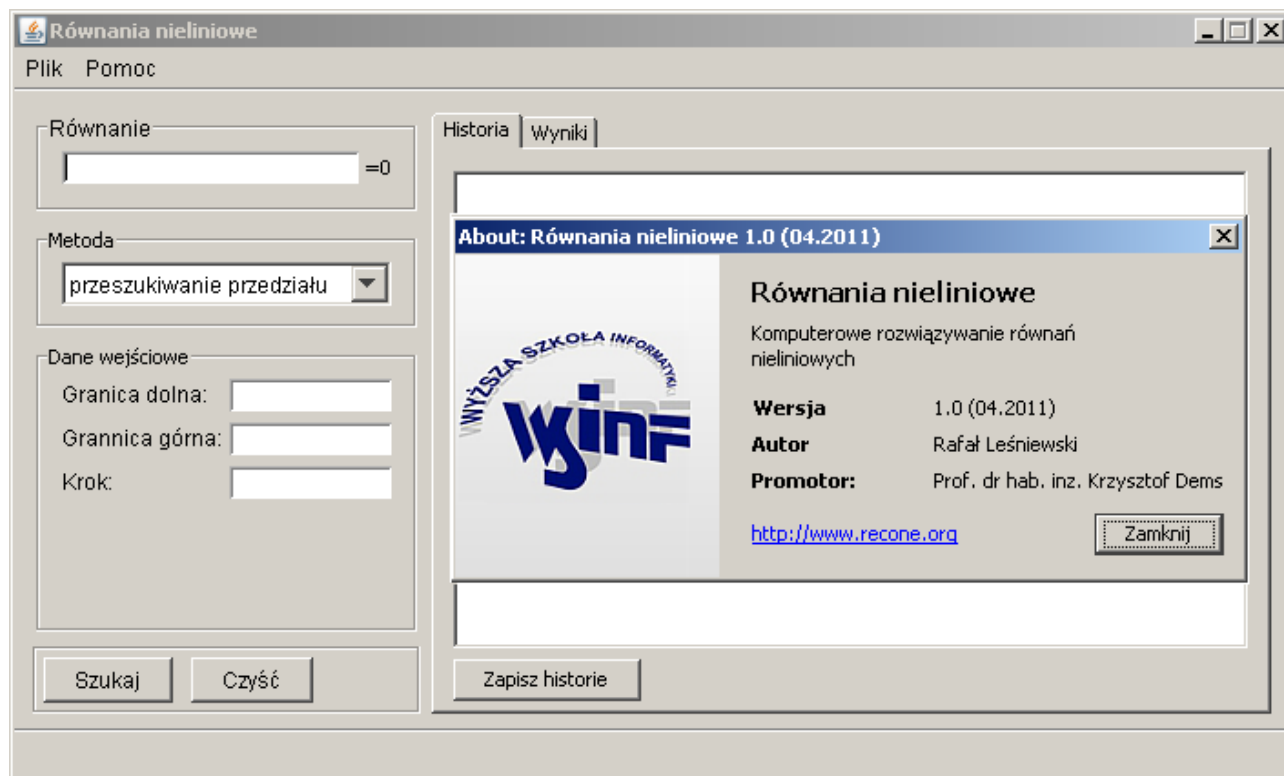
Rys. 8 Bloki działania aplikacji

6. Budowa i zastosowanie aplikacji

Budowa aplikacji

Aby aplikacja działała prawidłowo niezbędnym jest zainstalowanie w systemie operacyjnym Wirtualnej Maszyny Java (JRE). Jest to środowisko pozwalające uruchomić aplikacje napisane w języku Java.

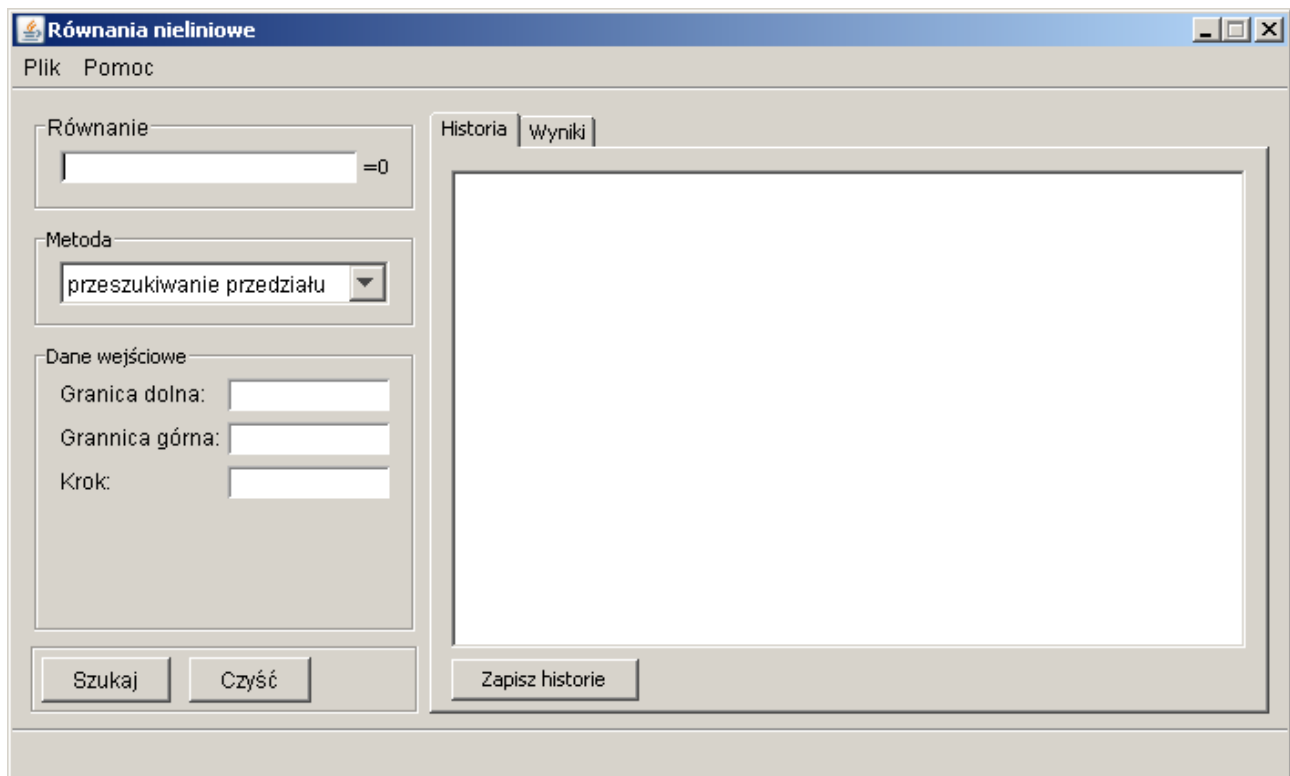
Program należy uruchomić przez standardowe kliknięcie na ikonę aplikacji. Jeżeli program nie zareaguje, należy go wywołać z wiersza poleceń (cmd) komendą: `java -jar „nazwa_pliku.jar”`. By ułatwić proces uruchamiania aplikacji stosuje się pliki wsadowe wiersza poleceń „*.bat” do uruchamiania aplikacji Java.



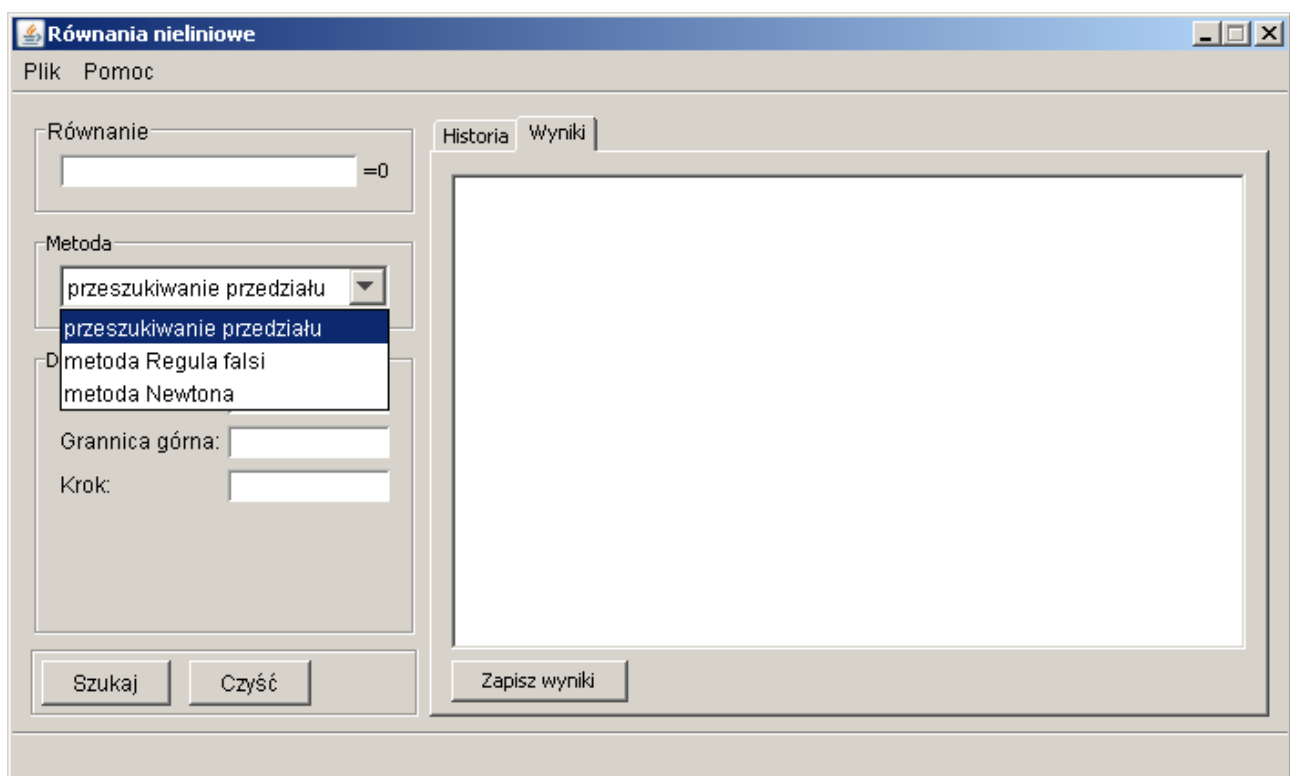
Rys. 9. Okno startowe aplikacji.

Po uruchomieniu aplikacji otwiera się oraz jej okno powitalne. Zawiera ono podstawowe informacje o aplikacji tj. numer wersji, datę kompilacji, nazwę autora aplikacji oraz nazwisko promotora. Po lewej stronie znajduje się również logo Wyższej Szkoły Informatyki w Łodzi.

Główne okno aplikacji pojawia się po zamknięciu okna powitalnego. Lewa część okna zawiera elementy formularza, w którym wprowadzamy równanie, wybieramy metodę oraz umieszczamy dane do jej zastosowania, np.: przedział poszukiwań. Właściwości pól formularza oraz ich liczba zmienia się w zależności od wyboru metody. Ich opis został przedstawiony przy opisie wykorzystania poszczególnych metod. Prawa część okna zbudowana jest z zakładek, w których wyświetlana jest historia działania aplikacji oraz wyniki.



Rys. 10. Okno główne aplikacji



Rys. 11. Wybór metody

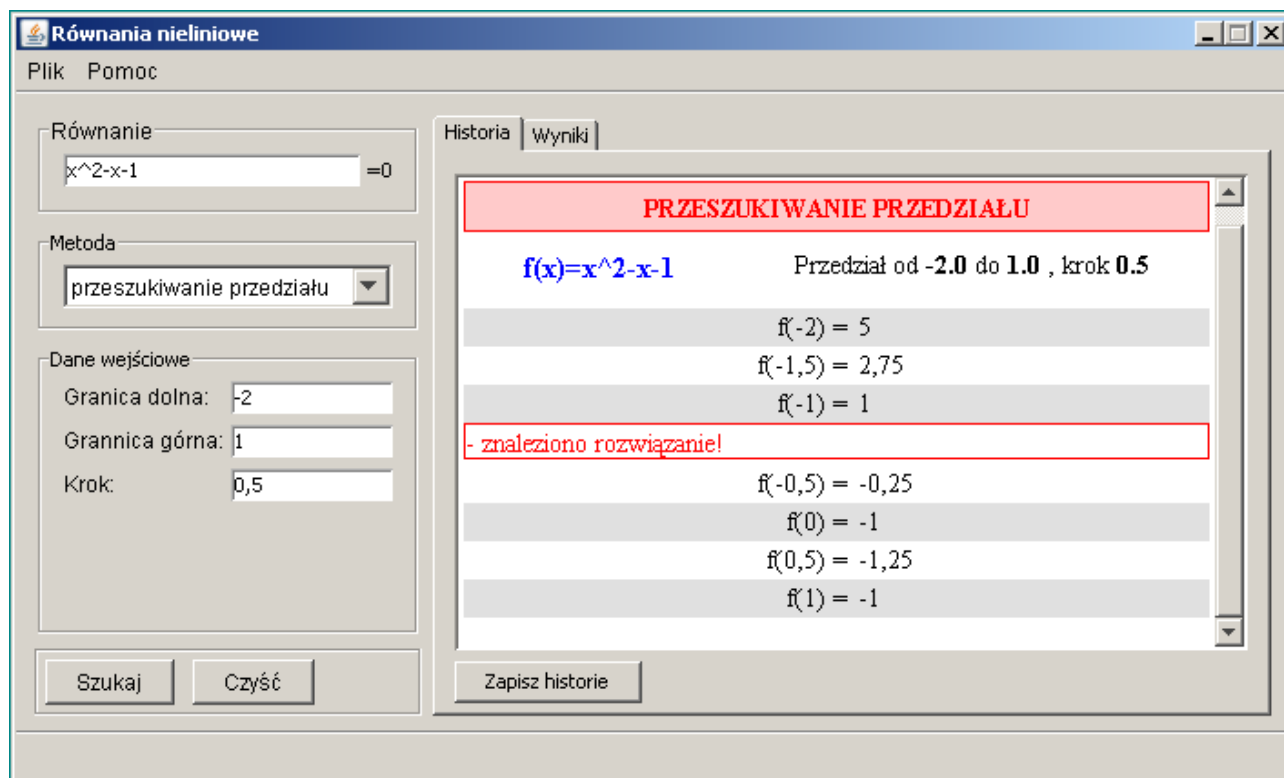
Błędy danych wejściowych

W pole *Równania* należy umieścić prawidłowy wzór równania, w którym niewiadomą jest znak „x”, jest to jedyne pole edycyjne, w którym można umieścić dowolny ciąg znaków. W pozostałych polach do edycji można wpisać tylko liczbę.

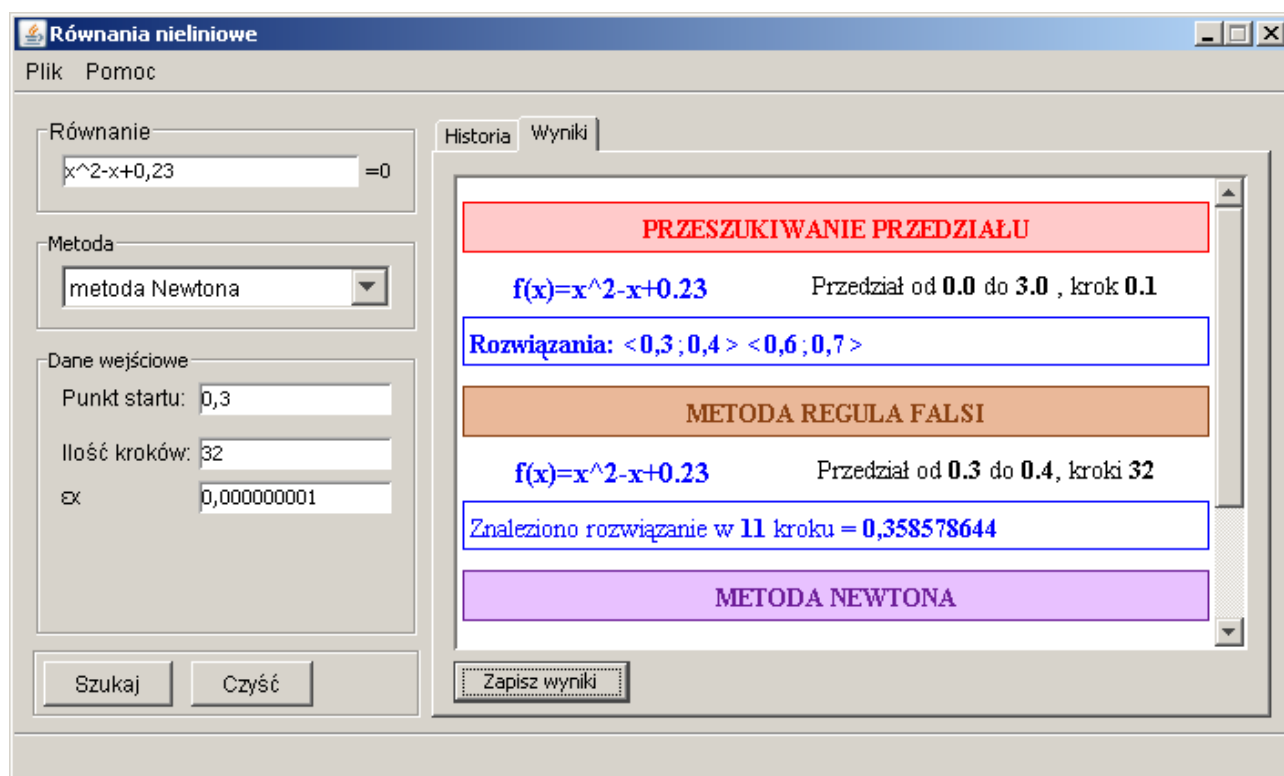
Pola formularza, które zostały niepoprawnie wypełnione bądź w których nie umieszczono wymaganych danych, zostaną zaznaczane czerwonym obramowaniem. Wypełnienie formularza danymi niezgodnymi ze specyfikacją metody również spowoduje zaznaczenie elementu formularza. Przy wprowadzaniu liczb w pola edycyjne należy pamiętać, że liczby rzeczywiste wprowadzamy z przecinkiem, wprowadzenie kropki lub innego znaku spowoduje odcięcie części dziesiętnej wprowadzonej wartości.

Rys. 12. Wykrywanie błędów formularza

Kroki działania metody oraz wynik są wyświetlane w zakładkach *Historia* i *Wynik*. Pierwsza z zakładek zawiera listę wykonanych obliczeń, etapy poszukiwania rozwiązania. Treść elementu jest sformatowana, nazwy metod są umieszczane w ramce, każda metoda wyróżniona jest innym kolorem. Pod danymi wyświetlane są etapy działania metody, każde wywołanie metody powoduje dopisanie nowych informacji. Odnalezienie wyniku powoduje dodanie wpisu w zakładce *Wyniki*, która zawiera informacje o metodzie, jej dane wejściowe oraz wynik.



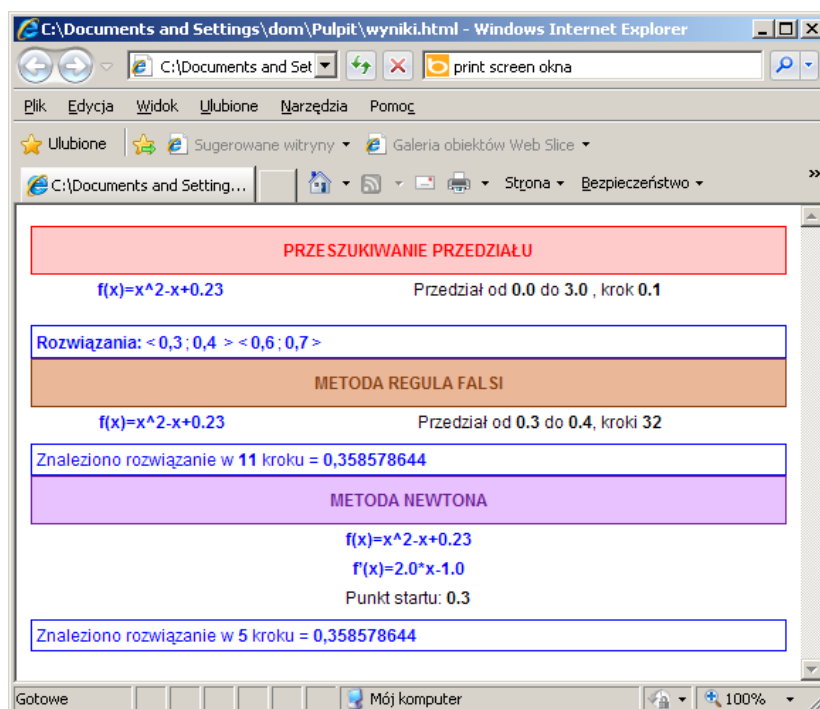
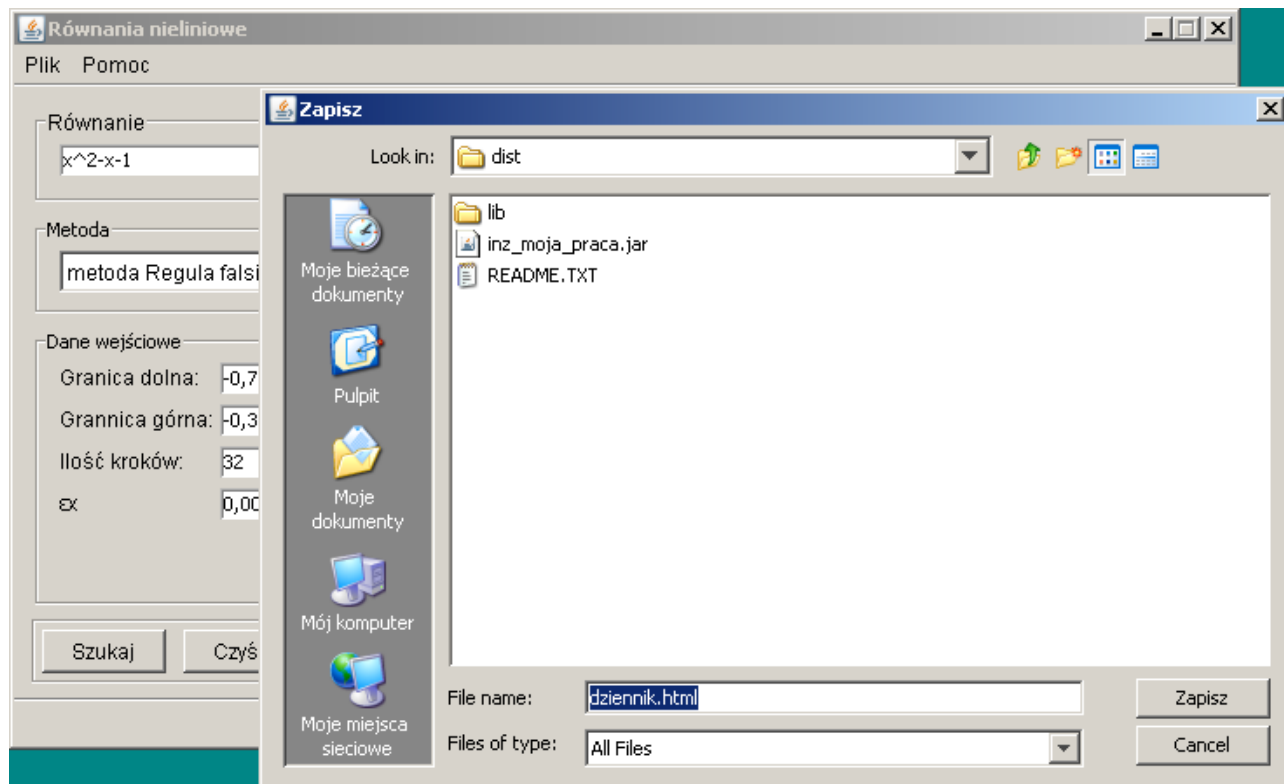
Rys. 13. Zakładka *Historia*



Rys. 14. Zakładka *Wyniki*

Zapisywanie do pliku

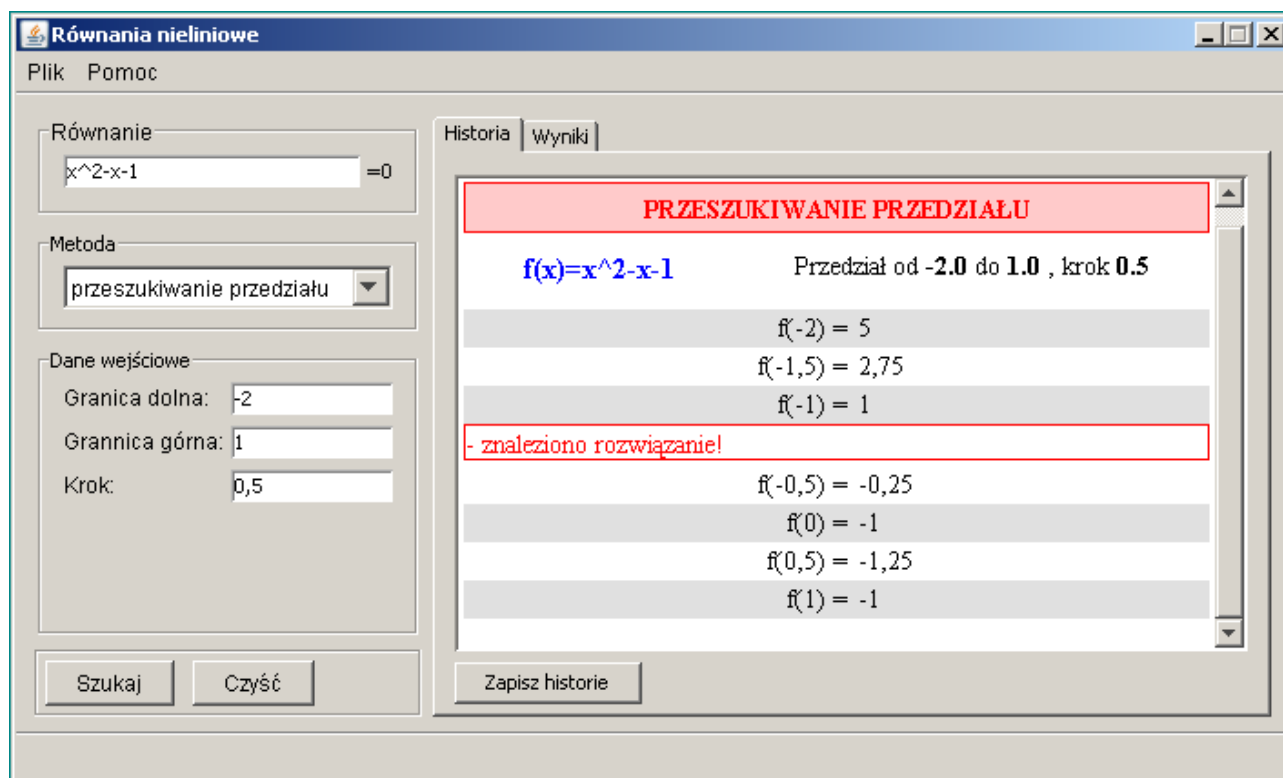
Treści wyświetlane w zakładkach *Historia* i *Wyniki* mogą zostać zapisane do pliku. Wywołanie funkcji *Zapisz historię lub Zapisz wynik* spowoduje wywołanie okna przygotowującego proces zapisywania danych do pliku. Program pozwala na zapisanie treści zakładek do sformatowanego pliku html, otwieranego przez przeglądarkę internetową.



Rys. 15. Wyniki zapisane w pliku html

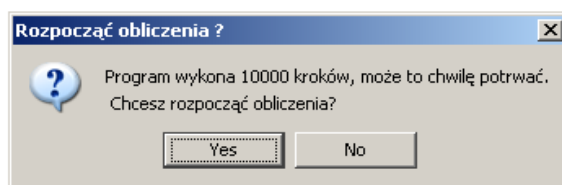
6.1. Przygotowanie procesu wyszukiwania metodą przeszukiwania przedziału

Przed rozpoczęciem procesu wyszukiwania pierwiastków metodą przeszukiwania przedziału należy wprowadzić równanie, wybrać tę metodę z rozwijanej listy. Dane wejściowe dla tej metody to: granice przedziału (dolna i górna) oraz krok przeszukiwania. Wartość dolna przedziału musi być mniejsza od górnej, krok odpowiednio mały.

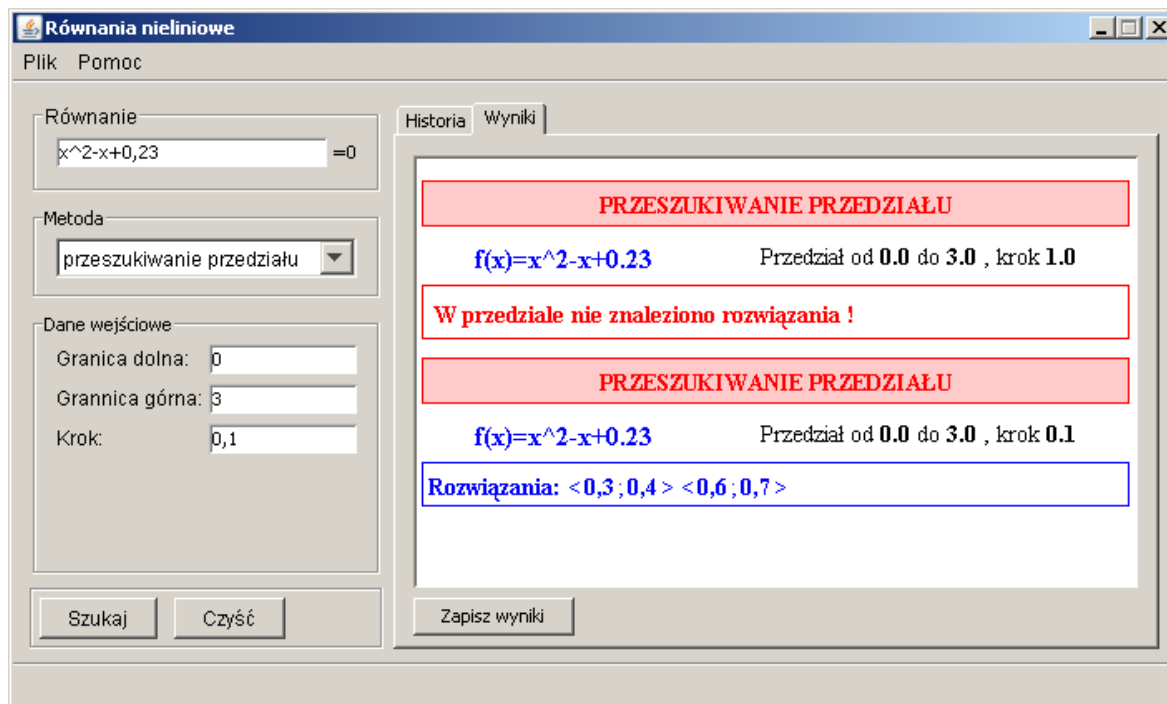


Rys. 16. Przeszukiwanie przedziału

Wprowadzenie zbyt dużego kroku może spowodować pominięcie części pierwiastków w przedziale, natomiast zbyt mały krok wymusza wykonanie dużej liczby obliczeń, może zwolnić działanie aplikacji a nawet ją zawiesić. Komunikat informuje użytkownika o przypuszczalnej liczbie obliczeń, możliwym zawieszeniu programu.

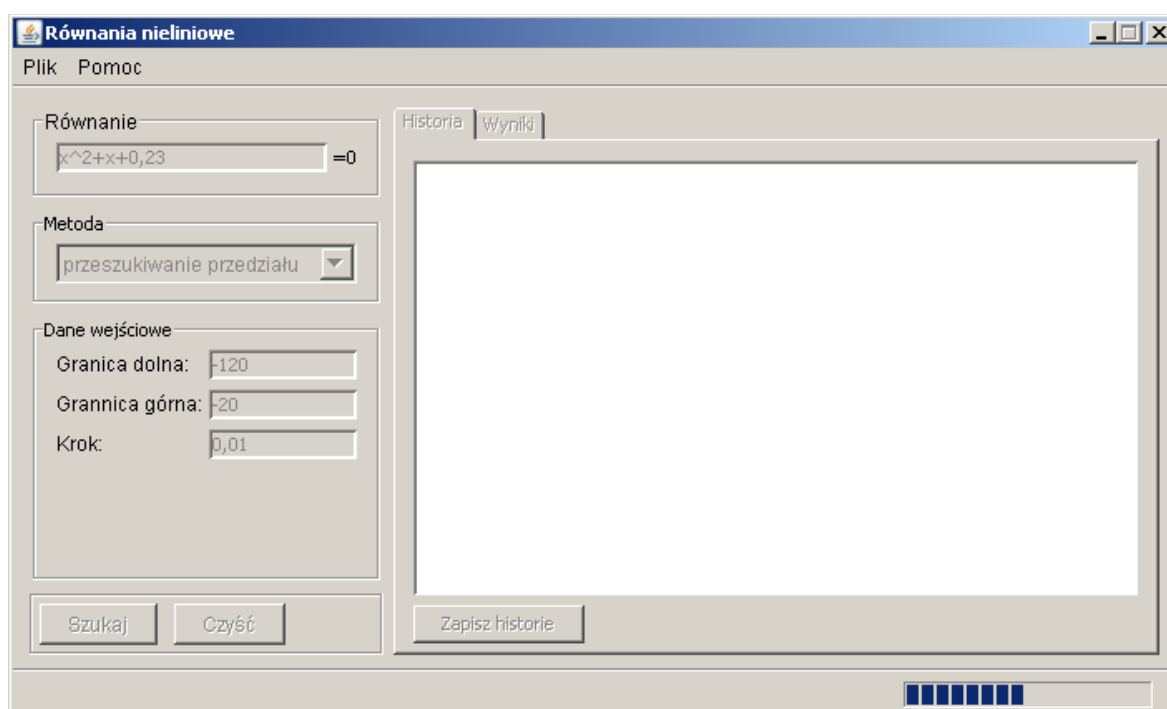


Rys. 17. Okno informujące o dużej liczbie obliczeń



Rys. 18. Pominięcie wyniku, zbyt duży krok

W czasie wykonywania obliczeń, o działaniu aplikacji informuje animowany pasek stanu. Zatrzymanie animacji paska stanu informuje użytkownika o przerwaniu procesu wyszukiwania pierwiastków, może również oznaczać zawieszenie aplikacji.



Rys. 19. Wyszukiwanie pierwiastków równania

Wszystkie aktywne elementy programu w czasie szukania rozwiązania są zablokowane. Użytkownik nie ma możliwości wprowadzać zmian. Po zakończeniu procesu okno wraca do swojego pierwotnego stanu, efekty pracy są wyświetlane w zakładkach *Historia* i *Wyniki*.

6.2. Przygotowanie procesu wyszukiwania metodą Regula falsi

Metoda *Regula falsi* należy do metod uściślenia rozwiązania, jest wykorzystywana do odnajdywania dokładnej wartości pierwiastka w przedziale o którym wiemy, że zawiera rozwiązanie równania. Przedział można określić metodą przeszukiwania przedziału.

The screenshot shows a software window titled "Równania nieliniowe" with a menu bar containing "Plik" and "Pomoc". The interface is divided into several sections:

- Równanie:** A text box containing the equation $x^2 - x - 1 = 0$.
- Metoda:** A dropdown menu set to "metoda Regula falsi".
- Dane wejściowe:** Input fields for:
 - Granica dolna: -0,7
 - Granica górna: -0,3
 - Ilość kroków: 32
 - ϵ : 0,000000001
- Buttons:** "Szukaj", "Czyść", and "Zapisz historie".
- History/Wyniki:** A tabbed area showing the results of the calculation.

The "Wyniki" tab displays the following information:

- METODA REGULA FALSI**
- $f(x) = x^2 - x - 1$**
- Przedział od -0.7 do -0.3, kroki 32**
- A list of 7 values:

1.	= -0,605
2.	= -0,617570499
3.	= -0,618017596
4.	= -0,618033409
5.	= -0,618033968
6.	= -0,618033988
7.	= -0,618033989

Rys. 20. Uściślenie rozwiązania metodą Regula falsi.

Metoda do działania wymaga podania małego przedziału poszukiwań (granica dolna i górna), ilości dozwolonych kroków oraz określenie dokładności wyniku. Wszystkie pola są przeznaczone do wprowadzania liczb. Standardowa ilość kroków (32) jest zazwyczaj wystarczająca, jednak można ją dobierać zgodnie z wymaganiami dokładności wyniku. Przekroczenie liczby kroków powoduje zatrzymanie procesu, wyświetlenie komunikatu. Zabezpiecza to program przed zapętleniem i zawieszeniem procesu wyszukiwania rozwiązania.

Równania nieliniowe

Plik Pomoc

Równanie
 $x^2 - x - 1 = 0$

Metoda
metoda Regula falsi

Dane wejściowe
Granica dolna: -0,7
Grannica górna: -0,3
Ilość kroków: 32
 ϵ : 0,000000001

Szukaj Czyść

Historia Wyniki

METODA REGULA FALSI

$f(x) = x^2 - x - 1$ Przedział od -0.7 do -0.3, kroki 32

Znaleziono rozwiązanie w 7 kroku = -0,618033989

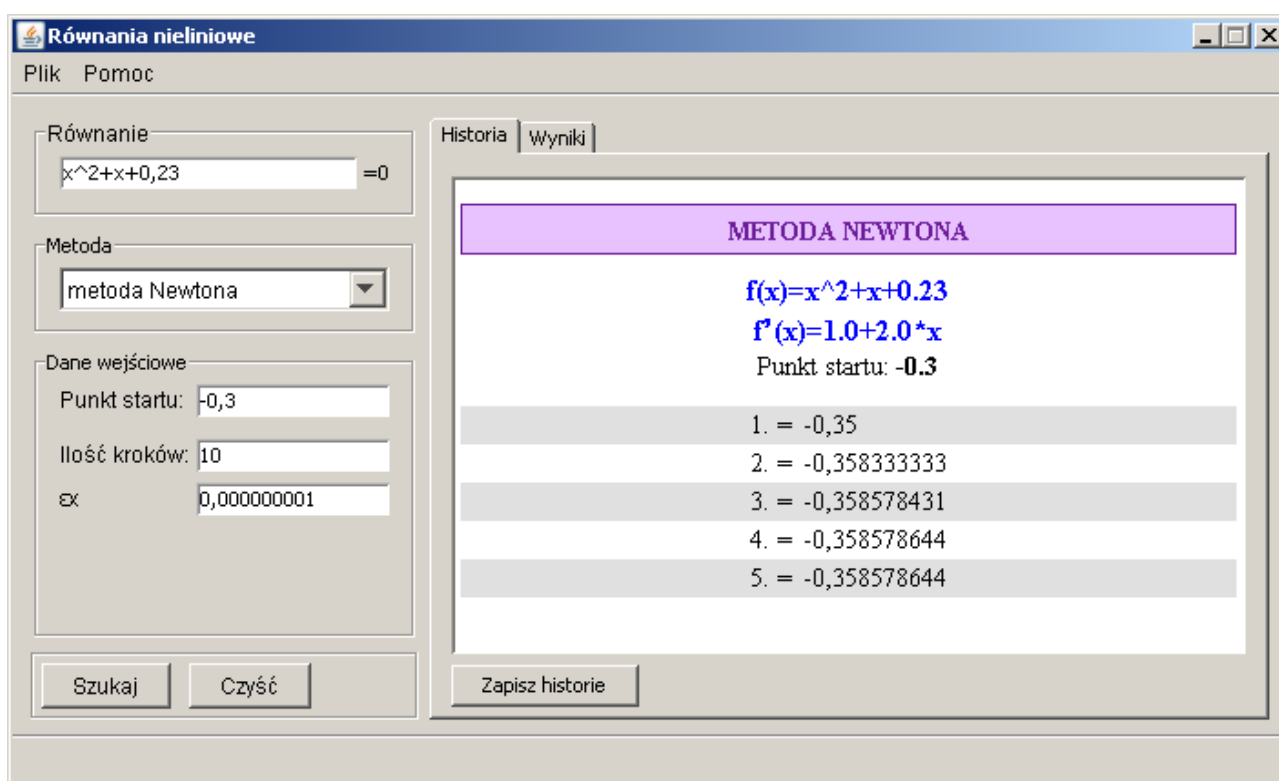
Zapisz wyniki

Rys. 21. Widok z dokładnym wynikiem działania metody

6.3. Przygotowanie procesu wyszukiwania metodą Newtona

Metoda Newtona, podobnie jak metoda *Regula falsi*, należy do metod uściślenia rozwiązania, czyli do obliczania dokładnej wartości pierwiastka rzeczywistego funkcji.

W danych wejściowych wprowadzamy trzy wartości, tzn. punkt startowy, ilość dozwolonych kroków i żadaną dokładność wyniku. Określanie przedziału w metodzie nie jest wymagane, gdyż kierunek poszukiwania rozwiązania jest determinowany przez wartości funkcji i jej pierwszej pochodnej. Odległość punktu startowego od rozwiązania oraz wymagana dokładność rozwiązania wpływa na ilość obliczeń niezbędnych do uzyskania rozwiązania. Etapy odnajdywania rozwiązania oraz samo rozwiązanie są wyświetlane w zakładkach.



Rys. 22. Uściślanie rozwiązania metodą Newtona

Przed zapętlaniem aplikacji w przypadku oddalania się od wyniku zabezpiecza ograniczona liczba uściśleń wyniku. Osiągnięcie maksimum powoduje przerwanie procesu wyszukiwania rozwiązania.

Równania nieliniowe

Plik Pomoc

Równanie: =0

Metoda:

Dane wejściowe:

Punkt startu:

Ilość kroków:

ϵ :

Szukaj Czyść

Historia Wyniki

METODA NEWTONA

$f(x)=x^2+x+0.23$
 $f'(x)=1.0+2.0*x$
Punkt startu: -0.3

Znaleziono rozwiązanie w 5 kroku = -0,358578644

Zapisz wyniki

Rys. 23. Wynik działania metody Newtona

7. Testy i porównania

W pracy opisano algorytmy numeryczne wykorzystywane do rozwiązywania równań nieliniowych. Metody te zostały zaimplementowane w aplikacji, która rozwiązuje funkcje nieliniowe metodami numerycznymi. Program został przetestowany a wyniki zanalizowano.

Testy prezentują działanie aplikacji oraz zawartych w niej metod, dowodzą zgodności zaimplementowanych algorytmów z teoretycznymi założeniami metod numerycznych użytych w pracy.

Dodatkowym celem wykonanych testów jest przedstawienie różnic dzielących użyte metody, wskazanie ich wad oraz zalet w różnych konfiguracjach.

Do analizy wykorzystano trzy równania:

$$2x^3 - 9.06843x^2 - 72.3753x + 114.834 = 0$$

$$x^2 \sin(2x) - x^3 - 16 \sin(2x) + 16x = 0$$

$$(x - \sqrt{3}) 2.54^{-x} = 0$$

Równania w formie akceptowalnej przez parser:

$$2*x^3-9,06843*x^2-72,3753*x+114,834 = 0$$

$$x^2*\sin(2*x)-x^3-16*\sin(2*x)+16*x = 0$$

$$(x-\text{sqrt}(3))*2,54^(-x) = 0$$

Każde z równań zostało rozwiązane trzema użytymi w pracy metodami, tzn: *przeszukiwania przedziału*, *Regula falsi* i *metodą Newtona*. W pierwszym etapie użyto metodę przeszukiwania, odnalezione przedziały z rozwiązaniami wykorzystano do uściślenia wyników pozostałymi metodami. W testach metody *Regula falsi* i *Newtona* używane są identyczne dane wejściowe, tzn. przedział poszukiwań i dokładność wyniku. Dla *metody Newtona* punktem startu jest mniejsza wartość krawędzi przedziału.

7.1. Badanie pierwszego równania

Metoda przeszukiwania

Badane równanie $2x^3 - 9.06843x^2 - 72.3753x + 114.834 = 0$, przedział przeszukiwania $<-20, 20>$

Metodę zbadano wykonując trzy przeszukiwania przedziału, za każdym razem wprowadzając inną wartość kroku.

Krok	Liczba iteracji	Przedziały zawierające pierwiastek rzeczywisty funkcji
2,6	15	$<-7; -4,4>$ $<0,8; 3,4>$ $<6; 8,6>$
1,33	30	$<-5,37; -4,04>$ $<1,28; 2,61>$ $<7,93; 9,26>$
0,6	66	$<-5; -4,4>$ $<1; 1,6>$ $<7,6; 8,2>$

W procesie przeszukiwania odnaleziono trzy przedziały zawierające rozwiązanie funkcji, zmiana kroków na mniejsze nie wykazała wcześniej pominiętych rozwiązań. Dalsze zmniejszanie kroku poniżej wartości 0,6 jest bezcelowe, metoda generuje zbyt dużo iteracji.



Rys. 24. Ilustracja przebiegu badanej funkcji

Wybieramy pierwszy odnaleziony przedział zawierający rozwiązanie $<-7; -4,4>$ i przystępujemy do uściślenia wyniku.

Metoda *Regula falsi*

Badane równanie $2x^3 - 9.06843x^2 - 72.3753x + 114.834 = 0$

1. Przedział poszukiwania rozwiązania $< -7; -4,4 >$

Dokładność wyniku	Liczba iteracji	Pierwiastek rzeczywisty funkcji nieliniowej $f(x)=0$
0,1	3	-4,973955988
0,00001	11	-4,999994001
0,000000001	20	-4,999998514

2. Przedział poszukiwania rozwiązania $< -5,37; -4,04 >$

Dokładność wyniku	Liczba iteracji	Pierwiastek rzeczywisty funkcji nieliniowej $f(x)=0$
0,1	2	-4,992499613
0,00001	6	-4,9999982
0,000000001	10	-4,999998515

3. Przedział poszukiwania rozwiązania $< -5; -4,4 >$

Dokładność wyniku	Liczba iteracji	Pierwiastek rzeczywisty funkcji nieliniowej $f(x)=0$
0,1	2	-4,999998515
0,00001	2	-4,999998515
0,000000001	3	-4,999998515

Metoda Newtona

Badane równanie $2x^3 - 9.06843x^2 - 72.3753x + 114.834 = 0$

1. Punkt startowy -7

Dokładność wyniku	Liczba iteracji	Pierwiastek rzeczywisty funkcji nieliniowej $f(x)=0$
0,1	3	-5,000728817
0,00001	5	-4,999998515
0,000000001	6	-4,999998515

2. Punkt startowy -5,37

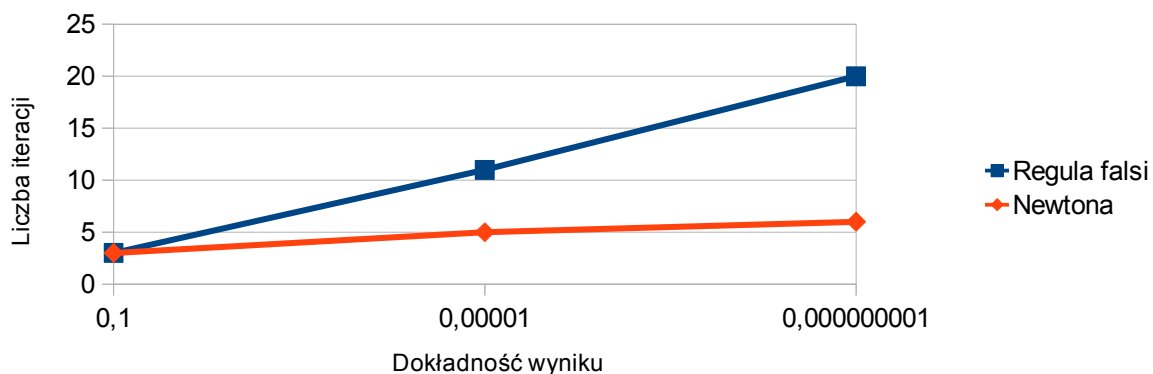
Dokładność wyniku	Liczba iteracji	Pierwiastek rzeczywisty funkcji nieliniowej $f(x)=0$
0,1	2	-5,000179061
0,00001	4	-4,999998515
0,000000001	5	-4,999998515

3. Punkt startowy -5

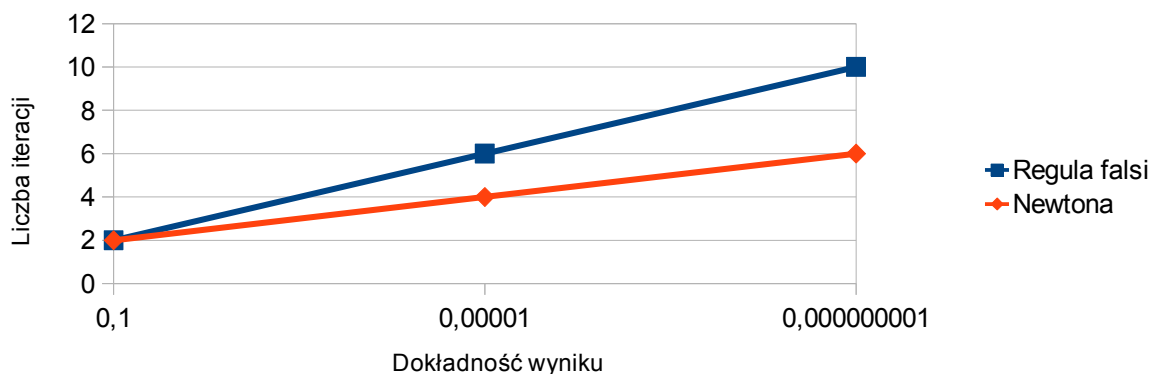
Dokładność wyniku	Liczba iteracji	Pierwiastek rzeczywisty funkcji nieliniowej $f(x)=0$
0,1	1	-4,999998515
0,00001	1	-4,999998515
0,000000001	2	-4,999998515

Porównania efektywności metod

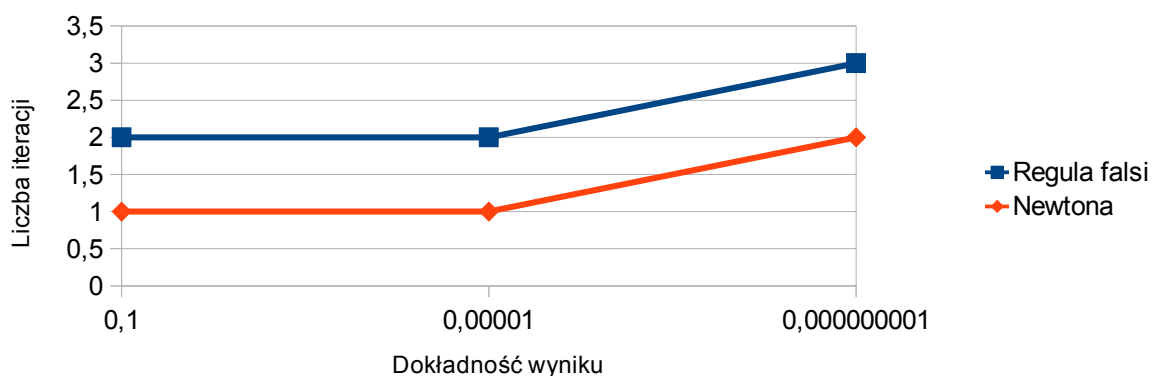
1. Pierwsza próba $<-7; -4,4>$



2. Druga próba $<-5,37; -4,04>$



3. Trzecia próba $<-5; -4,4>$



Na podstawie przeprowadzonych analiz i porównań można stwierdzić, że *metoda Newtona* jest efektywniejsza, wymaga mniejszej liczby iteracji, szybciej odnajduje pierwiastek. Różnica w szybkości odnajdywania dokładnego wyniku jest duża w szerokim przedziale poszukiwań, zaciera się gdy przedział poszukiwań jest odpowiednio mały.

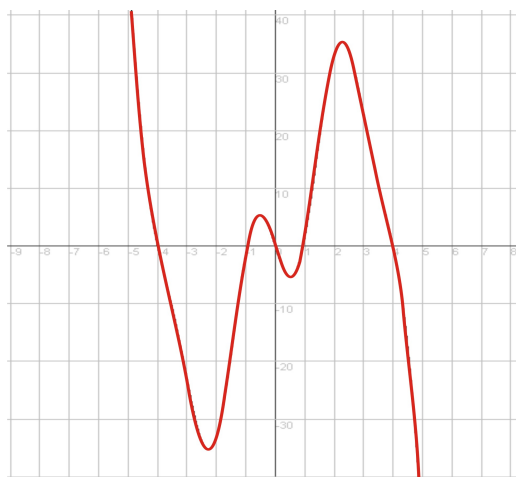
7.2. Badanie drugiego równania

Metoda przeszukiwania

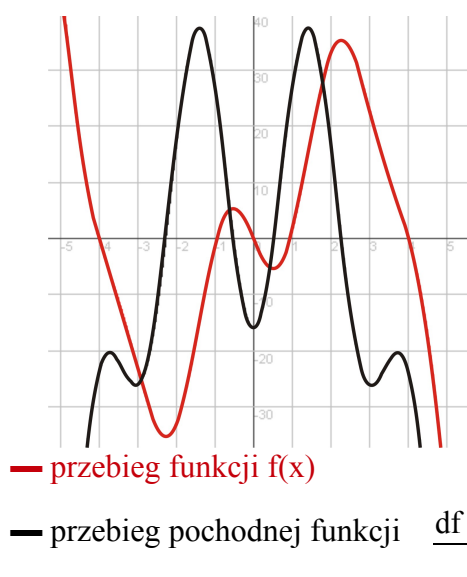
Badane równanie, $x^2 \sin(2x) - x^3 - 16 \sin(2x) + 16x = 0$, przedział przeszukiwania $\langle -20, 20 \rangle$

Krok	Liczba iteracji	Przedziały zawierające pierwiastek rzeczywisty funkcji
2,7	15	$\langle -6,5; -3,8 \rangle \langle -1,1; 1,6 \rangle \langle 1,6; 4,3 \rangle$
1,35	30	$\langle -5,15; -3,8 \rangle \langle 0,25; 1,6 \rangle \langle 2,95; 4,3 \rangle$
0,65	66	$\langle -4,4; -3,75 \rangle \langle -1,15; -0,5 \rangle \langle -0,5; 0,15 \rangle \langle 0,8; 1,45 \rangle \langle 3,4; 4,05 \rangle$

Podczas przeszukiwania przedziału z dużym krokiem część rozwiązań równania została pominięta. Zmiana kroku na mniejszą wartość pozwoliła odkryć lokalizację wszystkich pierwiastków w badanym przedziale. Pierwsze odnalezione przedziały zostaną użyte do uściślenia wyniku pozostałymi metodami.



Rys. 25. Ilustracja przebiegu badanej funkcji



Rys. 26. Ilustracja przebiegu badanej funkcji i jej pochodnej

Metoda *Regula falsi*

Badane równanie $x^2 \sin(2x) - x^3 - 16 \sin(2x) + 16x = 0$

1. Przedział poszukiwania rozwiązania $< -6,5 ; -3,8 >$

Dokładność wyniku	Liczba iteracji	Pierwiastek rzeczywisty funkcji nieliniowej $f(x)=0$
0,1	1	-3,872730054
0,00001	21	-3,999989683
0,000000001	40	-3,999999999

2. Przedział poszukiwania rozwiązania $< -5,15 ; -3,8 >$

Dokładność wyniku	Liczba iteracji	Pierwiastek rzeczywisty funkcji nieliniowej $f(x)=0$
0,1	1	-3,889438281
0,00001	17	-3,999990925
0,000000001	33	-3,999999999

3. Przedział poszukiwania rozwiązania $< -4,4 ; -3,75 >$

Dokładność wyniku	Liczba iteracji	Pierwiastek rzeczywisty funkcji nieliniowej $f(x)=0$
0,1	2	-3,986246164
0,00001	8	-3,999996779
0,000000001	15	-4

Testy przeprowadzone i opisane powyżej zostały zakończone pozytywnie. Zwiększenie dokładności szukanego rozwiązania w szerokim przedziale znacząco pogorszyło szybkość dochodzenia do rozwiązania.

Metoda Newtona

Badane równanie $x^2 \sin(2x) - x^3 - 16 \sin(2x) + 16x = 0$

1. Punkt startowy -6,5

Dokładność wyniku	Liczba iteracji	Pierwiastek rzeczywisty funkcji nieliniowej $f(x)=0$
0,1	3	-4,000151871
0,00001	4	-4
0,000000001	4	-4

2. Punkt startowy -5,15

Dokładność wyniku	Liczba iteracji	Pierwiastek rzeczywisty funkcji nieliniowej $f(x)=0$
0,1	3	-4,006816688
0,00001	6	-4
0,000000001	6	-4

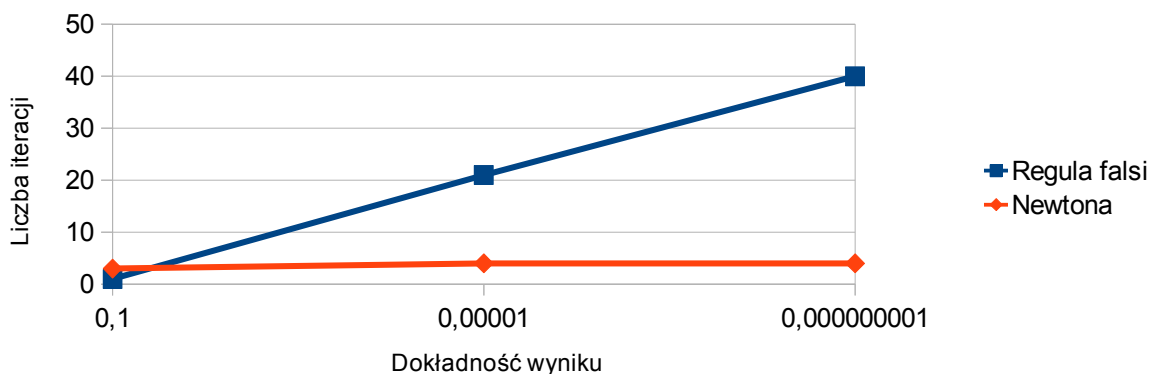
3. Punkt startowy -4,4

Dokładność wyniku	Liczba iteracji	Pierwiastek rzeczywisty funkcji nieliniowej $f(x)=0$
0,1	2	-4,005837982
0,00001	4	-4
0,000000001	4	-4

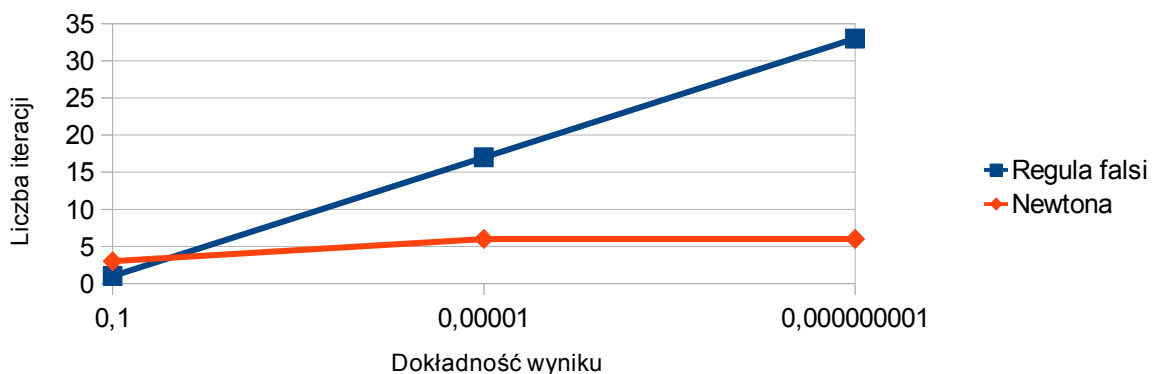
Wszystkie testy przebiegły prawidłowo, szybko odnaleziono pierwiastek rzeczywisty w przedziałach, szukanie dokładnego wyniku nie spowodowało znaczącego wzrostu iteracji.

Porównania efektywności metod

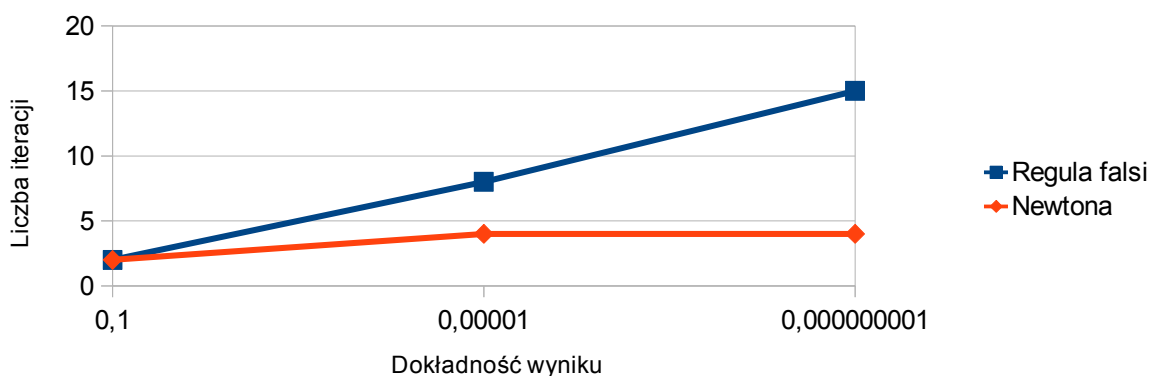
1. Pierwsza próba $<-6,5 ; -3,8>$



2. Druga próba $<-5,15 ; -3,8>$



3. Trzecia próba $<-4,4 ; -3,75>$



Dla danych wejściowych użytych do rozwiązywania funkcji metoda *Newtona* okazała się szybko zbieżna, we wszystkich próbach była szybsza, wymagała dużo mniejszej liczby iteracji od metody *Regula fałsi*.

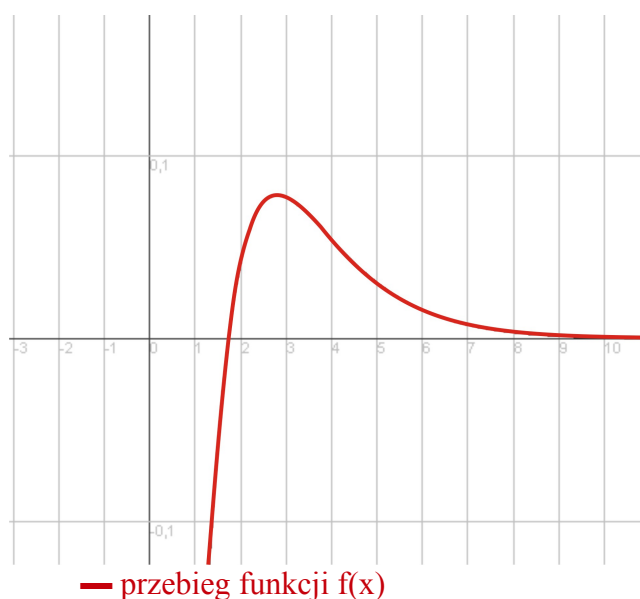
7.3. Badanie trzeciego równania

Metoda przeszukiwania

Badane równanie, $(x - \sqrt{3})2.54^{-x} = 0$, przedział przeszukiwania $<-20,3>$

Krok	Liczba iteracji	Przedziały zawierające pierwiastek rzeczywisty funkcji
1,01		$< 1,21 ; 2,22 >$
0,51		$< 1,42 ; 1,93 >$
0,25		$< 1,5 ; 1,75 >$

Przeszukiwanie przedziału z różnymi krokami wykazało, że w przedziale znajduje się jeden pierwiastek rzeczywisty funkcji. Podczas testów nie wykryto pominiętych przez duży krok pierwiastków funkcji.



Rys. 27. Ilustracja przebiegu badanej funkcji

Jak pokazuje ilustracja funkcja dąży do ciągłości w zerze, co uniemożliwia zastosowanie metody *Newrona* i *Regula Falsi* dla $x > 3$.

Metoda *Regula falsi*

Badane równanie $(x - \sqrt{3})2.54^{-x} = 0$

1. Przedział poszukiwania rozwiązania $< 1,21 ; 2,22 >$

Dokładność wyniku	Liczba iteracji	Pierwiastek rzeczywisty funkcji nieliniowej $f(x)=0$
0,1	3	1,767653958
0,00001	13	1,732053422
0,000000001	22	1,732050808

2. Przedział poszukiwania rozwiązania $< 1,42 ; 1,93 >$

Dokładność wyniku	Liczba iteracji	Pierwiastek rzeczywisty funkcji nieliniowej $f(x)=0$
0,1	2	1,745877234
0,00001	9	1,732051716
0,000000001	15	1,732050808

3. Przedział poszukiwania rozwiązania $< 1,5 ; 1,75 >$

Dokładność wyniku	Liczba iteracji	Pierwiastek rzeczywisty funkcji nieliniowej $f(x)=0$
0,1	1	1,735566648
0,00001	6	1,732051788
0,000000001	12	1,732050808

Metoda Newtona

Badane równanie $(x - \sqrt{3})2.54^{-x} = 0$

1. Punkt startowy 1,21

Dokładność wyniku	Liczba iteracji	Pierwiastek rzeczywisty funkcji nieliniowej $f(x)=0$
0,1	3	1,731547844
0,00001	5	1,732050808
0,000000001	5	1,732050808

2. Punkt startowy 1,42

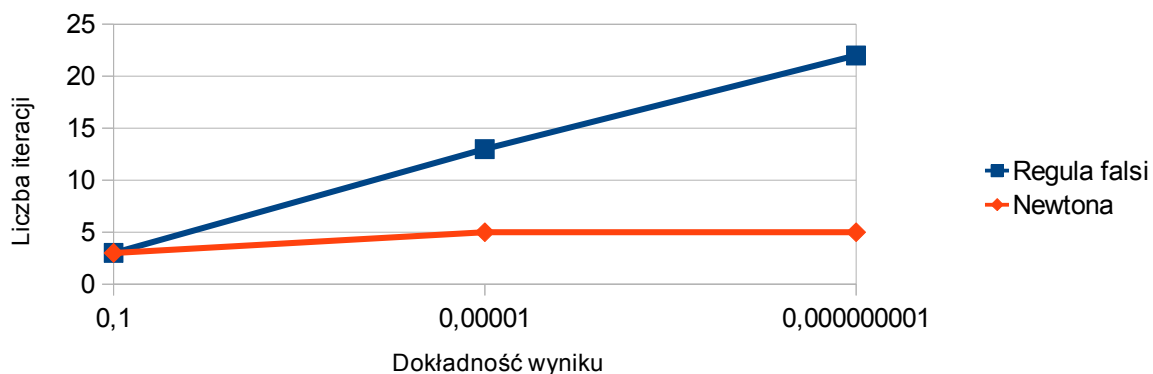
Dokładność wyniku	Liczba iteracji	Pierwiastek rzeczywisty funkcji nieliniowej $f(x)=0$
0,1	2	1,727725345
0,00001	4	1,732050808
0,000000001	4	1,732050808

3. Punkt startowy 1,5

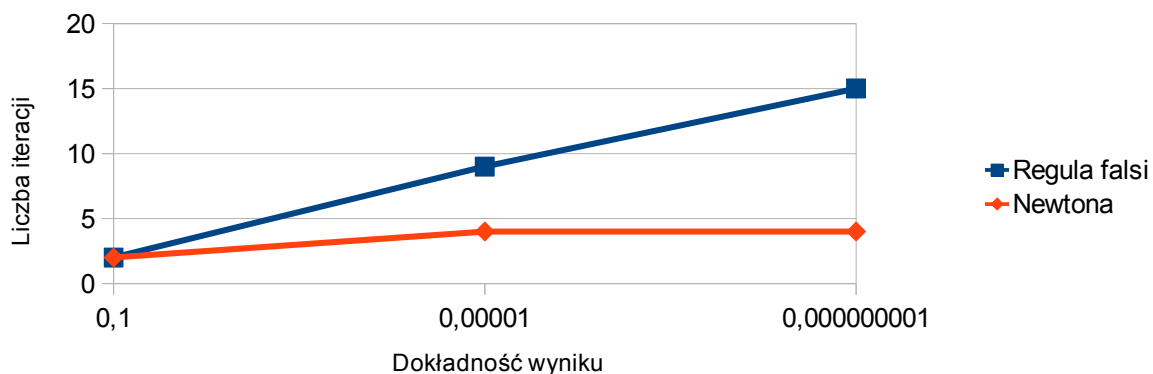
Dokładność wyniku	Liczba iteracji	Pierwiastek rzeczywisty funkcji nieliniowej $f(x)=0$
0,1	2	1,730522088
0,00001	4	1,732050808
0,000000001	4	1,732050808

Porównania efektywności metod

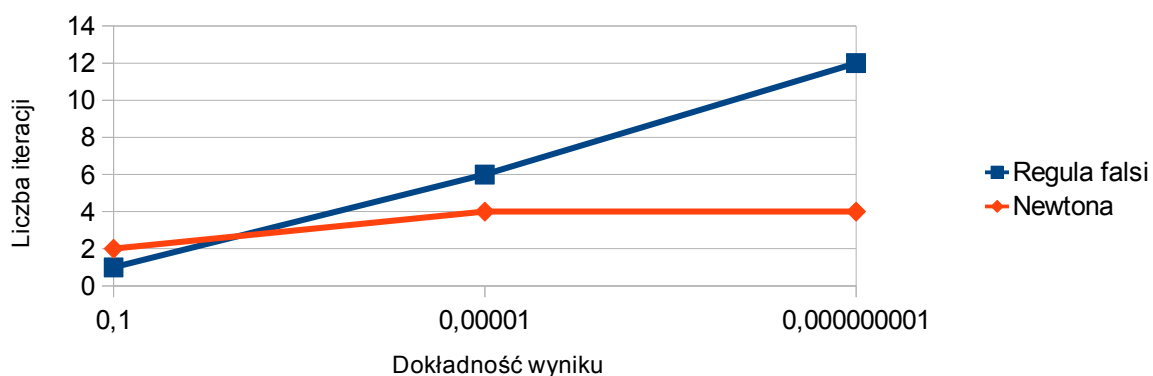
1. Porównanie wyników pierwszej próby dla obu metod



2. Porównanie wyników drugiej próby dla obu metod



3. Porównanie wyników trzeciej próby dla obu metod



W wykonanych testach bardzo dobre wyniki osiągnęła metoda Newtona, jest dla tej konfiguracji bardzo szybko bieżna.

8. Podsumowanie

Metoda przeszukiwania przedziału powinna być wykorzystywana tylko do lokalizacji przybliżenia pierwiastka rzeczywistego funkcji. Do uściślenia wartości pierwiastka należy używać metodę *Regula falsi* lub *Newtona*.

Najskuteczniejszą użytą w pracy metodą rozwiązywania równań nieliniowych jest *metoda Newtona*. Jest szybko zbieżna, we wszystkich wykonanych próbach wymagała mniejszej liczby iteracji niż metoda *Regula falsi*. Różnica w skuteczności obu metod jest bardzo wyraźna w dużych przedziałach i podczas szukania bardzo dokładnego wyniku.

Metodę obliczeń oraz dane wejściowe należy dobierać indywidualnie dla każdego przypadku.

Podczas wybierania metody uściślenia wyniku należy mieć na uwadze wymogi metody, które rygorystycznie określają warunki jej użycia.

Bibliografia:

I Opracowania:

1. K. Dems, *Podstawy metod numerycznych*, wykład czwarty,
2. B. Eckel, *Thinking In Java*, Wydawnictwo Helion 2003,
3. Z. Fortuna, B. Macukow, J. Wąsowski, *Metody numeryczne*, Warszawa 1998,
4. *Obliczenia naukowe: wybrane problemy*, red. S. Grzegórski, M. Miłosz, P. Muryjas, Lublin 2003,
5. E. Kącki, A. Małolepszy, A. Romanowicz, *Metody numeryczne dla inżynierów*, Łódź 1997,

II Strony internetowe:

1. The Java Tutorials
- <http://download.oracle.com/javase/tutorial/>
2. Strona internetowa projektu NetBeans
- www.netbeans.org
3. Strona internetowa Singulars System – JEP i DJEP
- www.singularsys.com
4. SingSurf - Interactive Geometry - Richard Morris
- www.singsurf.org
5. Function Viewer 1.0
- <http://warp.povusers.org/FuncViewer/>