

Projekt z przedmiotu

Sztuczna Inteligencja

**Zrealizować sieć neuronową LVQ uczącą się rozpoznawania kwiatów irysa**

**Imię i nazwisko:** Rafał Nazarko

**Grupa projektowa:** P5

**Numer indeksu:** 163982

Rzeszów 2021

**Spis treści**

[1. Opis projektu 2](#_Toc72539474)

[2. Zbiór danych 2](#_Toc72539475)

[2.1 Przedstawienie zbioru 2](#_Toc72539476)

[2.2 Przygotowanie i normalizacja danych 2](#_Toc72539477)

[3. Wprowadzenie teoretyczne 3](#_Toc72539478)

[3.1 Sieć neuronowa 3](#_Toc72539479)

[3.1.1 Model sztucznego neuronu 3](#_Toc72539480)

[3.1.2 Sieć jednowarstwowa 3](#_Toc72539481)

[3.1.3 Sieć wielowarstwowa 4](#_Toc72539482)

[3.2 Algorytm LVQ 4](#_Toc72539483)

[3.2.1 LVQ1 4](#_Toc72539484)

[3.2.2 LVQ2 4](#_Toc72539485)

[3.2.3 LVQ2.1 4](#_Toc72539486)

[3.2.4 LVQ3 4](#_Toc72539487)

[3.2.5 Warstwa Kohonena 4](#_Toc72539488)

[3.4 Walidacja krzyżowa 4](#_Toc72539489)

[4. Kod programu 4](#_Toc72539490)

[5. Eksperymenty 4](#_Toc72539491)

[6. Wnioski 4](#_Toc72539492)

[7. Bibliografia 5](#_Toc72539493)

# Opis projektu

Celem projektu jest stworzenie sieci neuronowej LVQ uczącej się rozpoznawania kwiatów irysa. Zakres realizacji obejmował przygotowanie danych, stworzenie algorytmu w języku Python oraz przeprowadzenie niezbędnych eksperymentów, w ramach których poszukiwane były najbardziej optymalne konfiguracje parametrów sieci a także najkrótszy czas wykonania.

# Zbiór danych

## Przedstawienie zbioru

Baza danych Iris Data Set [1] wykorzystana w tym projekcie pochodzi z repozytorium uczenia maszynowego UC Irvine. Jest to prawdopodobnie najbardziej znana baza danych, jaką można znaleźć w literaturze dotyczącej rozpoznawania wzorców. Zawiera 3 klasy po 50 instancji. Nie posiada brakujących/nieokreślonych wartości dlatego nie jest konieczne usuwanie rekordów. Każda z instancji składa się z 4 atrybutów w formacie liczb rzeczywistych oraz atrybutu klasy, która odnosi się do typu rośliny.

**Informacje o atrybutach:**

1. **Długość kielicha** (w centymetrach)
2. **Szerokość kielicha** (w centymetrach)
3. **Długość płatka** (w centymetrach)
4. **Szerokość płatka** (w centymetrach)
5. **Klasa:**
   * Iris Setosa
   * Iris Versicolour
   * Iris Virginica

## Przygotowanie i normalizacja danych

Klasy należało przekształcić z literałów na wartości liczbowe aby ujednolicić dane oraz sprawniej przeprowadzać operacje porównywania. Literały klas wymienione w poprzednim podpunkcie przekształcono odpowiednio w liczby naturalne z przedziału od 1 do 3. Separator atrybutów zamieniono z przecinka (‘,’) na średnik (‘;’).

Kolejnym krokiem była normalizacja danych w celu umożliwienia ich wzajemnego porównywania i dalszej analizy. Wszystkie atrybuty poza klasami zostały przeliczone na odpowiadające im wartości w przedziale od -1 do 1.

Poniżej zostały zamieszczone dane oryginalne oraz odpowiadające im dane po przygotowaniu i normalizacji, aby zobrazować zaistniałe zmiany.

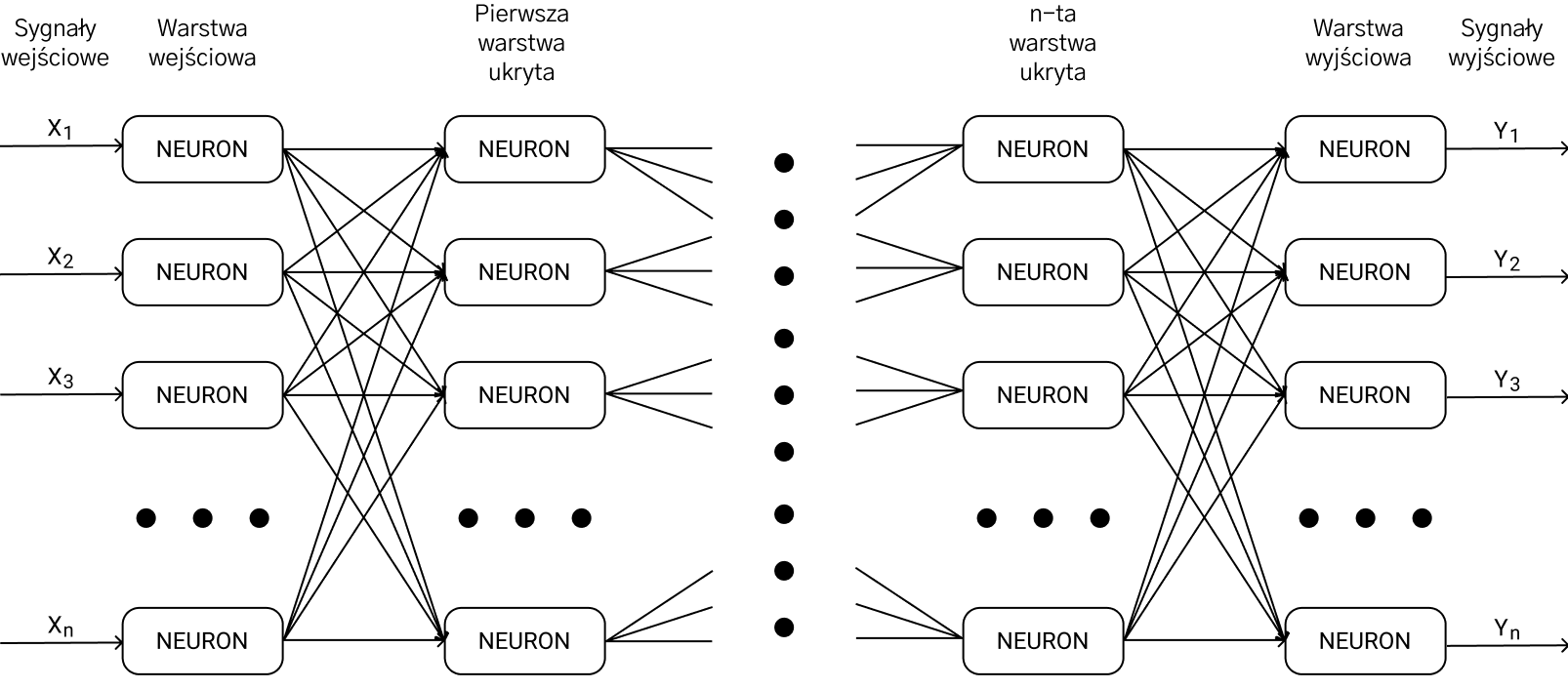
|  |
| --- |
| **Dane oryginalne** |
| 5.0,3.3,1.4,0.2,Iris-setosa  7.0,3.2,4.7,1.4,Iris-versicolor |

|  |
| --- |
| **Dane po przygotowaniu i normalizacji** |
| -0.611111;0.083333;-0.864407;-0.916667;1  0.5;0.0;0.254237;0.083333;2 |

# Wprowadzenie teoretyczne

## Sieć neuronowa

Termin sztuczne sieci neuronowe (SSN) lub również często określane jako po prostu sieci neuronowe (NN), obejmują rodzinę nieliniowych metod obliczeniowych, które przynajmniej we wczesnej fazie ich rozwoju były inspirowane funkcjonowaniem ludzkiego mózgu. Są to bardzo wyrafinowane narzędzia obliczeniowe zdolne do modelowania niezwykle złożonych funkcji. Składa się ona z dużej liczby elementów przetwarzających informacje. Elementy te, zwane *neuronami*, powiązane są ze sobą w sieć wzajemnie zależnych od siebie współczynników. Połączenia oraz ich parametry są podstawą działania sieci a jej topologia może różnić się w zależności od obranego algorytmu. Strukturę taką dzieli się na warstwy: *wejściową, wyjściową* oraz warstwy *ukryte*. W każdej sieci obligatoryjnie znajduje się warstwa wejściowa – przyjmująca zestaw danych, który ma zostać poddany analizie, oraz wyjściową – zwracającą wyniki działania algorytmu. Warstw ukrytych może istnieć dowolna ilość a decyduje o tym model wybranej sieci. W większości współczesnych zastosowań realizuje się sieci o modelu wielowarstwowym. *Sygnały* przekazywane są pomiędzy neuronami poszczególnych warstw aż do najwyższej warstwy dającej odpowiedź, a ich wpływ na odpowiedź neuronu jest zależna od *wag*, które są odpowiednio dopasowywane w procesie *uczenia sieci.*

**

*Rysunek 1. Przykładowy model sieci neuronowej.*

*Źródło: opracowanie własne*

### Model sztucznego neuronu

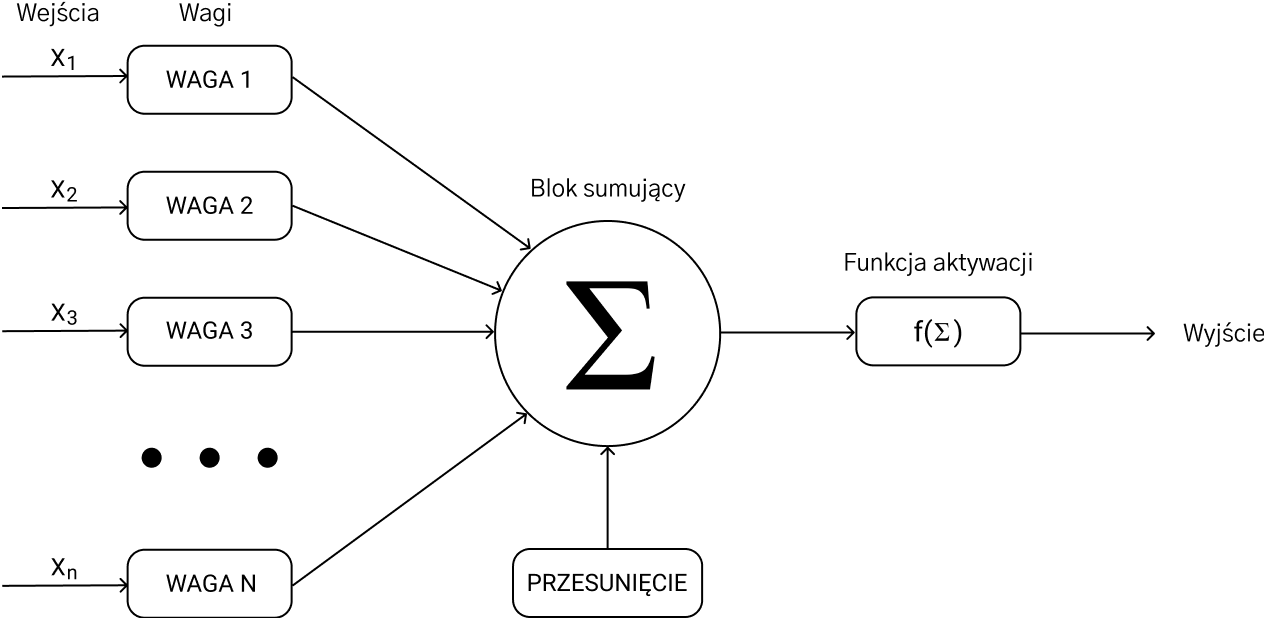
Mianem sztucznego neuronu określa się system przetwarzający sygnały przekazane do jego wejścia, na pojedynczą wartość wyjściową. Model sztucznego neuronu zwany *perceptronem* został stworzony na podstawie biologicznego neuronu.

Może on przyjmować dowolną ilość sygnałów wejściowych, najczęściej oznaczanych jako . Sygnały z wejść mnożone są z odpowiednimi wagami , a następnie sumowane wraz z przesunięciem w bloku sumującym [5]. Obliczona wartość poddawana jest działaniu funkcji aktywacji, która ostatecznie zwraca sygnał wyjściowy zwany *pobudzeniem neuronu*. Ogólny wzór na pobudzenie neuronu przedstawia poniższa zależność:

(3.1)

gdzie:

* – wartość i-tego sygnału wejściowego
* – waga i-tego sygnału wejściowego
* – ilość sygnałów wejściowych
* – przesunięcie
* – pobudzenie neuronu

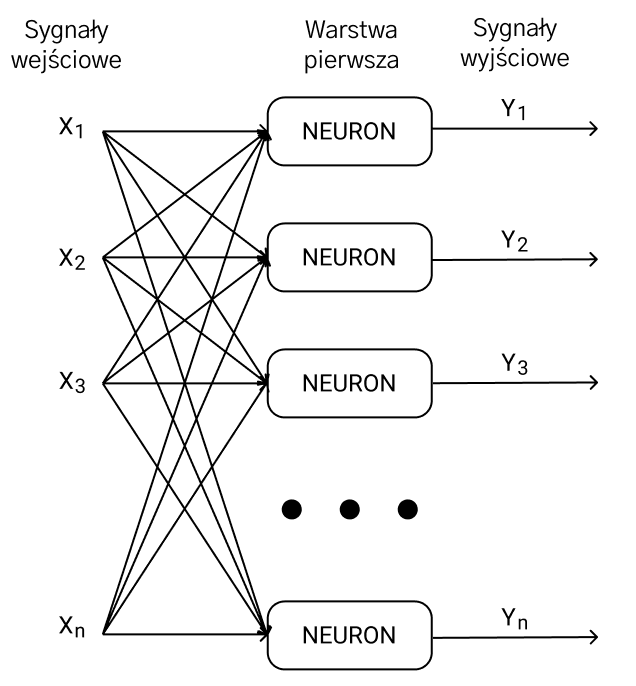


*Rysunek 2. Model sztucznego neuronu.*

*Źródło: opracowanie własne*

### Sieć jednowarstwowa

Sieć jednowarstwowa ma bardzo prymitywną budowę. Składa się jedynie z jednej warstwy neuronów oraz sygnałów wejściowych i wyjściowych. Nie posiada żadnych neuronów w warstwach ukrytych. Ma niewielkie znaczenie praktyczne, lecz nadal znajduje zastosowanie wszędzie tam, gdzie występuje niska złożoność problemu a istnienie jednej warstwy jest wystarczające do rozwiązania określonego zagadnienia. Neurony ułożone w pojedynczej warstwie działają niezależnie od siebie, stąd możliwości takiej sieci są ograniczone do możliwości pojedynczych neuronów [2]. Każdy z nich realizuje odwzorowanie funkcyjne pobudzenia neuronu (3.1). Przykładowa struktura sieci jednowarstwowej została zobrazowana poniżej.



*Rysunek 3. Model sieci jednowarstwowej.*

*Źródło: opracowanie własne*

### Sieć wielowarstwowa

Sieci wielowarstwowe znajdują zastosowanie w większości współczesnych realizacji sieci neuronowych. Tworzone są przez neurony ułożone w wielu warstwach [2]. Są rozszerzeniem sieci jednowarstwowej, jednak oprócz warstwy wejściowej i wyjściowej istnieje pomiędzy nimi co najmniej jedna warstwa ukryta. Sygnały wejściowe podawane są na wejścia neuronów warstwy wejściowej. Przetworzone dane są przekazywane na wyjścia neuronów, skąd trafiają na wejścia neuronów kolejnej warstwy. Proces ten jest powtarzany tyle razy ile jest warstw ukrytych. Na koniec, wyjścia ostatniej warstwy ukrytej są sygnałami wejściowymi ostatniej warstwy – warstwy wyjściowej. Działanie poszczególnych warstw można przedstawić według poniższej zależności:

(3.2)

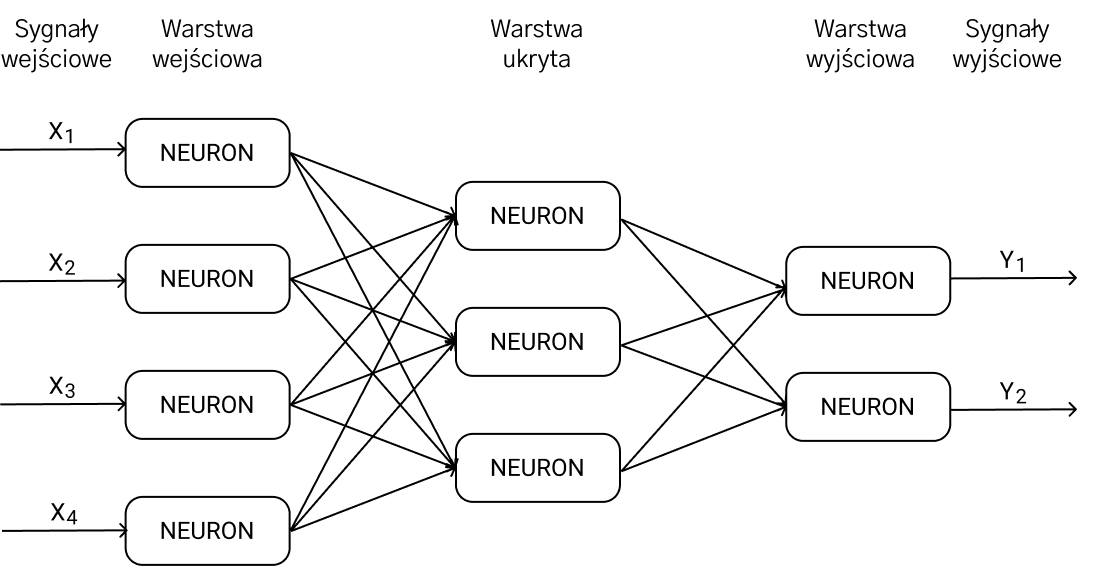
gdzie:

* – macierz wag dla n-tej warstwy
* – macierz wektorów wejściowych
* – wektor przesunięć dla n-tej warstwy
* – funkcja aktywacji dla n-tej warstwy
* – pobudzenie neuronu na n-tej warstwie

Dla zobrazowania działania sieci wielowarstwowych, poniżej został przedstawiony opis działania sieci trójwarstwowej [3]:

(3.3)

Do czego przydatne są warstwy ukryte? Powodem jest to, że jeśli nie ma ukrytych warstw to odwzorowywane dane wejściowe, nie są ze sobą w żaden sposób powiązane na wyjściach poszczególnych neuronów. W rzeczywistych problemach zmienne wejściowe są zwykle wysoce współzależne i wpływają na wynik w skorelowany sposób. Neurony warstwy ukrytej pozwalają uchwycić subtelne interakcje pomiędzy danymi wejściowymi, które ostatecznie wpływają wyjście. Ponadto, ukryte warstwy reprezentują cechy lub atrybuty wyższego poziomu naszych danych. Każdy z neuronów w warstwie ukrytej waży dane wejściowe inaczej, ucząc się innej pośredniczącej charakterystyki danych, a neuron wyjściowy jest wtedy ich funkcją zamiast surowych danych wejściowych. Włączając więcej niż jedną ukrytą warstwę, sieć dostaje możliwość nauczenia się wielu poziomów abstrakcji oryginalnych danych wejściowych przed uzyskaniem ostatecznego wyniku. Dzięki wykorzystaniu wielu warstw, sieć potrafi rozwiązywać problemy, z którymi nie radzi sobie sieć jednowarstwowa. Funkcje aktywacji zwiększają jej zdolność do wychwytywania nieliniowych relacji między wejściami a wyjściami. Łącząc ze sobą wiele nieliniowych transformacji za pomocą warstw, radykalnie zwiększa to elastyczność i zastosowanie sieci neuronowych [4].



*Rysunek 4. Przykładowa struktura sieci trójwarstwowej.*

*Źródło: opracowanie własne*

### Sieć jednokierunkowa

Sieć jednokierunkowa jest jednym ze sposobów działanie sieci neuronowych. Taką architekturę posiadają najprostsze sieci neuronowe. Informacje są podawane tylko w jednym kierunku – każdy neuron reaguje na jeden zestaw danych wejściowych tylko raz. Są częściej stosowane ze względu na prostotę w analizie. Sieci takie nazywane są czasami *heteroasocjacyjnymi* i z założenia są rozpatrywane jako twory statyczne, ponieważ ewentualne procesy przejściowe, zachodzące w sieci podczas jej pracy, nie mają znaczenia z punktu widzenia celu funkcjonowania sieci i mogą być pomijane[5].

### Sieć rekurencyjna

Bardziej złożone sieci są rekurencyjne, co znaczy, że dane wychodzące z ukrytych warstw są podawane pośrednio lub bezpośrednio do jednej z poprzednich warstw. Sprzężenie zwrotne powoduje, że sieć po jakimś czasie osiąga stan równowagi i „decyduje się” na konkretną odpowiedź. Takie sieci określane bywają jako *autoasocjacyjne* i odznaczają się bogatymi własnościami dynamicznymi, ponieważ sygnały mogą krążyć w sieci dowolnie długo, tym samym powodując powstanie różnorodnych i ciekawych przebiegów sieci przejściowych [5].

### Uczenie sieci neuronowej

Uczenie sieci to proces dobierania odpowiedniego zestawu wag dla danego zadania. Uczenie sieci można podzielić na uczenie nadzorowane oraz uczenie bez nauczyciela. Metodę uczenia pod nadzorem stosuje się gdy problem do rozwiązania jest zrozumiały dla człowieka oraz spodziewana jest określona odpowiedź od sieci. Wiedząc jakie sygnały są poprawne na wyjściu sieci dla każdego sygnału możemy je porównać z aktualną odpowiedzią sieci, po czym w razie potrzeby zmodyfikować wagi neuronów. Dobór odpowiednich wag odpowiada procesowi minimalizacji funkcji błędu, którą można zapisać jako:

(3.4)

gdzie:

* – odpowiedź n-tego neuronu
* – poprawny sygnał dla n-tego neuronu
* – ilość neuronów
* – błąd dopasowania

W zależności od wybranego algorytmu, funkcja skorygowania wag sygnałów może wyglądać w inny sposób. Jednakże w większości przypadków funkcja ta będzie przybierać postać zbliżoną do poniższej:

(3.5)

gdzie:

* – nowy wektor wag
* – aktualny wektor wag
* – współczynnik uczenia
* – błąd dopasowania
* – wektor wejściowy

W sytuacjach gdu problem jest zbyt skomplikowany albo określone przez ludzi cechy prowadzą do niesatysfakcjonującego rozwiązania, można używać sposobu nauczania bez nauczyciela. Sieć ucząca się w taki sposób musi być odpowiednio złożona, wielowarstwowa, żeby mogła wykrywać w danych wejściowych różnorodne zależności. Ponieważ sieci uczone bez nadzoru nie dążą do zadanego wyniku, mają potencjał wykrywania prawidłowości i zależności, które wcześniej nie były znane. To powoduje, że są użyteczne do analizy zjawisk, których model jest niedokładny albo niekompletny – a to można powiedzieć prawie o wszystkich zjawiskach społecznych albo finansowych. Jednak stopień złożoności architektury takiej sieci jest wysoko zaawansowany a również nie pokrywa się z tematem projektu, toteż nie zostanie bardziej szczegółowo opisany w tym raporcie.

## Algorytm LVQ

### LVQ1

### LVQ2

### LVQ2.1

### LVQ3

### Warstwa Kohonena

* 1. **Działanie sieci nauczonej**

## Walidacja krzyżowa

# Kod programu

# Eksperymenty

* 1. **Eksperyment 1**

Znalezienie kresu górnego i dolnego

* 1. **Eksperyment 2**

Znalezienie optymalnych parametrów Lr oraz S1

* 1. **Eksperyment 3**

Znalezienie odpowiedniej ilości epok i przedziałów

# Wnioski

Obliczenia przeprowadzone „na kartce” pokrywają się z tymi obliczonymi w programie Octave. Wyniki dla poszczególnych metod plasują się następująco:

* Metoda bisekcji:
* Metoda regula falsi:
* Metoda siecznych:

Można więc stwierdzić, że największą dokładnością wykazuje się metodą siecznych. Cechuje się najmniejszą ilością kroków do osiągnięcia zbieżności, a co za tym idzie, również najmniejszą złożonością czasową. Jednakowoż metoda bisekcji jest najprostszą w obliczaniu metodą rozwiązywania równań nieliniowych.

# Bibliografia