Uniwersytet Warmińsko-Mazurski w Olsztynie
Wydział Matematyki i Informatyki

Kierunek: **Informatyka**

Kamil Amarasekara

Aplikacja wspomagająca naukę algebry liniowej

Praca inżynierska wykonana
w Katedrze Informatyki Stosowanej i Modelowania Matematycznego
pod kierunkiem
dr hab. Viktorii Onyshchenko

Olsztyn, 2026 rok

University of Warmia and Mazury in Olsztyn
Faculty of Mathematics and Computer Science

Field of Study: **Computer Science**

Kamil Amarasekara

An application supporting the learning of linear algebra

Engineering Thesis is performed
in the Department of Applied Informatics and Mathematical Modeling
under the supervision of
dr hab. Viktoria Onyshchenko

Olsztyn, 2026

Spis treści

Streszczenie

Celem tej pracy było stworzenie oraz wdrożenie aplikacji webowej, która wspomaga naukę algebry liniowej w zakresie rozwiązywania układów równań liniowych metodą eliminacji Gaussa. Głównym priorytetem było podejście edukacyjne: aplikacja ma za zadanie nie tylko podawać odpowiedź, ale również pokazać szczegółowy proces przekształcania macierzy rozszerzonej [A|b] w formie kolejnych kroków. Każdy krok zawiera opis przeprowadzonej operacji na wierszach, a interfejs użytkownika sprawia, że interpretacja jest prostsza dzięki wyróżnianiu elementu wiodącego (pivotu) oraz oznaczaniu wierszy i kolumn, które są modyfikowane. Ustalono także zasady dotyczące wprowadzania danych: puste komórki traktowane są jako zero, zezwolono na zapisy dziesiętne z kropką lub przecinkiem oraz na zapisy ułamków w formie a/b.

Wykorzystane metody składały się z projektowania architektury klient-serwer oraz realizacji obliczeń w arytmetyce wymiernej przy użyciu typu Fraction, co umożliwia uzyskanie precyzyjnych wyników i czytelnych wartości pośrednich bez błędów zaokrągleń. Część serwerową stworzono w języku Python z zastosowaniem frameworka Flask. Udostępniono endpoint API, który zwraca etapy obliczeń w formacie JSON. Warstwa prezentacji została opracowana przy użyciu technologii HTML, CSS oraz JavaScript, a renderowanie notacji matematycznej zapewnia biblioteka MathJax.

Najważniejszym efektem pracy jest funkcjonalna aplikacja, która pozwala na rozwiązywanie układów z jednym rozwiązaniem, brakiem rozwiązań czy z nieskończoną liczbą rozwiązań, a także dostarcza czytelną prezentację kroków. Prawidłowość działania potwierdzono szeregiem testów jednostkowych (35 testów zakończonym powodzeniem) oraz wysokim pokryciem testami kluczowych modułów. Wnioskiem jest, że wizualizacja krokowej eliminacji Gaussa w aplikacji webowej stanowi efektywne wsparcie w procesie uczenia się i ułatwia zrozumienie wpływu operacji wierszowych na postać macierzy.

Abstract

The aim of this work was to create and implement a web application that supports the learning of linear algebra in the field of solving systems of linear equations using the Gauss elimination method. The main priority was the educational approach: the application is designed not only to provide the answer, but also to show the detailed proces of transforming the augmented matrix [A|b] in the form of successive steps. Each step contains a description of the operation performer on the rows, and the user interface makes interpretation easier by highlighting the pivot element and marking the rows and columns that are modified. Rules for data entry were also established: empty cells are treated as zero, decimal notation with a dot or comma is allowed, and fractions can be written in the form a/b.

The methods used consisted of designing a client-server architecture and performing calculations in rational arithmetic using the Fraction type, which allows for precise results and clear intermediate values without rounding erros. The server part was created in Python using the Flask framework. An API endpoint was made aviable, which returns the calculation steps in JSON format. The presentation layer was developed using HTML, CSS and JavaScript technologies, and the MathJax library provides rendering of mathematical notation.

The most important result of the work is a functional application that allows solving systems with a single solution, no solutions, or an infinite numer of solutions, and also provides a clear presentation of the steps. The correctness of the application has been confirmed by a series of unit tests (35 tests completed successfully) and high coverage of key modules. The conclusion is that the visualisation of Gaussian elimination in a web application provides effective support in the learning proces and facilitates understanding of the impact of row operations on the form of a matrix.

Wstęp

Algebra liniowa to jeden z fundamentalnych obszarów matematyki, który znajduje zastosowanie w informatyce i naukach inżynieryjnych. Terminy takie jak macierze, wektory i układy równań liniowych są kluczowe dla wielu praktycznych tematów, w tym analizy danych, grafiki komputerowej, metod numerycznych oraz algorytmów uczenia maszynowego. W edukacji akademickiej szczególną rolę odgrywa metoda eliminacji Gaussa, która pozwala na przekształcanie układów równań do postaci schodkowej i interpretację rodzaju rozwiązań.

Celem niniejszej pracy jest przedstawienie tworzenia oraz realizacji aplikacji webowej, która wspiera naukę eliminacji Gaussa. Aplikacja ma na celu pomóc użytkownikowi nie tylko w osiągnięciu wyniku, ale przede wszystkim w pojęciu kolejnych etapów przekształcania macierzy poprzez prezentację operacji na wierszach w formie krokowej, wzbogaconej o opisy oraz elementy graficzne.

Praca podzielona jest na pięć części. W pierwszym rozdziale zaprezentowano konkurencyjne narzędzia edukacyjne i obliczeniowe, co uzasadnia potrzebę stworzenia specjalistycznego rozwiązania. Drugi rozdział omawia cele projektu oraz określa wymagania funkcjonalne, jak i niefunkcjonalne. W trzecim rozdziale przedstawiono projekt oraz architekturę aplikacji, obejmując podział na różne moduły oraz przebieg danych między warstwą obliczeniową, a interfejsem użytkownika. Czwarty rozdział skupia się na głównych aspektach implementacji – w tym algorytmach obliczeniowych oraz realizacji warstwy prezentacyjnej. Ostatnia część zawiera informacje na temat testowania oraz potwierdzenia działania aplikacji na przykładach edukacyjnych.

1. Przegląd konkurencyjnych rozwiązań

1.1. Motywacja i kontekst pracy

Algebra liniowa jest podstawą wielu zagadnień stosowanych w informatyce i różnych dziedzinach technicznych. Technologie związane z uczeniem maszynowym, grafiką komputerową, analizą danych czy metodami numerycznymi w dużej mierze bazują na operacjach na macierzach i wektorach oraz na rozwiązywaniu układów równań liniowych. Dlatego umiejętność efektywnego korzystania z narzędzi algebry liniowej jest istotna na początkowych etapach nauki informatyki.

Jedną z kluczowych technik do rozwiązywania układów równań jest metoda eliminacji Gaussa. W kontekście akademickim często przedstawiana jest w sposób statyczny, co może powodować trudności w zrozumieniu dynamiki przekształceń przez niektórych studentów: wybór elementy wiodącego, normalizacja wiersza, eliminacja kolejnych współczynników oraz interpretacja otrzymanej postaci schodkowej. Problemy pojawiają się, zwłaszcza gdy w obliczeniach występują ułamki lub gdy trzeba zidentyfikować sytuacje, kiedy jest brak rozwiązania, bądź gdy jest ich nieskończenie wiele.

Równocześnie narzędzia informatyczne zyskują na znaczeniu w wspieraniu edukacji. Aplikacje edukacyjne mogą przeprowadzać użytkownika przez rozwiązanie krok po kroku, co

redukuje obciążenie poznawcze i ułatwia zrozumienie sensu operacji na wierszach. W kontekście algebry liniowej szczególnie użyteczne są rozwiązania, które nie tylko dostarczają wyniku, ale również w przejrzysty sposób ukazują proces przekształcania macierzy, zgodnie z metodologią nauczania na uczelniach.

1.2. Cel pracy

Celem tego projektu jest stworzenie aplikacji webowej, która pomoże w nauce algebry liniowej, szczególnie w zakresie eliminacji Gaussa. Aplikacja ma umożliwiać użytkownikom wprowadzanie danych, przeprowadzanie obliczeń oraz śledzenie poszczególnych kroków przekształcania macierzy rozszerzonej, wraz z opisem wykonywanych operacji. Szczególny nacisk położono na dydaktyczne przedstawienie algorytmu, aby użytkownik mógł nie tylko zobaczyć, co zostało zrobione, ale także zrozumieć, dlaczego dany etap jest właściwy i do czego prowadzi.

W celu realizacji głównego zamysłu określono cele szczegółowe:

- stworzenie algorytmu eliminacji Gaussa, który pozwala na generowanie zrozumiałych kroków (operacji wierszowych) prowadzących do rozwiązania,
- opracowanie mechanizmów wizualizacji zmian w macierzy, takie jak wskazywanie elementu wiodącego oraz wyróżnianie wierszy i kolumn istotnych w danym kroku,
- zapewnienie intuicyjnego interfejsu do wprowadzania macierzy oraz przeglądania rozwiązania,
- zintegrowanie warstwy obliczeniowej z prezentacją w strukturze aplikacji webowej
- sprawdzenie poprawności działania na zestawie edukacyjnych przypadków, obejmujących różne rodzaje układów.

Oczekiwanym wynikiem jest narzędzie, które pozwoli studentom samodzielnie się uczyć oraz być użyteczną pomocą dydaktyczną na zajęciach.

1.3. Zakres pracy i przyjęte założenia

Zakres pracy dotyczy stworzenia, wdrożenia oraz wstępnej analizy edukacyjnej aplikacji, której celem jest rozwiązywanie układów równań liniowych przy użyciu metody eliminacji Gaussa. Analizowane są układy, które można przedstawić w formie macierzy rozszerzonej oraz przekształcać za pomocą elementarnych operacji wierszowych.

Aplikacja obejmuje szczególnie:

- rozwiązywanie układów równań liniowych poprzez transformację macierzy rozszerzonej do postaci schodkowej (a w wybranym zakresie również do zredukowanej postaci schodkowej),
- rozpoznanie sytuacji: jedno rozwiązanie, brak rozwiązań oraz nieskończenie wiele rozwiązań,
- przedstawienie kroków obliczeń z towarzyszącym opisem operacji wierszowych oraz wizualnymi wskazówkami.

Jednocześnie, ze względu na charakter pracy i ograniczenia, poza zakresem pozostają:

- tematy dotyczące wartości i wektorów własnych,
- zaawansowane metody numeryczne dla bardzo dużych macierzy oraz dokładnej analizy wydajności obliczeniowej,
- rozbudowane funkcje edukacyjne takie jak rozpoznawanie błędów użytkownika, dynamiczny dobór zadań czy dostosowywanie ścieżki nauki.

Z technicznego punktu widzenia zastosowano architekturę aplikacji webowej. Warstwa serwerowa została stworzona w języku Python przy wykorzystaniu frameworka Flask, a interfejs użytkownika bazuje na HTML, CSS i JavaScript. Komunikacja między warstwami odbywa się za pośrednictwem interfejsu REST, a dane przesyłane są w formacie JSON. Do wyświetlania matematycznych zapisów w przeglądarce użyto biblioteki obsługującej notację LaTeX. Aplikacja ma być uruchamiana lokalnie lub na serwerze, z dostępem przez standardową przeglądarkę.

1.4. Analiza konkurencyjnych narzędzi

Aby umiejscowić projekt w kontekście już istniejących rozwiązań, poddano analizie trzy znane narzędzia stosowane w edukacji matematycznej: WolframAlpha, GeoGebra, Desmos. Wybór tych narzędzi wynika z ich powszechnej dostępności, łatwego dostępu przez przeglądarki oraz faktu, że często są wykorzystywane przez studentów do wspomagania obliczeń i wizualizacji. Analiza skupia się na tym, w jakim stopniu każde z narzędzi wspiera naukę eliminacji Gaussa, zwłaszcza w kontekście prezentacji poszczególnych kroków oraz wyjaśniania działań na wierszach.

1.4.1. WolframAlpha

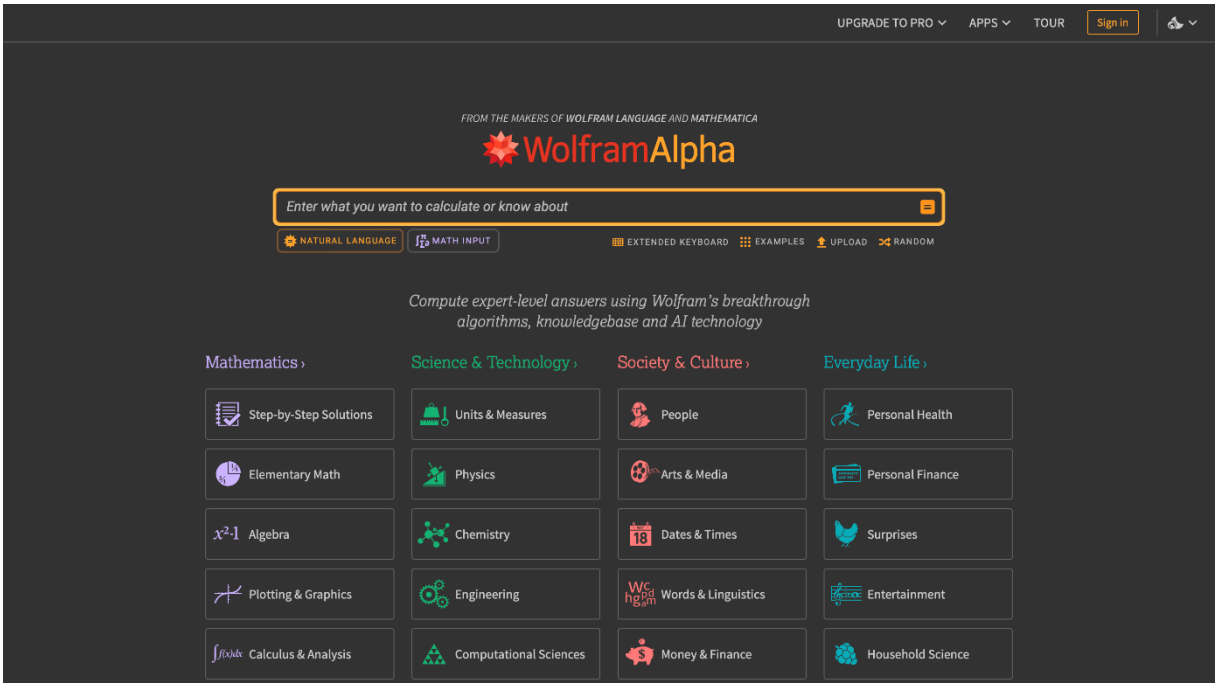
WolframAlpha jest narzędziem obliczeniowym “computational engine”, który dostarcza odpowiedzi na zapytania użytkowników oraz korzysta z wbudowanych algorytmów. W praktyce działa jako wszechstronny kalkulator dla wielu dziedzin, takich jak matematyka, fizyka, statystyka i inżynieria, co pozwala na szeroki zakres analizowanych problemów. Z perspektywy algebry liniowej narzędzie umożliwia m.in. rozwiązywanie układów równań, przeprowadzanie operacji na wierszach (takich jak wyznacznik, rząd, macierz odwrotna i różne rozkłady) oraz redukcję wierszową do formy schodkowej i zredukowanej.

Typowe korzystanie z WolframAlpha polega na wpisaniu zapytania w naturalnym języku lub matematycznej składni, po czym można przeglądać rezultaty obliczeń. W kontekście metody eliminacji Gaussa użytkownicy zazwyczaj dostają kocowy wynik (np. rozwiązanie układu), podczas gdy szczegółowe przedstawienie etapów transformacji bywa ograniczone. W wielu przypadkach pełny opis krok po kroku (“step-by-step solutions”) dostępny jest dopiero po wykupieniu subskrypcji lub w ramach trybu premium, co ma istotne znaczenie dla nauki: studenci korzystający z darmowej wersji często skupiają się głównie na rezultacie, a nie na procesie. [17]

Z perspektywy nauczania eliminacji Gaussa kluczowe jest nie tylko zaprezentowanie kolejnych macierzy pośrednich, ale również systematyczne wyjaśnienie, jakie operacje wierszowe zostały zastosowane oraz cele tych przekształceń (wybieranie elementu wiodącego, normalizacja wiersza, eliminowanie współczynników). WolframAlpha potrafi przedstawić operacje w formie notacji matematycznej, jednak sposób opisu jest uzależniony od ogólnego silnika obliczeniowego. Oznacza to, że użytkownik ma mały wpływ na szczegółowość komentarzy oraz na „tok dydaktyczny" dostosowany do konkretnego programu nauczania. Co więcej, narzędzie nie jest zaprojektowane jako aplikacja edukacyjna dla pojedynczego algorytmu: każde zadanie traktowane jest jako niezależne zapytanie, bez wbudowanej narracji czy ścieżki ćwiczeń skoncentrowanej tylko na eliminacji Gaussa.

W porównaniu do WolframAlpha, aplikacja stworzona w tej pracy ma węższy zakres tematyczny, ale koncentruje się na pełnej dostępności etapów obliczeń. Prezentacja rozwiązywania układów jest zaprojektowana jako integralny element procesu nauczania: użytkownik ma dostęp zarówno do wyników końcowych, jak i do kroków pośrednich, a te funkcjonalności są dostępne bez opłat w jednej, spójnej aplikacji edukacyjnej.

Przykładowy widok strony głównej serwisu WolframAlpha przedstawiono na rysunku 1.1. [17]



Rysunek 1.1.: Strona główna wolframalpha.com

1.4.2. GeoGebra

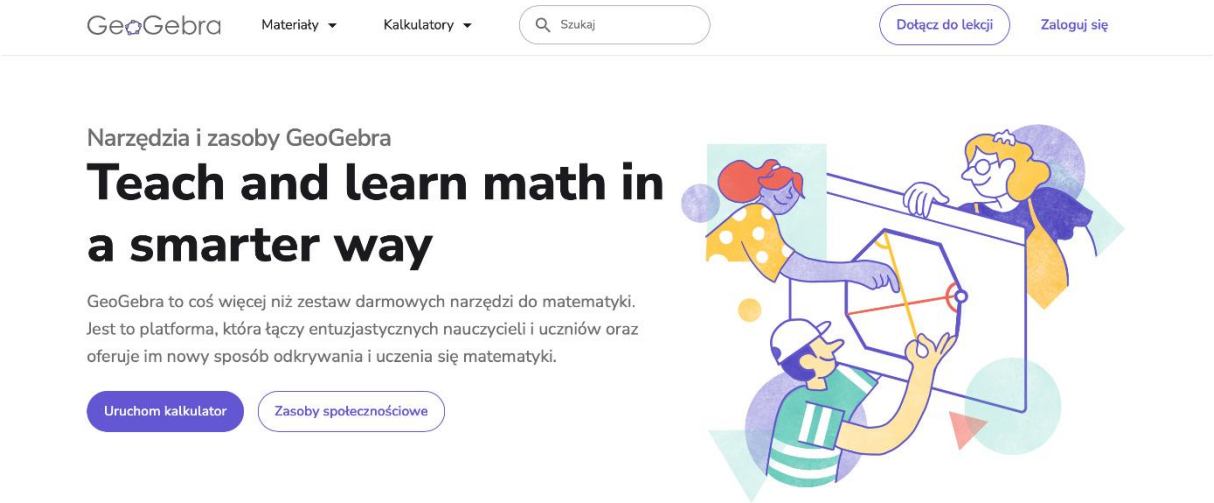
GeoGebra to edukacyjna platforma interaktywna, która łączy narzędzia z dziedziny geometrii, algebry, statystyki i analizy matematycznej. Oprócz typowych kalkulatorów oraz modułów, oferuje również rozbudowaną bazę materiałów dydaktycznych tworzonych przez społeczność składającą się z nauczycieli, uczniów i studentów. Z tego powodu GeoGebra jest często używana w edukacji jako środowisko do tworzenia i dzielenia się apletami oraz interaktywnymi ćwiczeniami.

W zakresie algebry liniowej GeoGebra dostarcza różne opcje do pracy z macierzami i układami równań, jednak w praktyce nie spełnia roli jednego, centralnego narzędzia „do eliminacji Gaussa”. Zamiast tego użytkownicy mają do dyspozycji różnorodne aplety lub zasoby edukacyjne, które mogą dotyczyć redukcji wierszowej, postaci schodkowej, interpretacji rzędu macierzy czy wizualizacji przekształceń liniowych. W zależności od konkretnego materiału można podejmować działania ręcznie (interaktywnie) lub obserwować poszczególne etapy redukcji. Jest to ważna cecha, ponieważ interaktywność może wspierać zrozumienie zasad operacji wierszowych – student nie tylko widzi końcowy wynik, ale może także wykonywać następne kroki.

Z perspektywy tej pracy, należy również zauważyć, że GeoGebra ma znaczące ograniczenia jako narzędzie do systematycznej nauki eliminacji Gaussa. Przede wszystkim, dostępne zasoby są rozproszone: różni autorzy stosują różną notację, odmienne zasady opisywania kroków oraz różny poziom szczegółowości w komentarzach. W rezultacie studenci mogą trafić na materiały wysokiej jakości, ale także na uproszczone aplety, które pokazują jedynie fragment procesu lub skupiają się jedynie na ostatecznym wyniku. Brakuje też jednolitej, wcześniej zaplanowanej ścieżki nauczania (np. zestawu ćwiczeń o wzrastającym poziomie trudności) skoncentrowanej wyłącznie na metodzie eliminacji Gaussa i omówieniu przypadków szczególnych (brak rozwiązań, nieskończona liczba rozwiązań). [18]

W porównaniu do tej metody, aplikacja opracowana w tej pracy oferuje spójne środowisko nauki: jednolity format przedstawiania kroków, konsekwentną notację dla operacji wierszowych oraz mechanizmy wizualnego wyróżniania kluczowych elementów (np. elementu wiodącego). Dzięki temu użytkownik zyskuje spójny proces, niezależnie od konkretnego przykładu, co sprzyja utrwaleniu umiejętności.

Przykładowy widok strony głównej serwisu GeoGebra przedstawiono na rysunku 1.2. [18]



Rysunek 1.2.: Strona główna geogebra.org

1.4.3. Desmos

Desmos to platforma edukacyjna, która jest najbardziej znana z kalkulatora graficznego oraz nowoczesnego, przejrzystego interfejsu. W kontekście edukacji najczęściej jest

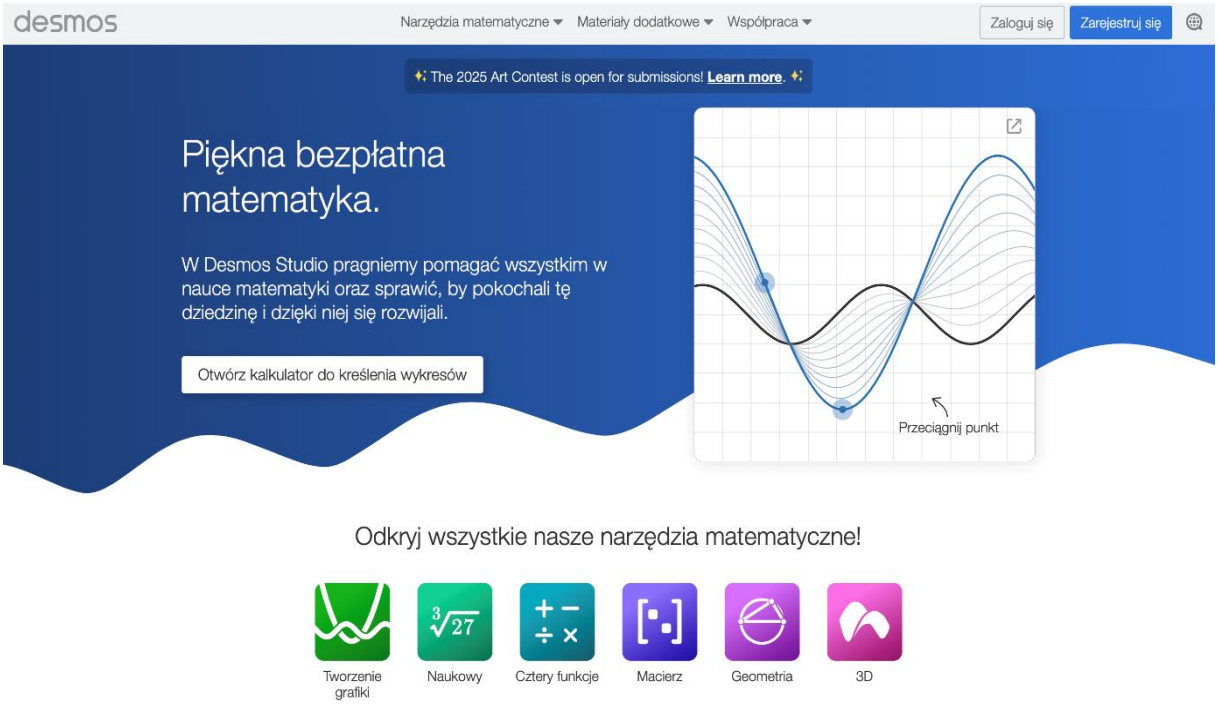
wykorzystywana do analizy równań i funkcji za pomocą wykresów, a także do tworzenia interaktywnych modeli matematycznych. Ważnym atutem Desmosa jest to, że jest przyjazny dla użytkownika; można szybko generować wizualizacje, zmieniać parametry, oraz obserwować, jak te zmiany wpływają na wykresy.

W zakresie algebry liniowej Desmos udostępnia podstawowe funkcje pracy z macierzami i wektorami, w tym przeprowadzanie określonych operacji na macierzach oraz tworzenie konstrukcji, które pomagają zrozumieć przekształcenia liniowe. Możliwość wizualizacji efektu działania macierzy na wektorach w przestrzeni 2D jest szczególnie cenna, ponieważ wspomaga intuicję geometryczną. Tego rodzaju podejście doskonale uzupełnia algebrę o wizualną interpretację, co bywa pomocne w procesie nauki.

Jednakże w Desmosie algorytm eliminacji Gaussa, rozumiany jako formalny sposób redukcji wierszowej, nie jest przedstawiany w sposób krok po kroku i dydaktyczny. Narzędzie nie jest zaprojektowane do "rozwiązywania równań metodą Gaussa w sposób krokowy", ale raczej jako platforma do obliczeń i wizualizacji. W rezultacie użytkownik zazwyczaj nie otrzymuje listy operacji wierszowych z wyjaśnieniem ani nie widzi kolejnych macierzy pośrednich. Brakuje także elementu interpretacyjnego, który jest typowy dla zajęć akademickich, czyli wskazania, w jakich sytuacjach forma schodkowa oznacza sprzeczność układu, a w jakich dowodzi istnienia nieskończonej liczby rozwiąż. [19]

Z punktu widzenia niniejszej pracy, Desmos można uznać za narzędzie wspierające wizualizację przekształceń, natomiast nie jako system, który prowadzi ucznia przez procedurę eliminacji Gaussa. Tworzona aplikacja ma na celu wypełnienie tej luki: oferuje kompletna sekwencję kroków redukcji wierszowej oraz końcowy wynik, a wszystko to jest dostępne za darmo i zaprojektowane z myślą o zastosowaniach dydaktycznych.

Przykładowy widok strony głównej serwisu Desmos przedstawiono na rysunku 1.3. [19]



Rysunek 1.3.: Strona główna desmos.com

1.5. Tabela porównawcza

Kryterium	WolframAlpha	GeoGebra	Desmos
Główna rola narzędzia	Uniwersalny silnik obliczeniowy do szerokiego zakresu problemów	Interaktywne narzędzia i materiały edukacyjne	Kalkulator graficzny i środowisko wizualizacji matematyki
Praca na macierzy rozszerzonej	Tak - obsługa układów równań i redukcji wierszowej	Możliwa, ale zależna od konkretnego narzędzia	Ograniczona; praca na macierzach i wektorach, bez nacisku na macierz rozszerzoną
Krokowa prezentacja metody eliminacji Gaussa	Zależna od trybu; rozwiązanie krokowe często w wersji płatnej	Zależna od apletu; brak jednolitego standardu prezentacji	Zwykle brak pełnej prezentacji krok po kroku
Objaśnienie operacji wierszowych	Częściowe; komentarz podporządkowany silnikowi obliczeniowemu	Zależna od autora materiału; jakość nierówna	Zazwyczaj brak objaśnień operacji wierszowych
Interpretacja postaci schodkowej	Możliwa, ale nie zawsze eksponowana dydaktycznie	Możliwa w wybranych materiałach	Ograniczona; brak prowadzenia w tym kierunku
Wbudowana ścieżka nauki / zestaw ćwiczeń	Brak spójnej ścieżki dla Gaussa; zadania jako niezależne zapytania	Dużo zasobów, ale rozproszonych i niespójnych	Brak ukierunkowanej ścieżki dla eliminacji Gaussa
Największa zaleta w tym kontekście	Szybkie uzyskanie wyniku i duże możliwości obliczeniowe	Interaktywność i możliwość adaptacji materiałów przez nauczyciela	Przejrzysty interfejs silna strona wizualizacji
Największe ograniczenie w tym kontekście	Kroki często ograniczone w wersji darmowej + mała kontrola nad formą wyjaśnień	Brak jednorodnej notacji i narracji w materiałach	Brak dydaktycznej prezentacji metody Gaussa

1.6. Wnioski

Przegląd narzędzi wskazuje, że znane platformy edukacyjne i obliczeniowe wspierają algebrę liniową, ale nie zawsze skutecznie odpowiadają na wymagania związane z nauczaniem metody eliminacji Gaussa w sposób jednolity i osiągalny dla wszystkich użytkowników. WolframAlpha ma bardzo szerokie możliwości obliczeniowe i potrafi prezentować rozwiązania w sposób etapowy, lecz w praktyce szczegółowe przedstawienie kroków może być ograniczone w wersji darmowej, a metody wyjaśniania są uzależnione od ogólnego silnika obliczeniowego. GeoGebra oferuje bogate interaktywne materiały, jednak ich podział oraz brak jednolitych standardów notacji i stopnia szczegółowości sprawiają, że trudno ją traktować jako spójne narzędzie do systematycznego uczenia się Gaussa. Desmos z kolei dobrze wspiera intuicyjne zrozumienie wizualne i interpretacje geometryczną, lecz nie proponuje prezentacji redukcji wierszowej w sposób krok po kroku, razem z wyjaśnieniem operacji i interpretacją przypadków szczególnych.

Wyniki przeprowadzonej analizy potwierdzają potrzebę stworzenia dedykowanego rozwiązania, które nie ma na celu konkurowania z rozbudowanymi platformami, lecz zapewnienia spójnej prezentacji jednego algorytmu w formie dydaktycznej. Aplikacja stworzona w ramach tej pracy skupia się na metodzie eliminacji Gaussa, a jej szczególną cechą jest pełna dostępność procesu rozwiązywania: użytkownik ma dostęp zarówno do ostatecznego wyniku, jak i do kroków pośrednich z wizualnym podkreśleniem kluczowych elementów (np. elementu wiodącego, wierszy i kolumn podlegających eliminacji). Ważne jest, że te funkcje są dostępne bez opłat, co umożliwia korzystanie z aplikacji jako narzędzia do samodzielnej nauki oraz wsparcia edukacyjnego w trakcie zajęć akademickich.

2. Cele i wymagania projektowanej aplikacji

2.1. Cel główny projektowanej aplikacji

Celem opracowanej aplikacji jest ułatwienie nauki metody eliminacji Gaussa poprzez zaoferowanie narzędzia, które potrafi rozwiązywać układy równań liniowych zapisane w postaci macierzy rozszerzonej [A|b] oraz przedstawić cały proces obliczeń w sposób edukacyjny. Aplikacja ma zapewnić użytkownikowi możliwość nie tylko uzyskania ostatecznego wyniku, ale także umożliwić prześledzenie poszczególnych operacji wierszowych, które prowadzą do uzyskania postaci schodkowej.

W przeciwieństwie do narzędzi uniwersalnych, które zazwyczaj skupiają się jedynie na rezultacie, opracowane rozwiązanie koncentruje się na jasnym prowadzeniu użytkownika przez algorytm: prezentowanie etapów pośrednich, opis wykonywanej czynności oraz wizualne wyróżnianie kluczowych elementów w danym czasie redukcji. Dodatkowo aplikacja jest dostępna bezpłatnie, a pełne przedstawienie kroków i wyników można zobaczyć bez konieczności wykupywania subskrypcji.

2.2. Cele szczegółowe

W ramach realizacji głównego celu wyznaczone zostały konkretne cele szczegółowe:

- **Implementacja eliminacji Gaussa z generowaniem kroków**

Algorytm nie tylko przekształca macierz do formy schodkowej, ale także tworzy zorganizowaną listę kroków pośrednich, którą użytkownik może zobaczyć.

- **Przejrzysta prezentacja operacji na wierszach**

Użytkownik powinien móc zobaczyć przy każdej operacji, co zostało zrobione (np. zamiana wierszy, normalizacja wierszy, eliminacja współczynnika) oraz z jakiego powodu.

- **Wizualne wsparcie do zrozumienia algorytmu**

Aplikacja powinna pokazywać element wiodący (pivot) oraz wskazywać odpowiednie wiersze i kolumny związane z aktualnym krokiem redukcji, by użytkownik mógł lepiej zrozumieć operację w kontekście zmian w macierzy.

- **Łatwe wprowadzanie danych i zarządzanie błędami**

Interfejs ma ułatwiać wprowadzanie współczynników macierzy, minimalizować ryzyko popełniania błędów oraz jasno informować o błędach w danych.

- **Spójna integracja warstwy obliczeniowej i interfejsu**

Warstwa serwera zajmuje się obliczeniami, natomiast warstwa klienta odpowiada za wyświetlanie i poruszanie się po krokach. Komunikacja powinna być jasna i powtarzalna.

- **Sprawdzenie poprawności**

Projekt zakłada stworzenie testów oraz weryfikację działania aplikacji na przykładach układów różnego typu (jedno rozwiązanie, brak rozwiązań, nieskończoność rozwiązań).

2.3. Użytkownicy i scenariusze użycia

Aplikacja jest dedykowana przede wszystkim studentom rozpoczynającym naukę na kierunkach technicznych, którzy uczą się metody eliminacji Gaussa. W praktyce, aplikacja znajdzie zastosowanie w dwóch głównych sytuacjach:

1. Nauka indywidualna – użytkownik wprowadza układ równań, uruchomia proces rozwiązania i analizuje każdy krok, korzystając z komentarzy oraz podświetleń.
2. Wsparcie dydaktycznie podczas zajęć – nauczyciel krok po kroku ilustruje zmiany w macierzy, zatrzymując się na kluczowych momentach (np. wybór pivotu, zerowanie elementów poniżej wiodącego, interpretacja postaci schodkowej).

2.4. Założenia projektowe i ograniczenia

Przyjęto następujące założenia projektowe:

- Aplikacja jest zaprojektowana do pracy z małymi układami, które są typowe dla ćwiczeń edukacyjnych; użytkownik ma możliwość wyboru rozmiaru w zakresie 2x2 do 9x9.
- Dane są wprowadzane w formie macierzy rozszerzonej, a rezultaty są prezentowane z pełnym opisem kroków.
- Wprowadzone liczby mogą być całkowite, dziesiętne lub w formie ułamków zwykłych.
- Matematyczne zapisy w opisach i działaniach są ukazane w formacie przypominającym ten używany w materiałach uniwersyteckich.

Ograniczenia związane z zakresem działania:

- Aplikacja skupia się na metodzie eliminacji Gaussa oraz uzyskiwaniu formy schodkowej; nie obejmuje innych działów algebry liniowej.
- Celem nie jest obsługa bardzo dużych macierzy ani optymalizacja obliczeń, lecz przejrzystość procesu oraz poprawność w zakresie nauczania.
- Nie zaplanowano rozbudowanych funkcji związanych z platformą e-learningową (zarządzanie kontami użytkowników, monitorowanie postępów, automatycznego przypisywania zadań).

2.5. Wymagania funkcjonalne

2.5.1. Wprowadzanie układu i walidacja danych

Aplikacja pozwala użytkownikowi na wybór rozmiaru układu, a następnie umożliwia wprowadzenie wartości do siatki, która odpowiada macierzy rozszerzonej [A|b]. Interfejs powinien umożliwiać szybkie wprowadzanie informacji, a jednocześnie minimalizować typowe błędy.

Wprowadzanie danych podlega następującym zasadom:

- **Pusta komórka traktowana jest jako 0.**

Oznacza to, że użytkownik nie musi wypełniać wszystkich pól, aby przeprowadzić obliczenia – brak danych nie uniemożliwia działania i jest interpretowany jako współczynnik równy zeru.

- **Dozwolone są liczby całkowite, liczby dziesiętne oraz ułamki**

Ułamki powinny być wprowadzone w formie a/b, gdzie a i b to liczby całkowite, a znak / służy jako separator.

- **Nie dopuszcza się wprowadzania liter oraz niewłaściwych znaków specjalnych.**

Interfejs nie akceptuje ciągów takich jak abc ani znaków, które nie pasują do formatu liczbowego.

- **Dozwolone znaki w polu danych to:**

„.”, „,”, „-”, „/”

gdzie:

  - . oraz , mogą być użyte jako separator do miejsc dziesiętnych,
  - - umożliwia wprowadzanie liczb ujemnych,
  - / pozwala na zapisywanie ułamków.
- **Użytkownik otrzymuje jasny komunikat o błędzie**, gdy wpisze niedozwolony znak (np. litery) lub niepoprawny format liczby.

Aplikacja oferuje również funkcje, które ułatwiają korzystanie z formularza, takie jak usunięcie wprowadzonych danych i rozpoczęcie pracy z nowym rozmiarem układu.

2.5.2. Uruchomienie obliczeń i generowanie kroków

Po wprowadzeniu wszystkich danych przez użytkownika, system zaczyna przeprowadzać obliczenia. Oprogramowanie analizuje rozszerzoną macierz i tworzy sekwencję kroków związanych z eliminacją Gaussa. W zależności od typu układu, wynik końcowy może być przedstawiony w następujący sposób:

- jednoznaczne rozwiązanie,
- informacja o braku rozwiązań,
- informacja o nieskończenie wielu rozwiązaniach.

W przypadku każdego etapu aplikacja pokazuje:

- aktualny stan macierzy przeprowadzonej operacji,
- zapis operacji wierszowej w formie matematycznej,
- krótkie wyjaśnienie, co jest celem danego kroku (np. „zerowanie elementów w kolumnie poniżej elementu wiodącego”).

2.5.3. Nawigacja po krokach rozwiązania

Aplikacja pozwala na oglądanie rozwiązania w sposób etapowy. Użytkownik ma możliwość:

- wrócić do wcześniejszego kroku oraz przejść do kolejnego,
- zresetować przegląd kroków do jego pierwotnego stanu.

W interfejsie znajduje się wskaźnik postępu (np. „Krok 7/15”), co ułatwia zrozumienie długości rozwiązania oraz w miejscu, w którym obecnie znajduje się użytkownik.

2.5.4. Wizualizacja: wyróżnianie pivota i elementów macierzy

Kluczowym aspektem nauczania jest wizualne podkreślanie składników macierzy powiązanych z bieżącym etapem redukcji. Program powinien:

- wyraźnie wskazać element wiodący (pivot),
- wyróżniać wiersze i kolumny związane z daną operacją,
- ułatwiać użytkownikowi zrozumienie relacji pomiędzy operacją wierszową, a zmianą w konkretnych komórkach macierzy.

Takie podświetlania wspierają zrozumienie procesu: użytkownik dostrzega nie tylko „co uległo zmianie", ale także „gdzie" i „dlaczego" ta zmiana została wprowadzona.

2.5.5. Prezentacja zapisu matematycznego

W aplikacji kroki oraz operacje są przedstawiane w postaci matematycznej (przy użyciu notacji przypominającej LaTeX). Dzięki temu użytkownik uzyskuje zapis zgodny z materiałami akademickimi, a komunikaty są klarowane i prostsze do zestawienia z rozwiązaniami omawianymi podczas zajęć.

2.6. Wymagania niefunkcjonalne

Aplikacja musi spełniać standardy ważne dla narzędzia edukacyjnego:

- **Czytelność i przejrzystość**: interfejs nie powinien być przytłaczający dla użytkownika. Kroki, opisy i macierz muszą być wyraźnie oddzielone.
- **Spójność prezentacji**: oznaczenia i styl przedstawienia kroków powinny być jednolite we wszystkich przykładach.
- **Poprawność obliczeń**: aplikacja musi dostarczać prawidłowe wyniki matematyczne. Szczególnie ważne jest właściwe traktowanie ułamków i liczb ujemnych.
- **Odporność na błędne dane**: aplikacja powinna stabilnie radzić sobie z nieprawidłowymi danymi wejściowymi (np. zabronionymi znakami) bez zatrzymania działania i komunikatem zrozumiałym dla użytkownika.
- **Dostępność w przeglądarkach**: aplikacja powinna funkcjonować w standardowych przeglądarkach i nie wymagać instalacji dodatkowego oprogramowania po stronie użytkownika.
- **Zdolność do testowania i rozwoju**: rozwiązanie powinno być zaprojektowane w sposób modułowy, umożliwiający rozwój oraz automatyczną weryfikację.

2.7. Kryteria akceptacji

Przyjęto, że projekt spełnia wymagania, jeśli:

- użytkownik ma możliwość wyboru wielkości układu i może wprowadzić macierz rozszerzoną [A|b],
- puste miejsca są właściwie interpretowane jako zera, a niedozwolone znaki (litery i inne znaki specjalne) są blokowane lub zgłaszane jako komunikat,
- użytkownik ma możliwość wprowadzania liczb całkowitych, dziesiętnych oraz ułamków a/b przy użyciu zaakceptowanych symboli (., ,, -, /),
- aplikacja tworzy i pokazuje pełną sekwencję kroków eliminacji Gaussa oraz końcowy wynik,
- jest dostępna opcja nawigacji między krokami (poprzednia/następna) oraz opcja resetu,
- w każdym etapie wyróżniane są elementy wiodące (pivot) oraz obszary macierzy związane z bieżącą operacją,
- aplikacja prawidłowo identyfikuje układ z jednym rozwiązaniem, brakiem rozwiązań oraz nieskończoną liczbą rozwiązań.

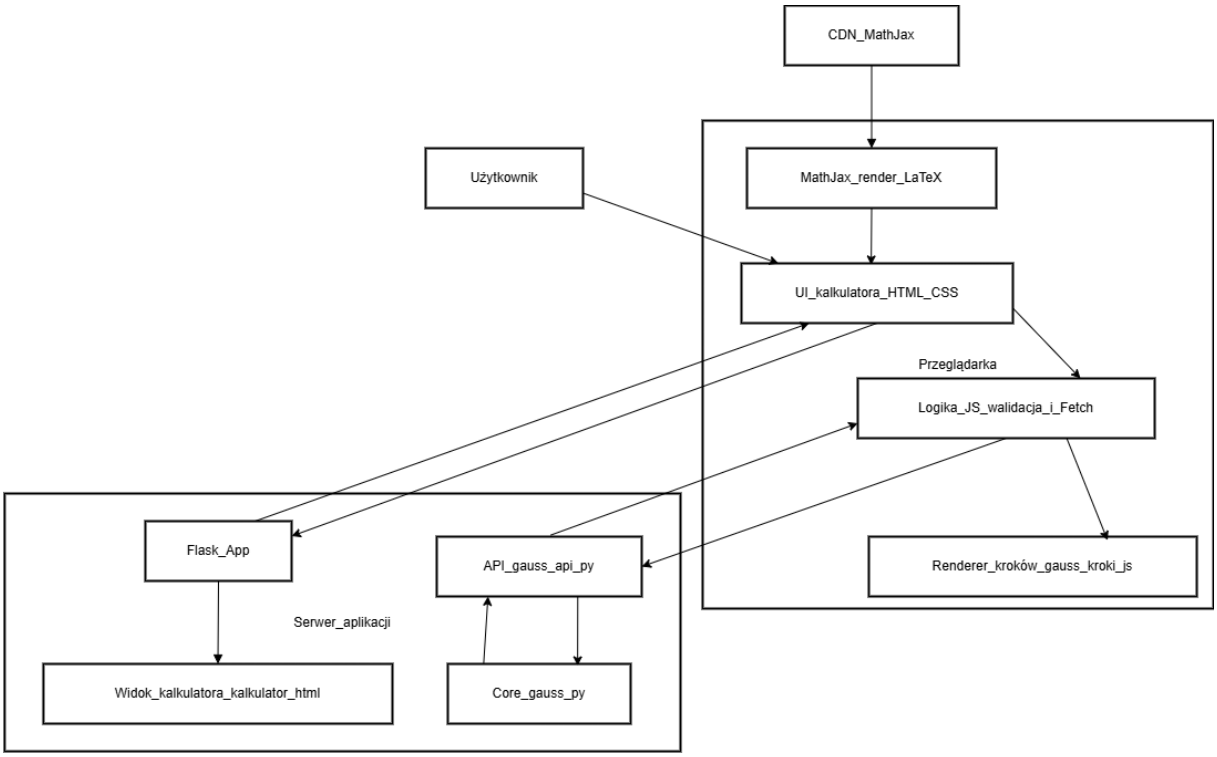
3. Projekt i architektura aplikacji

3.1. Założenia architektoniczne

Projektowana aplikacja jest narzędziem edukacyjnym, które ma na celu wspomaganie nauki metody eliminacji Gaussa. Istotnym elementem projektu było podzielenie odpowiedzialności na trzy warstwy:

- **warstwę obliczeniową** – która odpowiada za realizację algorytmu eliminacji Gaussa oraz generowanie kroków pośrednich,
- **warstwę usługową (API)** – która umożliwia korzystanie z funkcji obliczeniowych poprzez interfejs HTTP oraz wymianę danych w formacie JSON,
- **warstwę prezentacji** – która zawiera interfejs webowy umożliwiający wprowadzanie danych, uruchomienie obliczeń oraz przeglądanie kroków w sposób edukacyjny.

Aplikacja została opracowana jako rozwiązanie webowe w architekturze klient–serwer. Po stronie serwera wykorzystano framework Flask [3], a interfejs użytkownika stworzona przy użyciu HTML, CSS oraz JavaScript [10][11][12]. Komunikacja między warstwą prezentacyjną a warstwą usługową zachodzi poprzez żądania HTTP (w tym poprzez Fetch Api) [13], a dane przesyłane są w formacie JSON [16]. Taki sposób podejścia zapewnia przejrzysty przepływ danych i pozwala na niezależny rozwój zarówno algorytmu, jak i interfejsu.



Rysunek 3.1.: Diagram komponentów aplikacji (frontend-backend-moduł obliczeniowy-renderowanie wzorców).

3.2. Struktura projektu i podział na moduły

Projekt został zbudowany w sposób modułowy. W strukturze katalogów można wyróżnić kilka grup, które odpowiadają za różnorodne funkcje aplikacji:

- **uruchomienie i konfiguracja aplikacji**
  - `main.py` – główny plik aplikacji Flask, który zajmuje się inicjalizacją oraz rejestrowaniem tras i modułów,
  - `requirements.txt` – zestaw wymagań dotyczących środowiska uruchomieniowego.
- **warstwa API (REST)**
  - `api/__init__.py` – ustawienia dla modułu API (Blueprint) [3],
  - `api/gauss_api.py` – przetwarzanie żądania, które oblicza kroki eliminacji Gaussa i zwraca wynik w formacie JSON [16].
- **warstwa obliczeniowa**
  - `core/gauss.py` – implementacja eliminacji Gaussa, tworzenie kolejnych kroków oraz interpretacja wyników.
- **warstwa widoków (strony HTML)**
  - `templates/` - szablony HTML używane w aplikacji (w tym `base.html` oraz inne specjalistyczne podstrony). W aplikacjach Flask szablony są przetwarzane przez silnik Jinja2 [4],
  - `calculator/routes/` - trasy (Blueprint) dla interaktywnego kalkulatora.
- **zasoby statyczne**

    - static/css/ i static/js/ - pliki ze stylami i skryptami ogólnymi, static/shared/gauss-kroki.css, static/shared/gauss-kroki.js – komponent obsługujący wyświetlanie kroków oraz ich wyróżnienia w macierzy,
    - static/shared/mathjax-init.js – dostosowanie MathJax [14], które umożliwia renderowanie wzorów w notacji LaTeX; biblioteka może być pozyskiwania z CDN (np. jsDelivr) [15].
- **testy**
    - tests/ - zestaw testów jednostkowych dotyczących kluczowych aspektów projektu, w tym logiki obliczeniowej oraz API (pytest) [7].

Taki podział sprzyja utrzymaniu porządku w projekcie, a także ułatwia rozwój i testowanie różnych elementów w sposób niezależny.

3.3. Warstwa serwerowa

Backend aplikacji został stworzony przy użyciu Flask [3]. Serwer wykonuje dwie główne funkcje:

1. **Dostarcza strony oraz zasoby statyczne** – szablony HTML (generowane przez Jinja2) [4] oraz pliki CSS/JavaScript [10][11][12].
2. **Udostępnia API** – endpoint przyjmuje dane wejściowe (macierz rozszerzoną) i zwraca rezultaty w postaci listy kroków w formacie JSON [16].

W projekcie wprowadzono podział funkcji na moduły (Blueprinty) [3]. Dzięki temu kalkulator ma możliwość posiadania własnej sekcji tras oraz własnych zasobów statycznych, a API może być rozwijane jako odrębna część aplikacji, niezależna od warstwy prezentacji.

3.4. Warstwa obliczeniowa

Najistotniejszym elementem aplikacji jest moduł core/gauss.py, w którym wdrożono metodę eliminacji Gaussa. W przeciwieństwie do standardowych kalkulatorów, celem nie jest wyłącznie podanie ostatecznego rezultatu, lecz opracowanie kompletnej sekwencji kroków dydaktycznych.

3.4.1. Dokładność obliczeń i ułamki wymierne

W aplikacji zastosowano typ Fraction z standardowej biblioteki Pythona [2], co umożliwia precyzyjne wykonywanie obliczeń na liczbach wymiernych. Jest to ważne z perspektywy edukacyjnej, ponieważ:

- eliminowane są błędy wynikające z zaokrągleń, które występują w liczbach zmiennoprzecinkowych,
- pozwala na przedstawienie rezultatów w formie ułamków, które odpowiadają obliczeniom wykonywanym podczas zajęć.

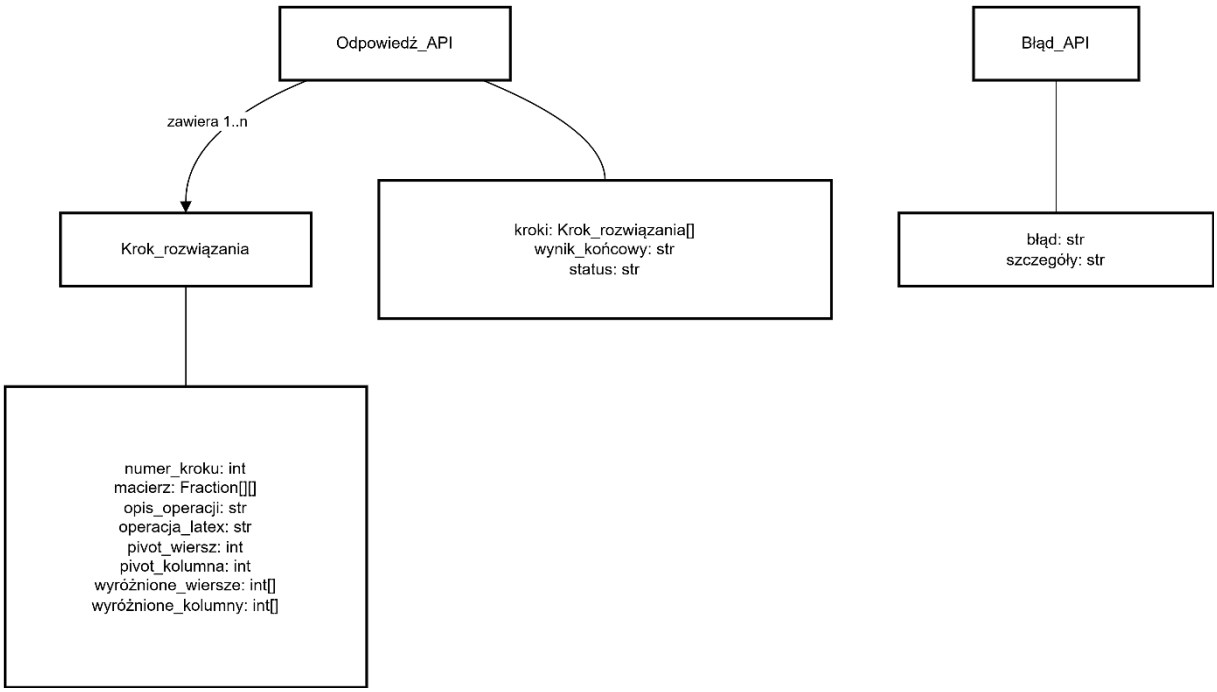
W praktyce użytkownik może wprowadzać ułamki w zapisie a/b, a algorytm interpretuje je jako wartości wymierne, zachowując pełną dokładność przekształceń [2].

3.4.2. Model danych kroku rozwiązania

Każdy etap obliczeń jest przedstawiany przez strukturę danych, która zawiera wszystkie elementy niezbędne do wizualizacji w interfejsie użytkownika. W szczególności każdy etap obejmuje:

- bieżący stan macierzy rozszerzonej,
- opis wykonywanych operacji (np. „zerujemy elementy w kolumnie poniżej elementu wiodącego”),
- informację o lokalizacji elementu wiodącego (pivotu),
- dodatkowe dane do wyróżniania wierszy i kolumn w UI.

Takie podejście pozwala na spójne i klarowne pokazanie przebiegu algorytmu: użytkownik może dostrzec formalną czynność, jak i jej wpływ na macierz.



Rysunek 3.2.: Diagram klas przedstawiający logiczny model danych kroku rozwiązania zwracany przez API.

3.5. Warstwa usługowa

Komunikacja między frontendem (JavaScript), a backendem (Flask) realizowana jest za pomocą interfejsu API w standardzie REST. Użytkownik Wysyła do serwera dane macierzy w formacie JSON [16], a serwer odsyła odpowiedź również w tym samym formacie, zawierającą listę kroków oraz informacje o wyniku.

Po stronie klienta, zapytanie do API jest przeprowadzane przy użyciu Fetch API [13]. Taki sposób komunikacji ułatwia integrację: interfejs nie musi zajmować się obliczeniami, lecz wystarczy, że potrafi przesyłać dane i wyświetlić wynik.

W przypadku wystąpienia błędów, takich jak niewłaściwy format liczby lub błędny rozmiar macierzy, API powinno dostarczać jasny komunikat, który można pokazać użytkownikowi. Dzięki temu unika się sytuacji, w której aplikacja przestaje działać bez podania przyczyny problemu.

3.6. Warstwa klienta

Interfejs dla użytkowników funkcjonuje w przeglądarkach i został stworzony przy pomocy HTML, CSS, JavaScript [10][11][12]. Najistotniejszym elementem aplikacji z punktu widzenia użytkownika jest interaktywny kalkulator do eliminacji Gaussa.

3.6.1. Walidacja danych wejściowych

Kluczowym aspektem użyteczności jest weryfikacja danych w trakcie wprowadzania wartości do macierzy. W aplikacji ustalono następujące zasady:

- pusta komórka jest traktowana jako wartość 0,
- niedozwolone są litery (np. „abc…”) oraz różnorodne znaki specjalne,
- akceptowane są znaki: `.`, `,`, `-`, `/`,
- znak `/` jest używany do zapisu ułamków w formie `a/b`.

Te ograniczenia redukują liczbę błędów po stronie API oraz zwiększa komfort użytkownika, ponieważ błędy są sygnalizowane natychmiast.

3.6.2. Prezentacja kroków i wyróżnianie pivotów

Aplikacja prezentuje rozwiązanie krok po kroku. W widoku wyróżniane są elementy istotne dla aktualnej operacji:

- element wiodący (pivot),
- wiersze aktualnie przekształcane,
- obszary macierzy związanych z eliminacją współczynników.

Mechanizm ten sprawia, że użytkownik może łatwo śledzić, które elementy macierzy są modyfikowane oraz jaki jest cel danego etapu obliczeń.

3.7. Renderowanie notacji matematycznej

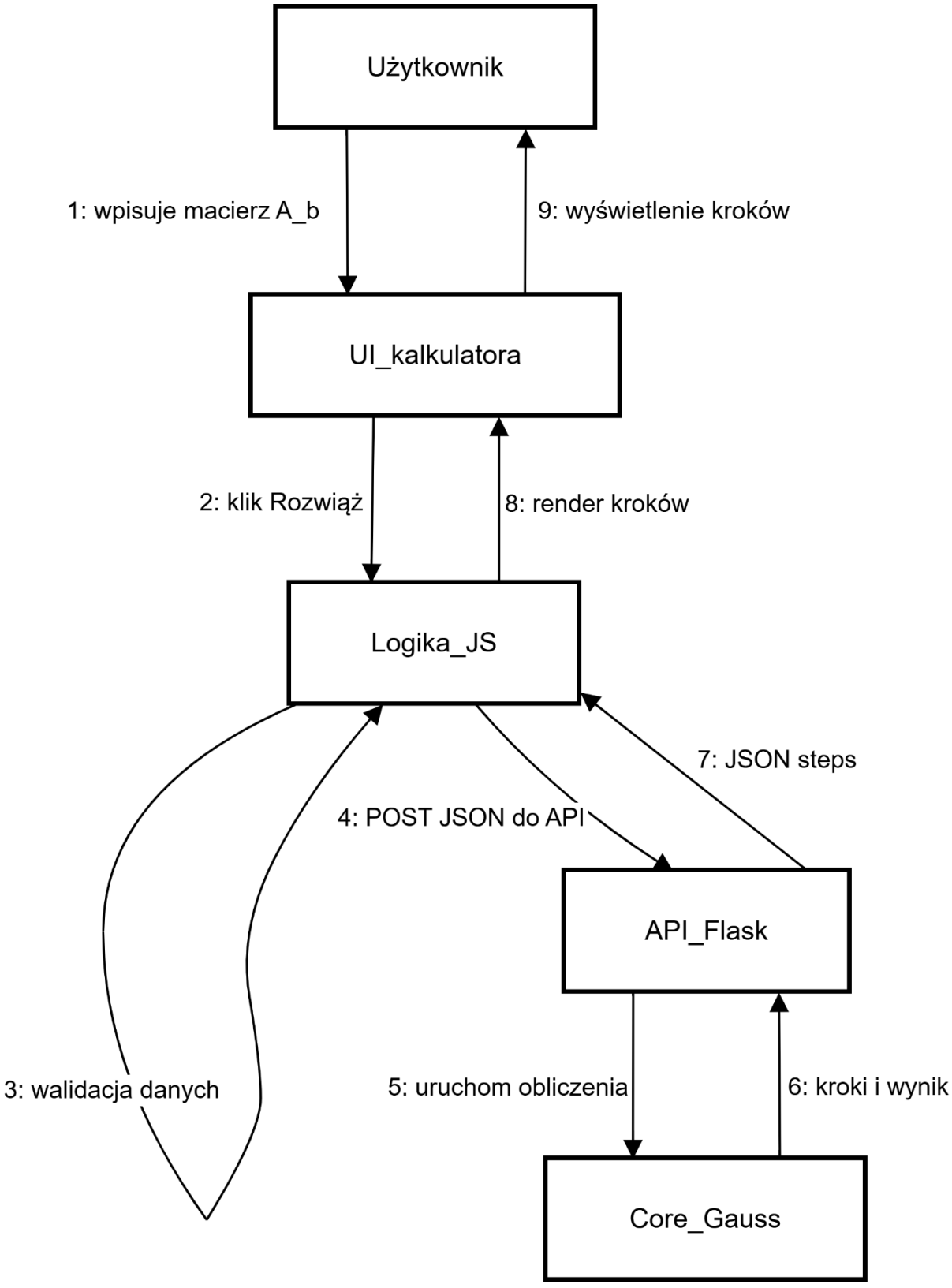
W tej aplikacji wykorzystano MathJax do wyświetlania wzorów matematycznych w formacie LaTeX w przeglądarkach [14]. To pozwala na umieszczanie opisów kroków, które zawierają operacje wierszowe zgodnie z notacją używaną podczas zajęć.

MathJax może być załadowany przez sieć CDN, na przykład jsDelivr [15], a jego ustawienia są zapisane w pliku mathjax-init.js. Użycie tej biblioteki poprawia przejrzystość materiału edukacyjnego w porównaniu do pokazywania wzorów jako zwykłego tekstu.

3.8. Przepływ danych

Działanie kalkulatora można ująć w następujące etapy:

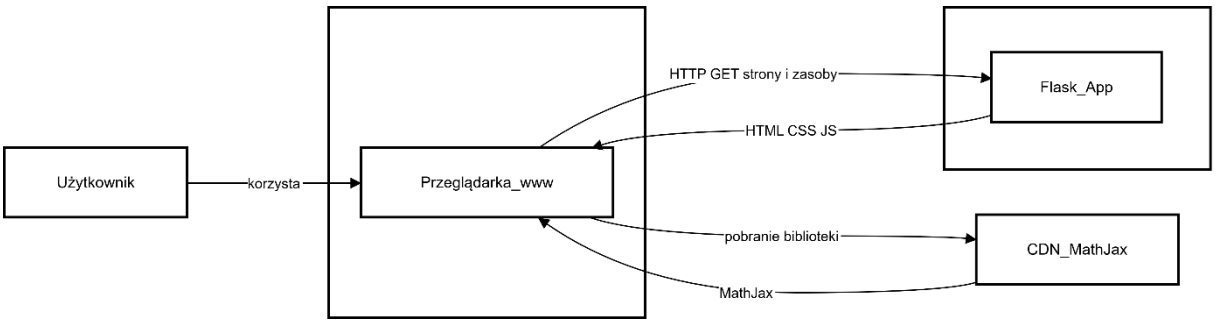
1. Użytkownik decyduje o wymiarach układu i wprowadza macierz rozszerzoną [A|b].
2. Frontend przeprowadza weryfikację danych wejściowych (format licz, dozwolone znaki) [10][12].
3. Frontend wysyła zapytanie do API za pomocą Fetch API [13].
4. Backend aktywuje algorytm eliminacji Gaussa (core/gauss.py) i tworzy poszczególne kroki.
5. Backend przekazuje wynik w formacie JSON [16].
6. Frontend wizualizuje kolejne etapy: macierz, opis operacji, podświetlania (pivot, wiersze, kolumny).



Rysunek 3.3.: Diagram sekwencji scenariusza „Rozwiąż układ” (walidacja → API → obliczenia → render kroków).

3.9. Wdrożenie i środowisko uruchomieniowe

Aplikacja ma możliwość działania w lokalnym środowisku lub na serwerze. Użytkownik stosuje przeglądarkę internetową, a wszystkie obliczenia odbywają się na serwerze aplikacji, który działa w Flask [3]. Kolejnym składnikiem architektury jest pobranie biblioteki MathJax z zewnętrznego źródła (CDN) [15], co zmniejsza rozmiar projektu i upraszcza aktualizację biblioteki.



Rysunek 3.4.: Diagram wdrożenia aplikacji (przeglądarka-serwer Flask-CDN dla MathJax).

4. Implementacja

4.1. Warstwa serwerowa i punkt końcowy API

Aplikacja udostępnia endpoint API, którego zadaniem jest przetworzenie macierzy rozszerzonej [A|b] w formacie JSON, wykonanie algorytmu eliminacji Gaussa na serwerze oraz zwrócenie listy kroków rozwiązania. Implementacja została zrealizowana przy użyciu frameworka Flask.

Głównym komponentem warstwy API jest funkcja obsługująca żądanie POST /gauss/kroki. Na początku następuje odczyt danych w formacie JSON, a następnie weryfikacja czy pole matrix jest obecne. W przypadku wystąpienia błędów, zwracany jest spójny obiekt error wraz z kodem HTTP.

```
from flask import request, jsonify
from . import api_bp
from core.gauss import gauss_steps

@api_bp.post("/gauss/kroki")
def gauss_kroki():
    try:
        data = request.get_json(force=True, silent=False)
    except Exception as e:
        return jsonify({"error": {"code": "BAD_JSON", "message": str(e)}}), 400

    matrix = (data or {}).get("matrix")
    if not matrix:
        return  jsonify({"error":  {"code":  "NO_MATRIX",  "message":  "Brak  pola
'matrix'."}}), 400

    try:
        result = gauss_steps(matrix)
        if "error" in result:
            code = result["error"].get("code", "")
            status = 400 if code in ("PARSE_ERROR", "BAD_SHAPE") else 500
            return jsonify(result), status
        return jsonify(result), 200
    except Exception as e:
        return  jsonify({"error": {"code": "SERVER_ERROR", "message": str(e)}}),
500
```

Rysunek 4.1.: Endpoint API zwracający kroki eliminacji Gaussa.

Z perspektywy działania aplikacji to podejście jest użyteczne: interfejs zawsze otrzymuje odpowiedź w formacie JSON i może wyświetlić komunikat użytkownikowi (np. o nieprawidłowych danych) bez odwoływania się do „surowych” wyjątków.

4.2. Reprezentacja liczb i parsowanie danych wejściowych

Aplikacja przyjmuje wartości wpisane w komórkach, a następnie przekształca je do formatu Fraciton. Dzięki temu obliczenia są przeprowadzane precyzyjnie, co ma istotne znaczenie edukacyjne: użytkownik ma możliwość zobaczenia ułamków w formie, która nie jest zaokrąglona do dziesiętnych wartości.

W funkcji parsującej wzięto pod uwagę:

- pustą komórkę traktowaną jako 0,
- zmianę przecinka dziesiętnego (zamiana , → .),
- zapis ułamków w postaci a/b.

```
from fractions import Fraction

def _to_frac(s: str) -> Fraction:
    s = (s or "0").strip().replace(",", ".")
    if "/" in s:
        a, b = s.split("/", 1)
        return Fraction(int(a), int(b))
    return Fraction(s)
```
Rysunek 4.2.: Parsowanie wartości tekstowej do Fraction.

4.3. Formatowanie wartości do interfejsu i notacji matematycznej

Aby użytkownik otrzymał czytelny zapis, w implementacji występują dwa formaty:

1. zapis tekstowy do wyświetlania w tabeli (np. 5/3),
2. zapis LaTeX (np. \tfrac{5}{3}) do renderowania przez MathJax

```
def _fstr(x: Fraction) -> str:
    if x.denominator == 1:
        return str(x.numerator)
    return f"{x.numerator}/{x.denominator}"

def _latex_frac(x: Fraction) -> str:
    if x.denominator == 1:
        return str(x.numerator)
    sgn = "-" if x < 0 else ""
    a = abs(x.numerator)
    b = x.denominator
    return f"{sgn}\\tfrac{{{a}}}{{{b}}}"
```
Rysunek 4.3.: Funkcje formatujące ułamki do UI i do LaTeX.

Dodatkowo dodano funkcję pomocniczą, która wymusza zapis „inline” w LaTeX (w nawiasach \(…\)), co ułatwia przedstawienie krótkich opisów poszczególnych kroków.

```
def _inline(tex: str) -> str:
    if tex is None:
        return None
    t = tex.strip()
    if t.startswith("\\(") and t.endswith("\\)"):
        return t
    return f"\\({t}\\)"
```
Rysunek 4.4.: Wymuszenie zapisu LaTeX inline.

4.4. Serializacja macierzy i model pojedynczego kroku rozwiązania

Wynik pracy algorytmu nie jest przedstawiany jako „suche” liczby, lecz jako sekwencja kroków. Każdy krok zawiera zarówno stan macierzy, jak i dodatkowe informacje do wizualizacji, na przykład pivot oraz zakres wyróżnień.

Najpierw macierz jest przekształcana na format tekstowy (np. Fraction(5,3) → „5/3”), aby mogła być przesłana w formacie JSON bezpośrednio.

```
def _serialize_aug(m):
    return [[_fstr(v) for v in row] for row in m]
```

Rysunek 4.5.: Serializacja macierzy rozszerzonej do listy stringów.

Każdy krok jest budowany przez funkcję _krok(…), która zapewnia jednolity schemat danych. Szczególnie istotne są pola:

- desc_tex – opis kroku (HTML + LaTeX),
- op_chip_tex – skrót operacji (np. do etykiety w UI),
- highlight – pozycja pivotu (row, col),
- shade_rows, shade_cols – podświetlenia wierszy/kolumn.

```
from typing import List, Dict

def _krok(steps: List[Dict], **kwargs):
    steps.append({
        "aug": kwargs.pop("aug"),
        "desc_tex": kwargs.pop("desc_tex", None),
        "op_chip_tex": kwargs.pop("op_chip_tex", None),
        "side_eq_tex": kwargs.pop("side_eq_tex", None),
        "highlight": kwargs.pop("highlight", None),
        "shade_rows": kwargs.pop("shade_rows", []),
        "shade_cols": kwargs.pop("shade_cols", []),
        "type": kwargs.pop("type", "step"),
        **kwargs
    })
```

Rysunek 4.6.: Struktura kroku rozwiązania.

4.5. Walidacja rozmiaru macierzy [A|b] i krok inicjalizacji

Funkcja gauss_steps(…) zaczyna od próby sparsowania danych wejściowych. Jeżeli użytkownik poda niepoprawne wartości, algorytm zwraca błąd PARSE_ERROR. Następnie sprawdzany jest rozmiar macierzy $n \times (n + 1)$.

```
def gauss_steps(aug_matrix: List[List[str]]) -> Dict[str, any]:
    try:
        M = [[_to_frac(x) for x in row] for row in aug_matrix]
    except Exception as e:
        return {"error": {"code": "PARSE_ERROR", "message": f"Nie udało się
sparsować liczb: {e}"}}

    n = len(M)
    if n == 0 or any(len(r) != n + 1 for r in M):
        return {"error": {"code": "BAD_SHAPE", "message": "Macierz rozszerzona
musi mieć rozmiar n×(n+1)."}}

    steps: List[Dict] = []

    _krok(steps,
          type="init",
          desc_tex="Macierz rozszerzona \\([A|b]\\) na wejściu.",
          aug=_serialize_aug(M))
```

Rysunek 4.7.: Walidacja wejścia i pierwszy krok „init”.

4.6. Eliminacja w przód

W eliminacji w przód pętla przechodzi po kolumnach i wybiera pivot. Pivot jest zapamiętywany i używany do wizualizacji. Po wyborze pivotu dodawany jest krok typu pivot, a jego pozycja trafia do highlight, co umożliwia podświetlenie elementu wiodącego w UI.

```
row = 0
pivots = []

for col in range(n):
    pivot = None
    best_abs = Fraction(0)
    for r in range(row, n):
        v = M[r][col]
        if v != 0 and abs(v) > best_abs:
            best_abs = abs(v)
            pivot = r
    if pivot is None:
        continue

    if pivot != row:
        M[row], M[pivot] = M[pivot], M[row]

    _krok(steps,
          type="pivot",
          desc_tex=f"Wybieramy element wiodący \\(K_{{{col + 1}}}\\), znajduje
się w \\(W_{{{row + 1}}}\\).",
          aug=_serialize_aug(M),
          highlight={"row": row, "col": col})
```

Rysunek 4.8.: Wybór pivotu i krok „pivot”

Następnie wiersz pivotu jest normalizowany do wartości 1 (jeśli pivot ≠ 1). Operacja jest opisana w op_chip_tex.

```
piv_before = M[row][col]
if piv_before != 1:
    mul = Fraction(1, 1) / piv_before
    for c in range(col, n + 1):
        M[row][c] *= mul

    _krok(steps,
          type="normalize",
          desc_tex=f"Normalizujemy \\(W_{{{row + 1}}}\\), aby element wiodący był
równy \\(1\\).",
          op_chip_tex=_inline(
              f"K_{{{col + 1}}}:\\; W_{{{row + 1}}} \\leftarrow W_{{{row + 1}}}
\\cdot {_latex_frac(mul)}"),
          aug=_serialize_aug(M),
          highlight={"row": row, "col": col},
          shade_rows=[row])
```

Rysunek 4.9.: Normalizacja wiersza pivotu.

Na końcu wykonywana jest eliminacja elementów poniżej pivotu. Dla każdego wiersza, gdzie element w kolumnie pivotu jest niezerowym wykonywana jest operacja wierszowa:

$$W_r \leftarrow W_r - factor \cdot W_{pivot}$$

a sama operacja jest zapisywana w liście ops_tex do późniejszego przedstawienia w UI.

```
ops_tex = []
changed_rows = []
for r in range(row + 1, n):
    if M[r][col] == 0:
        continue
    factor = M[r][col]
    for c in range(col, n + 1):
        M[r][c] -= factor * M[row][c]
    ops_tex.append(
        f"W_{{{r + 1}}} \\leftarrow W_{{{r + 1}}} - ({_latex_frac(factor)})\\,
W_{{{row + 1}}}"
    )
    changed_rows.append(r)

if changed_rows:
    sh = [row, changed_rows[-1]] if changed_rows else [row]
    _krok(steps,
          type="eliminate_block",
          desc_tex=f"Zerujemy elementy w kolumnie \\(K_{{{col + 1}}}\\) poniżej
elementu wiodącego.",
          op_chip_tex=_inline(f"K_{{{col    +    1}}}:\\;    "    +    "\\;,\\;
".join(ops_tex)),
          aug=_serialize_aug(M),
          highlight={"row": row, "col": col},
          shade_rows=sh)
```

```
pivots.append((row, col))
row += 1
```
Rysunek 4.10.: Zerowanie elementów poniżej pivotu + opis operacji wierszowych.

4.7. Wczesne wykrywanie sprzeczności

Po eliminacji w przód sprawdzany jest klasyczny warunek sprzeczności: wiersz zerowy w części współczynników i niezerowa wartość po prawej stronie.

```
for r in range(n):
    if all(M[r][c] == 0 for c in range(n)) and M[r][n] != 0:
        _krok(steps, type="result",
              desc_tex="Układ sprzeczny – brak rozwiązań.",
              aug=_serialize_aug(M))
        return {"steps": steps}
```
Rysunek 4.11.: Wykrywanie sprzeczności i zakończenie działania algorytmu.

4.8. Eliminacja wstecz

Aby przygotować dane do podstawiania wstecznego, wykorzystywana jest lista pivotów. Pętla przechodzi po pivotach w odwrotnej kolejności i zeruje elementy **nad** pivotem.

```
macierz_trojkatna = [wiersz[:] for wiersz in M]

for r, c in reversed(pivots):
    for rr in range(r - 1, -1, -1):
        if M[rr][c] != 0:
            factor = M[rr][c]
            for k in range(c, n + 1):
                M[rr][k] -= factor * M[r][k]
```
Rysunek 4.12.: Eliminacja wstecz na podstawie listy pivotów.

4.9. Podstawienie wsteczne z generowaniem objaśnień

Po uzyskaniu macierzy trójkątnej górnej wykonywane jest podstawienie wsteczne. W implementacji ważne jest to, że oprócz wyliczenia x_i budowany jest opis w LaTeX pokazujący sposób obliczenia.

```
rozwiazania = {}

for i in range(n - 1, -1, -1):
    wiersz = macierz_trojkatna[i]
    b = wiersz[n]

    skladniki = []
    skladniki_tex = []
```

```
    suma = Fraction(0)

    skladniki_zmiennych_tex: List[str] = []
    skladniki_wartosci_tex: List[str] = []

    for j in range(i + 1, n):
        wspolczynnik = wiersz[j]
        if wspolczynnik != 0:
            wartosc_xj = rozwiazania[j]
            suma += wspolczynnik * wartosc_xj

            skladniki.append((wspolczynnik, j, wartosc_xj))
            skladniki_tex.append(f"{_latex_frac(wspolczynnik)} \\cdot
{_latex_frac(wartosc_xj)}")
            skladniki_zmiennych_tex.append(_latex_iloczyn(wspolczynnik, f"x_{{{j
+ 1}}}"))
            skladniki_wartosci_tex.append(_latex_iloczyn(wspolczynnik,
_latex_frac(wartosc_xj)))

    xi = b - suma
    rozwiazania[i] = xi
```

Rysunek 4.13.: Podstawienie wsteczne – budowa opisu i wartości x_i.

Jeżeli występują składniki sumy, tworzony jest pełny zapis obliczeń, pokazujący przejścia od wyrażenia ze zmiennymi do podstawienia wartości.

```
if skladniki:
    zmienne_tex = _polacz_wyrazenia(skladniki_zmiennych_tex)
    wartosci_tex = _polacz_wyrazenia(skladniki_wartosci_tex)
    obliczenia = (
        "\\["
        f"x_{{{i + 1}}} = {_latex_frac(b)} - \\left({zmienne_tex}\\right)"
        f" = {_latex_frac(b)} - \\left({wartosci_tex}\\right)"
        f" = {_latex_frac(b)} - {_latex_frac(suma)}"
        f" = {_latex_frac(xi)}"
        "\\]"
    )
    desc = f"Wyznaczamy \\(x_{{{i + 1}}}\\) z równania w \\(W_{{{i +
1}}}\\).<br/>{obliczenia}"
else:
    obliczenia = f"\\[x_{{{i + 1}}} = {_latex_frac(b)}\\]"
    desc = f"Wyznaczamy \\(x_{{{i + 1}}}\\) z równania w \\(W_{{{i +
1}}}\\).<br/>{obliczenia}"
```

Rysunek 4.14.: Generowanie opisu obliczeń w LaTeX.

Na końcu dodawany jest krok typu back_substitute. Poza opisem zawiera on też side_eq_tex (krótkie równanie do wyświetlenia obok) oraz metadane do wyróżnienia wiersza.

```
_krok(steps,
      type="back_substitute",
      desc_tex=desc,
      side_eq_tex=_inline(f"x_{{{i + 1}}} = {_latex_frac(xi)}"),
      aug=_serialize_aug(M),
      highlight={"row": i, "col": i},
      shade_rows=[i],
      target_row=i)
```

Rysunek 4.15.: Dopisanie kroku podstawiania wstecznego do listy kroków.

4.10. Klasyfikacja rozwiązania na podstawie rang i zwrot wyniku

Po wykonaniu obliczeń algorytm klasyfikuje przypadek układu na podstawie rang macierzy A i [A|b]. Rangi są wyznaczane przez analizę wierszy niezerowych.

```
def _rankA_rankAb(m) -> Tuple[int, int]:
    rankA = 0
    rankAb = 0
    for r in m:
        a_nz = any(c != 0 for c in r[:-1])
        ab_nz = a_nz or (r[-1] != 0)
        if a_nz:
            rankA += 1
        if ab_nz:
            rankAb += 1
    return rankA, rankAb
```

Rysunek 4.16.: Wyznaczanie rang macierzy A i [A|b].

Następnie dodawany jest krok końcowy typu result, zależny od przypadku:

- sprzeczny: brak rozwiązań,
- ranga mniejsza od liczby niewiadomych: nieskończenie wiele rozwiązań,
- w przeciwnym wypadku: rozwiązanie jednoznaczne.

```
rankA, rankAb = _rankA_rankAb(M)
if rankA < rankAb:
    _krok(steps, type="result",
          desc_tex="Układ sprzeczny – brak rozwiązań.",
          aug=_serialize_aug(M))
elif rankA < n:
    _krok(steps, type="result",
          desc_tex="Nieskończenie wiele rozwiązań.",
          aug=_serialize_aug(M))
else:
    sol = ",\\;".join([f"x_{{{i + 1}}}={_latex_frac(rozwiazania[i])}" for i in
range(n)])
    _krok(steps, type="result",
          desc_tex="Rozwiązanie układu równań:",
          aug=_serialize_aug(M),
          solution_tex=f"\\({sol}\\)")
```

```
return {"steps": steps}
```
Rysunek 4.17.: Krok końcowy result – trzy możliwe przypadki.

5. Testowanie

5.1. Założenia testowania

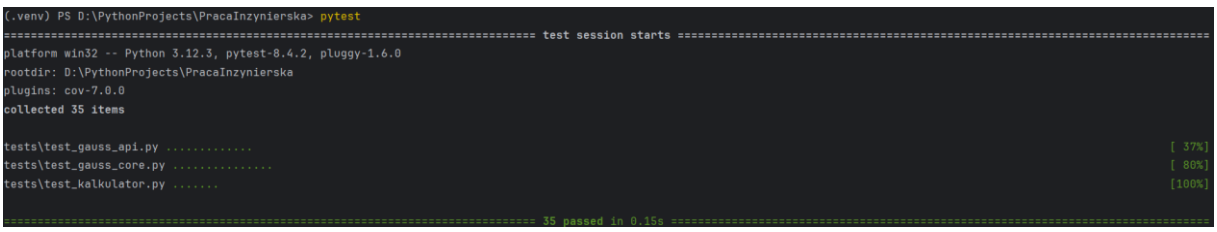
Testy w projekcie miały na celu potwierdzenie, że aplikacja funkcjonuje poprawnie z punktu widzenia użytkownika końcowego: co umożliwia wprowadzenie macierzy rozszerzonej [A|b], przesłanie danych do serwera, przeprowadzenie obliczeń z wykorzystaniem metody eliminacji Gaussa oraz zwracanie i wyświetlenie kroków pośrednich z końcowym wynikiem. Z tego powodu testy skupiono na trzech obszarach architekturze aplikacji:

- logika obliczeniowa
- interfejs API
- warstwa prezentacji

Do przeprowadzenia testów zastosowano framework pytest [7]. Pomiar pokrycia testami wykonano za pomocą narzędzi pytest --cov [8] oraz coverage.py [9]. Ponieważ aplikacja serwerowa została stworzona w oparciu o Flask [3], w testach wykorzystano testowego klienta tego frameworka, co umożliwia wielokrotne symulowanie żądań HTTP bez potrzeby uruchamiania przeglądarki.

5.2. Środowisko uruchomieniowe i sposób wykonywania testów

Testy uruchamiane były w środowisku lokalnym z interpreterem Python 3.12 [1] oraz zależnościami projektu. Podstawowe uruchomienie testów zostało wykonane poleceniem: pytest. W ramach uruchomienia zebrano łącznie 35 testów, które zakończyły się sukcesem (35 passed).



Rysunek 5.1.: Wynik uruchomienia testów (pytest) – 35/35 testów zakończonych sukcesem.

Dodatkowo uruchomiono pomiar pokrycia kodu testami: pytest --cov. Raport --cov (generowany przez pytest-cov [8] oraz coverage.py [9]) przedstawia liczbę wykonanych

instrukcji w poszczególnych plikach oraz wskazuje linie pominięte podczas testów. W praktyce umożliwia to ocenę, czy testy obejmują kluczowe ścieżki działania aplikacji.

5.3. Organizacja testów w projekcie

Testy zostały umieszczone w katalogu tests/ i podzielone według funkcji poszczególnych modułów:

- test_gauss_core.py – testy dotyczące algorytmu eliminacji Gaussa i zwracanych kroków,
- test_gauss_api.py – testy dotyczące endpointu API, które przetwarza macierz i zwraca wyniki,
- test_kalkulator.py – testy dostępności interfejsu kalkulatora oraz podstawowej struktury HTML,
- conftest.py – ustawienia testów, fixtury (np. klient testowy oraz zestaw macierzy).

Taki podział ułatwia rozwój projektu: przy modyfikacji logiki obliczeń głównie zmienia się testy warstwy obliczeniowej, natomiast przy aktualizacji komunikacji klient-serwer dostosowuje się testy API.

5.4. Testy interfejsu API

Aplikacja komunikuje się z klientem przez API, które przyjmuje dane w formacie JSON [16]. Z perspektywy działania kalkulatora istotne jest, aby:

1. dane wejściowe były prawidłowo walidowane,
2. użytkownik był informowany w sposób jasny o błędach,
3. odpowiedzi związane z prawidłowymi danymi zawierały kroki rozwiązania w jednolitym formacie.

Podczas testów API analizowane są zarówno właściwe scenariusze, jak i typowe błędy wejściowe: brak pola matrix, nieodpowiedni rozmiar macierzy, wartości, które nie mogą być przetworzone (np. litery) oraz niepoprawny format JSON. W każdym z tych przypadków testy weryfikują, że serwer zwraca odpowiedni kod HTTP oraz odpowiednio skonstruowany obiekt błędu w odpowiedzi.

```
def test_api_poprawne_200(klient, macierze_poprawne):
    r = postuj(klient, {"matrix": macierze_poprawne["prosty_2x2"]})
    assert r.status_code == 200
    body = r.get_json()
    assert "steps" in body
    assert isinstance(body["steps"], list)
    assert len(body["steps"]) > 0
```

Rysunek 5.2.: Przykład testu poprawnego żądania API.

Powyższy test sprawdza podstawowy wymóg prawidłowego funkcjonowania API: odpowiedź musi mieć status 200 i zawierać wykaz kroków. Z perspektywy użytkownika jest to istotne, ponieważ kroki te są później wyświetlane w przeglądarce.

Jednocześnie testy obejmują sytuacje, w których serwer powinien zwrócić kod 400 i zrozumiały komunikat o błędzie (zamiast nieprzewidzianego wyjątki).

```
@pytest.mark.parametrize("payload, kod", [
    ({}, 400),  # brak matrix
    ({"matrix": [["1", "2"], ["3", "4", "5"]]}, 400),  # zły kształt
    ({"matrix": [["a", "1", "2"], ["1", "2", "3"]]}, 400),  # nieparsowalne
])
def test_api_bledy_wejscia(klient, payload, kod):
    r = postuj(klient, payload)
    assert r.status_code == kod
    body = r.get_json()
    assert "error" in body
```

Rysunek 5.3.: Przykład testu walidacji danych wejściowych.

Zastosowanie parametryzacji w pytest [7] umożliwia zgromadzenie różnych rodzajów błędnych danych w jednym miejscu, co poprawia przejrzystość testów oraz ułatwia ich rozwój.

5.5. Testy warstwy obliczeniowej

Warstwa obliczeniowa zajmuje się realizacją eliminacji Gaussa oraz generowaniem kroków rozwiązania. W testach uwzględniono różnorodne układy, w tym zarówno klasyczne, jak i nietypowe:

- układy z jednoznacznymi rozwiązaniami,
- układy sprzeczne,
- układy, które prowadzą do nieskończonej liczby rozwiązań.

Kluczowym aspektem aplikacji jest praca z ułamkami oraz liczbami wymiernymi, co pozwala na uniknięcie błędów zaokrągleń, które występują w obliczeniach zmiennoprzecinkowych. W projekcie wykorzystano typ `Fraction` z bibliotek Pythona [2], co pozwala na zachowanie prawidłowości obliczeń oraz zapewnienie przejrzystości wyników z perspektywy edukacyjnej.

W testach oceniano nie tylko końcowy rezultat, ale także to, czy etapy rozwiązania są prezentowane w formie nadającej się do wyświetlania w interfejsie (np. czy występuje ostatni krok oznaczony jako „result”, czy lista kroków nie jest pusta oraz czy informacja o rodzaju układu odpowiada oczekiwaniom).

```
from fractions import Fraction

def test_poprawne_rozwiazanie_wymierne(macierze_poprawne):
    M = macierze_poprawne["ułamki_3x3"]
    wynik = gauss_steps(M)
    assert "error" not in wynik

    kroki = wynik["steps"]
    assert isinstance(kroki, list) and len(kroki) > 0

    # sprawdzenie, że w wyniku pojawiają się wartości wymierne (Fraction)
    macierze = [k.get("matrix") for k in kroki if "matrix" in k]
    assert any(isinstance(el, Fraction) for m in macierze for w in m for el in w)
```

Rysunek 5.4.: Przykład testu jednoznacznego rozwiązania i weryfikacji arytmetyki wymiernej.

5.6. Testy widoku kalkulatora

Ponieważ aplikacja oferuje widok kalkulatora, który jest tworzony jako strona HTML na serwerze, ważne jest, aby upewnić się, że:

- strona zwraca właściwą odpowiedź HTTP,
- HTML zawiera niezbędne elementy do prawidłowego działania interfejsu,
- wszystkie potrzebne zasoby statystyczne (skrypty i style) są dołączone.

Testy w tym zakresie obejmują wysłanie żądania GET do widoku kalkulatora i ocenę otrzymanej odpowiedzi.

```
def test_widok_kalkulatora_dostepny(klient):
    r = klient.get("/kalkulator/")
    assert r.status_code == 200

    html = r.data.decode("utf-8")
    assert "Rozwiąż" in html
    assert "Wyczyść" in html
    assert "kalkulator.js" in html
```

Rysunek 5.5.: Przykład testu dostępności widoku kalkulatora.

Tego typu testy umożliwiają identyfikację problemów takich, jak: nieprawidłowy routing, brak wzorca, błędna lokalizacja plików statycznych lub przypadkowe usunięcie elementów interfejsu, które są istotne dla użytkownika.

5.7. Pokrycie testami i interpretacja wyniku

Pomiar pokrycia kodu przy użyciu testów (wykonanie `pytest --cov`) wykazał, że najważniejsze moduły aplikacji są w dużym stopniu przetestowane. W szczególności:

- moduł zajmujący się API uzyskał pełne **pokrycie 100%**,
- moduł obliczeniowy uzyskał **pokrycie na poziomie 97%**.

W raporcie znajduje się także wartość `TOTAL`, która jest niższa, ponieważ zawiera również części modułów pomocniczych oraz przykładowych (np. dodatkowe ścieżki lub materiały demonstracyjne). Z perspektywy zakresu pracy istotne jest jednak, że moduły odpowiedzialne za kluczową funkcjonalność kalkulatora eliminacji Gaussa, czyli obliczenia i API, mają wysokie pokrycie [8], [9].

Podsumowanie

W ramach pracy zaprojektowano i zaimplementowano aplikację webową wspomagającą naukę algebry liniowej, skupiającą się na rozwiązywaniu układów równań liniowych metodą eliminacji Gaussa. Głównym celem projektu było opracowanie narzędzia o charakterze dydaktycznym, które nie ogranicza się do przedstawiania końcowego wyniku, a umożliwia użytkownikowi śledzenie całego procesu przekształcania macierzy rozszerzonej [A|b] krok po kroku. Wprowadzono opisy operacji wierszowych oraz wyróżniono element wiodący (pivot) oraz modyfikowane wiersze, co ułatwia zrozumienie działania algorytmu oraz łączy rachunki z ich interpretacją w formacie macierzowy.

Rozwiązanie zostało oparte na architekturze klient-serwer. Część serwerowa została zaimplementowana w języku Python z wykorzystaniem frameworka Flask i umożliwia dostęp do endpointów API, zwracających dane w formacie JSON. Warstwa obliczeniowa wykorzystuje arytmetykę wymierną (Fraction), która pozwala uzyskiwać dokładne wyniki pośrednie i końcowe bez błędów zaokrągleń. Warstwa prezentacji została stworzona za pomocą technologii HTML, CSS i JavaScript, a renderowanie notacji matematycznej zrealizowano za pomocą MathJax, umożliwiając prezentację kroków w formacie zbliżonym do zapisu stosowanego w materiałach akademickich.

W pracy przeprowadzono analizę istniejących narzędzi wspierającą naukę matematyki, co pozwoliło uzasadnić potrzebę stworzenia aplikacji skupionej wokół eliminacji Gaussa oraz przedstawiającej pośrednie kroki bez użycia płatnych funkcji. Udokumentowano cele i wymagania aplikacji, przedstawiono jej projekt oraz architekturę. Opisano kluczowe elementy implementacji w kontekście przepływu danych i sposobu generowania kroków. Poprawność działania aplikacji została zweryfikowana za pomocą testów jednostkowych obejmujących warstwę obliczeniową, API oraz widok kalkulatora.

Opracowane rozwiązanie spełnia założenie pracy i może stanowić praktyczne wsparcie w nauce eliminacji Gaussa dla studentów kierunków technicznych. Potencjalne kierunki dalszego rozwoju obejmują między innymi rozbudowę zestawu przykładów i ćwiczeń, rozszerzenie obsługi zadań o większy zakres algebry liniowej (np. rząd macierzy, wyznaczniki, macierz odwrotna), dodanie trybu nauki z elementami podpowiedzi czy sprawdzania kroków użytkownika oraz rozwój funkcji personalizacji poziomu szczegółowości objaśnień.

Bibliografia

[1] Python – dokumentacja Python 3.12, https://docs.python.org/3.12/, (data dostępu: 07.01.2026).

[2] Python – moduł fractions, https://docs.python.org/3.12/library/fractions.html, (data dostępu: 07.01.2026).

[3] Flask – dokumentacja, https://flask.palletsprojects.com/, (data dostępu: 07.01.2026).

[4] Jinja2 – dokumentacja, https://jinja.palletsprojects.com/, (data dostępu: 07.01.2026).

[5] Werkzeug – dokumentacja, https://werkzeug.palletsprojects.com/, (data dostępu: 07.01.2026).

[6] NumPy – dokumentacja, https://numpy.org/doc/, (data dostępu: 07.01.2026).

[7] pytest – dokumentacja, https://docs.pytest.org/, (data dostępu: 07.01.2026).

[8] pytest-cov – dokumentacja, https://pytest-cov.readthedocs.io/, (data dostępu: 07.01.2026).

[9] coverage.py – dokumentacja, https://coverage.readthedocs.io/, (data dostępu: 07.01.2026).

[10] MDN Web Docs – HTML, https://developer.mozilla.org/en-US/docs/Learn/Forms, (data dostępu: 07.01.2026).

[11] MDN Web Docs – CSS, https://developer.mozilla.org/en-US/docs/Web/CSS, (data dostępu: 07.01.2026).

[12] MDN Web Docs – JavaScript, https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model, (data dostępu: 07.01.2026).

[13] MDN Web Docs – Fetch API, https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API, (data dostępu: 07.01.2026).

[14] MathJax – dokumentacja, https://docs.mathjax.org/, (data dostępu: 07.01.2026).

[15] jsDelivr – CDN, https://www.jsdelivr.com/, (data dostępu: 07.01.2026).

[16] JSON – RFC 8259, https://www.rfc-editor.org/rfc/rfc8259, (data dostępu: 07.01.2026).

[17] WolframAlpha – strona serwisu, https://www.wolframalpha.com/, (data dostępu: 07.01.2026).

[18] GeoGebra – strona serwisu, https://www.geogebra.org/, (data dostępu: 07.01.2026).

[19] Desmos – strona serwisu, https://www.desmos.com/, (data dostępu: 07.01.2026).