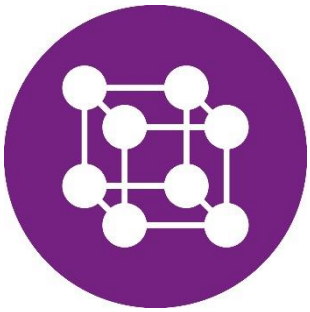


Politechnika Poznańska

Wydział Inżynierii Materiałowej i Fizyki Technicznej

PRACA DYPLOMOWA INŻYNIERSKA

Analiza numeryczna układu zawieszenia w pojeździe

Łukasz Lechna, Michał Wielgat

Promotor:
dr hab. Tomasz Stręk, prof. uczelni

Poznań 2023

Spis treści

STRESZCZENIE

W pracy przedstawiono Analizę numeryczną modelu matematycznego zawieszenia samochodowego z wykorzystaniem oprogramowania SageMath. Na początku przedstawiona istotność układów zawieszenia w pojazdach oraz krótko omówiono współcześnie stosowane rozwiązania w układach tłumiących wraz z ich graficzną reprezentacją. Następnie krótko opisana została historia programu SageMath, oraz możliwości jakie nam oferuje. Kolejno opisano jak z wykorzystaniem Sage wykonywać obliczenia na równaniach różniczkowych i wykonać ich eksport do czytelnej postaci wykresów. Na podstawie danych z przeanalizowanej literatury wykonane zostały modele graficzne, oraz matematyczne układów zawieszenia. Obliczenia wykonano dla dwóch modeli ćwiartki pojazdu z zawieszeniem pasywnym, z czego model drugi bierze pod uwagę również analizę reakcji pasażera na pracę badanego układu. Wykazano możliwości analizy numerycznej z wykorzystaniem oprogramowania komputerowego oraz przedstawiono korzyści płynące z tej szczególnej metody analizy.

ABSTRACT

The paper presents a numerical analysis of a mathematical model of a car suspension using the SageMath software. First was presented the importance of suspension systems in vehicles and briefly discusses the currently used solutions in damping systems along with their graphical representation. Then the history of the SageMath program and the possibilities it offers us were briefly described. Subsequently, it was described how to perform calculations on differential equations using Sage and export them in the form of readable graphs. Based on the data from the analyzed literature, graphical and mathematical models of the suspension systems were made. Calculations were made for two quarter models of the vehicle with passive suspension, of which the second model also takes into account the analysis of the passenger's reaction to the operation of the tested system. The possibilities of numerical analysis with the use of computer software have been demonstrated with illustrating benefits from using this particular method of analysis.

WSTĘP

Praca dyplomowa dotyczy zagadnienia Numerycznej analizy układu zawieszenia pojazdów samochodowych. Celem pracy było wyprowadzenie wybranych modeli matematycznych zawieszenia samochodowego i użycie ich w programie wykonanym w SageMath do wykonania numerycznej analizy pracy ćwiartki zawieszenia samochodowego poddanego okresowym wymuszeniom drogi. Rozdział pierwszy pracy jest krótkim wprowadzeniem w ramy zagadnienia. W drugim rozdziale przyjrzano się istotnym aspektom pracy zawieszeń samochodowych wraz z omówieniem najważniejszych funkcji jakie pełnią i problemami jakie napotykają ich konstruktorzy. Opisane zostały także współcześnie stosowane systemy zawieszeń oraz krótko omówiono wpływ drgań na pasażera pojazdu. Trzeci rozdział opisuje historię oraz możliwości programu SageMath, którego użyto do uzyskania wyników wyprowadzonych obliczeń wraz z reprezentacją w postaci kodu zapisanego w programie i uzyskanymi wynikami. W rozdziale czwartym omówiono metody obliczania równań różniczkowych w SageMath z wykorzystaniem zawartym w nim funkcji. W rozdziale piątym oraz szóstym wyprowadzono modele matematyczne badanych w pracy układów. Rozdziały siódmy oraz ósmy dotyczą analizy uzyskanych z użyciem programu wyników, wraz z ich reprezentacją w postaci wykresów wykreowanych w programie Sagemath.

1. WPROWADZENIE (OPRACOWAŁ ŁUKASZ LECHNA)

Motoryzacja jest z nami i rozwija się intensywnie od ponad stu lat i przez te ostatnich kilka dekad mogliśmy zaobserwować gwałtowny rozwój technologiczny na wielu polach związanych z tą dziedziną. Jednym z bardziej kluczowych pól rozwoju jest rozwój zawieszeń samochodowych, które od zawsze jest ich integralną częścią. Wpływa ono w znacznym stopniu na wygodę i walory użytkowe auta, swoją pracą tłumiąc nierówności drogi przenoszone na masę samochodu i pasażerów. Najbardziej znanym i rozpowszechnionym rodzajem zawieszeń pojazdów jest zawieszenie pasywne w którym to wymagany jest odpowiedni dobór współczynnika tłumienia, którego wartość spełnia dwa przeciwstawne sobie zadania: zapewnienia odpowiedniego komfortu i bezpieczeństwa jazdy. Badania zjawisk występujących w codziennym ruchu drogowym często są związane z koniecznością angażowania znacznych środków i posiadania rozbudowanego zaplecza technicznego dla badań drogowych, dzięki którym jest możliwe określenie zachowania badanego zawieszenia samochodowego. Wspomniane badania drogowe nie są zawsze konieczne, gdyż w ocenie większości cech związanych z pracą zawieszenia wystarczy dysponowanie odpowiednim modelem matematycznym który można poddać symulacji z wykorzystaniem środowiska wirtualnego. Takowy model odzwierciedla obiekt rzeczywisty z wykorzystaniem oprogramowania, którego obecnie dostępny jest szeroki wachlarz. Od strony mechanicznej zawieszenie samochodowe jest układem dynamicznym przenoszącym i tłumiącym drgania na pojazd i pasażerów wywołane przez nierówność podłoża. W celu zapewnienia odpowiedniego poziomu bezpieczeństwa i komfortu jazdy wykonuje się analizy tzw. modeli połówkowych lub ćwiartkowych pojazdów [**1,2,3**]. W niniejszej pracy wykonano analizę dwóch ćwiartkowych modeli matematycznych pojazdów wyposażonych w zawieszenia pasywne, wykorzystując w tym celu modele zaczerpnięte z literatury[**1,2**], wraz z porównaniem ich zachowań, i wpływu zmiany parametrów drogi na ich pracę.

2. UKŁAD ZAWIESZENIA W POJAZDACH I JEGO WPŁYW NA PASAŻERA (OPRACOWALI ŁUKASZ LECHNA, MICHAŁ WIELGAT)

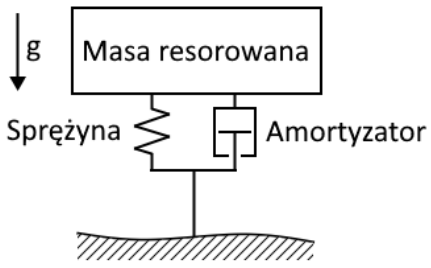
2.1 OPIS I CEL STOSOWANIA UKŁADÓW ZAWIESZENIA

Zawieszenia samochodowe w pojazdach mają kilka kluczowych zadań: ochronę pasażerów pojazdu od nadmiernie silnych wibracji spowodowanych jazdą po nierównej drodze, podtrzymywanie masy samochodu, oraz na utrzymaniu stabilności pojazdu podczas jazdy. Główne zawieszenie samochodu jest układem elementów w którego to skład, wchodzą między innymi elementy tłumiące oraz elementy sprężyste. Zadaniem układu zawieszenia jest izolowanie pasażerów auta od wpływu wibracji które są spowodowane nierównościami drogi po której porusza się jadący pojazd. Zawieszenia główne mają także drugie równie ważne zadanie którym jest zapewnienie stabilności pojazdu w określonych warunkach jazdy, co wiąże się z utrzymaniem odpowiednich standardów bezpieczeństwa przy poruszaniu się pojazdem. Wcześniej wspomniana stabilność jadącego pojazdu i komfort są dwoma najważniejszymi kryteriami oceny zawieszenia.

Często wyżej wspomniane czynniki jednak kolidują ze sobą utrudniając pracę inżynierom zajmującym się zaprojektowaniem układu. Układ o wysokim stopniu stabilności często okazuje się mało komfortowy dla pasażerów. Tymczasem zawieszenie zapewniające wysoki stopień komfortu może okazać się w pewnych warunkach jazdy silnie niestabilne przez co może doprowadzić do sytuacji niebezpiecznej. Przy projektowaniu zawieszeń pasywnych zawsze trzeba więc dążyć do kompromisu pomiędzy tymi dwoma czynnikami, gdyż nie jest możliwe uzyskanie idealnych rezultatów bez wykorzystania alternatywnych rozwiązań konstrukcyjnych które pozwoliłyby na dodatkową możliwość regulacji tych parametrów i usprawnienie działania układu. Takie alternatywne rozwiązania wykorzystano w innych rodzajach zawieszeń takich jak zawieszenia aktywne, które wymagają także osobnego układu sterowania które umożliwia adaptacje parametrów układu do warunków jazdy.

2.2 STOSOWANE WSPÓŁCZEŚNIE SYSTEMY ZAWIESZENIA

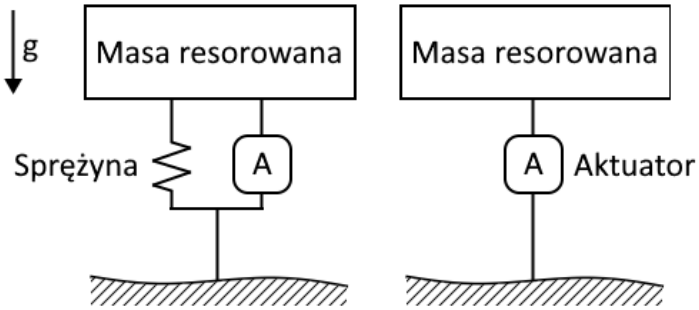
W samochodach powszechnie używane są pasywne systemy zawieszenia tłumiące ze względu na niewielkie koszty produkcji takiego układu i względnie zadowalające efekty. Taki układ opiera się zwykle na elementach o nieliniowym tłumieniu oraz liniowej sprężystości. W pasywnych elementach tłumiących wibracje ulegają rozproszeniu i ich energia jest zamieniana na energię cieplną. Tego typu zawieszenia obecnie stosowane są w różnorodnych konfiguracjach elementów sprężystych oraz tłumiących. Podczas jazdy współczynnik tłumienia spełniać bowiem musi tu dwa kolidujące wzajemnie zadania. Dla uzyskania dobrego komfortu pasażerów podczas jazdy wartość współczynnika powinna być wystarczająco niska by zawieszenie tłumiło drgania na wysokich częstotliwościach wymuszenia które są niekorzystne dla człowieka i negatywnie wpływają na wygodę podczas jazdy. Tymczasem aby pojazd zachowywał się stabilnie przy wykonywaniu manewrów na drodze współczynnik także musi być odpowiednio wysoki. Dobrze zaprojektowane zawieszenie pasywne posiada więc optymalnie dostosowany współczynnik wartości tłumienia osiągając odpowiedni kompromis pomiędzy komfortem pasażerów pojazdu a bezpieczeństwem jazdy.



Rysunek 2.1 Uproszczony schemat zawieszenia pasywnego [4].

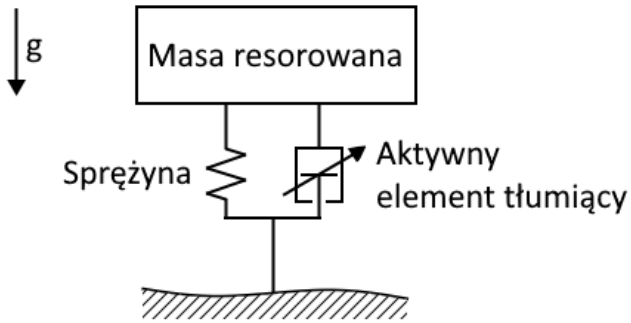
Zawieszenia aktywne to takie układy w których pasywne czy sprężyste elementy tłumiące mogą być zastępowane aktuatorem czyli sterowanym mechanicznie siłownikiem. Siła ma tam za zadanie neutralizowanie sił występujących w układzie oraz na aktywnym neutralizowaniu wymuszeń pochodzących od drogi powodując podwyższenie komfortu i bezpieczeństwa poruszania się pojazdem. Takie rozwiązanie znacznie podwyższa komfort i bezpieczeństwo jazdy jednak używanie go wiąże się ze znacznym poborem energii przez co nie jest ono rozwiązaniem preferowanym w

samochodach osobowych. Ze względu na zapewnianie wysokiego poziomu komfortu jest ono jednak stosowane w obiektach stacjonarnych z dostępem do zewnętrznego stałego źródła energii.



Rysunek 2.2Uproszczone schematy przykładowych systemów zawieszenia aktywnego [4].

Zawieszenia semiaktywne posiadają w sobie element sprężysty i tłumiący charakterystyczne dla zawieszeń pasywnych, z tą różnicą że wartość współczynnika tłumienia zawieszenia może być zmienna w czasie sprawiając że zawieszenie może elastycznie dostosować się w sposób ciągły do aktualnie panujących warunków drogowych podczas jazdy pojazdu. Zmiana jest wykonywana w sposób automatyczny przez specjalistyczny układ sterowania reagujący na warunki w których znajduje się pojazd za pośrednictwem zespołu czujników będących w stałej komunikacji ze sterownikiem. Zawieszenie semiaktywne nie generuje bezpośrednio siły kontrującej drgania jak w przypadku zawieszenia aktywnego, jednak znaczną przewagą tego zawieszenia nad układami w pełni aktywnymi jest jego niskie zapotrzebowanie energetyczne przy zapewnieniu bardzo wysokiej poprawy komfortu jazdy oraz poprawie stabilności pojazdu i bezpieczeństwa pasażerów. W układach semiaktywnych ważną rolę pełni odpowiednie skonfigurowanie układu sterowania gdyż optymalnie działający układ sterowania przyczynia się do znacznej poprawy parametrów pracy zawieszenia jednak niepoprawnie działający układ semiaktywny może także te wartości znacznie pogorszyć.



Rysunek 2.3 Uproszczony schemat zawieszenia semiaktywnego[4].

2.3 WPŁYW DRGAŃ NA PASAŻERA POJAZDU

Samochód jest skomplikowanym układem drgającym, drgania auta wywoływane są podczas jego eksploatacji w dużej mierze właśnie przez nierówność drogi po której pojazd się porusza. Drgania w zależności od amplitudy, częstości, czy czasu trwania mogą mieć wpływ na ludzki organizm wywołując zmiany fizykalne, psychologiczne czy fizjologiczne. Nagłe silne drgania wywołane przez np. szybki przejazd przez próg zwalniający mogą prowadzić do uszkodzeń ciała, takie jak naderwania mięśni, ukruszenia lub wybicia zębów, stłuczeń lub innych obrażeń. W przypadku wystawienia na długotrwałe drgania nawet o umiarkowanej intensywności, powodują dyskomfort i będą prowadzić do szybszego zmęczenia, zaburzenia skupienia i efektywności kierowania pojazdem. Organizm ludzki wykazuje różną wrażliwość na drgania o różnych częstotliwościach:

- niskie częstotliwości drgań do ~0.5 Hz mogą wywoływać u człowieka objawy choroby morskiej;
- drgania w zakresie około 1-2 Hz czyli takie które odpowiadają częstotliwości wstrząsów jakich doznaje człowiek podczas chodzenia są znoszone dobrze;
- w przypadku pasma w okolicach 4-8 Hz występuje istotne zmniejszenie ludzkiej sprawności działania.

3. SYSTEM ALGEBRY KOMPUTEROWEJ SAGEMATH – OPIS I MOŻLIWOŚCI (OPRACOWAŁ ŁUKASZ LECHNA)

3.1 WPROWADZENIE

SageMath jest rozbudowanym systemem algebry komputerowej dostępnym dla systemów operacyjnych Microsoft Windows i Linux z czego SAGE jest skrótem od: Software for Algebra and Geometry Experimentation. Oprogramowanie to zostało napisane w języku Python, łączy ono w sobie wiele darmowych pakietów oprogramowania matematycznego takich jak NumPy, matplotlib, SciPy i inne. Program jest stale rozwijany, oraz rozpowszechniany na warunkach licencji GPL (General Public Licence). Sagemath jest darmową alternatywą dla programów takich jak Wolfram Mathematica czy Mathworks Matlab. W tym rozdziale pracy omówiona będzie krótko historia powstania i rozwoju programu Sagemath, omówione zostaną też możliwości na jakie pozwala nam Sage i wyprowadzone przykłady dla zobrazowania działania programu.

3.2 HISTORIA I ROZWÓJ PROGRAMU

Początkowym inicjatorem i twórcą programu jest William A. Stein, uwczesny nauczyciel akademicki uniwersytetu Washington który, zapoczątkował projekt Sagemath. W roku 2005 światu ukazana została pierwsza wersja tego oprogramowania. Jego celem miało być stworzenie darmowego programu komputerowego do zastosowań matematycznych. Chęć stworzenia takiego oprogramowania wynikała wysokich cen zakupu licencji dla programów takich jak Matlab, Maple

czy Magma, których ceny zakupu podstawowej licencji oscylowały w granicach kilku tysięcy dolarów, w dodatku na rynku tego typu oprogramowania od lat panowała stagnacja i nie było żadnych alternatyw dla drogich programów komercyjnych. Niezadowolenie nie wynikało tylko i wyłącznie z kwestii finansowych, programy komercyjne zwykle nie określały sposobów w jakich przeprowadzane były obliczenia co oznaczało że inni matematycy nie byli w stanie przeanalizować kodu i sposobu w jaki obliczenia komputerowe zostały przeprowadzone dla uzyskania wyniku wyjściowego.

Organizowane były tzw. „Dni Sage" będące swego rodzaju warsztatami dla chętnych deweloperów i studentów organizowanymi przez Steina na których skupiano się nad rozwojem programu, implementacją nowych rozwiązań i rozwijaniem funkcjonalności, który w tamtym czasie(lata 2005-2007) wciąż był bardzo ubogi na wielu polach i znacznie odbiegał od programów takich jak wiodąca wtedy Magma czy Mathematica, chociażby dlatego że nie posiadał on interfejsu w środowisku notatnika (notebook interface), czyli notatnika komputacyjnego będącego wirtualnym środowiskiem używanym do programowania słownego. W roku 2007 Sage zyskał interfejs pozwalający na wykonywanie równań symbolicznych w programie bez potrzeby wykonywania skomplikowanych konwersji, dzięki czemu mógł być używany do pracy na zajęciach ze studentami. W tymże roku Sage został nominowany do nagrody Trophees du Libre gdzie zajął pierwsze miejsce w kategorii dla oprogramowania naukowego, zapewniając programowi zastrzyk popularności. Wiele osób zainteresowanych chęcią poprawienia pracy środowiska przyłączało się do projektu i z biegiem czasu nad programem pracowały setki developerów rozwijając go w różnych dziedzinach zastosowań, co uczyniło program tym czym jest dziś.

Sam twórca obecnie zrezygnował z pracy akademickiej, stał się współzałożycielem SageMath Inc. w 2016r. Firma stoi za stworzeniem i dalszym udoskonalaniem serwisu CoCalc (Collaborative Calculation and Data Science) będącego rozwinięciem idei Sage i pozwalającego na wykonywanie obliczeń i analizy danych, umożliwiając współpracę w czasie rzeczywistym. CoCalc pozwala na przechowywanie notatników Jupyter w chmurze, oraz pozwala na równoległą pracę w nich przez kilku użytkowników na raz co otwiera zupełnie nowe możliwości prowadzenia zajęć, czy pracy z plikami. Sagemath jest jednym z wielu zespolonych elementów oprogramowania które CoCalc obecnie w sobie łączy pozwalając na łatwą i płynną pracę z nimi w jednym zintegrowanym środowisku.

3.3 OPIS MOŻLIWOŚCI

Program Sagemath daje wiele możliwości, umożliwia on nie tylko wyliczanie równań, może on także rozkładać na czynniki ,lub upraszczać wyrażenia. Całkowanie, obliczanie granic i nie tylko. Sage może manipulować wyrażeniami zawierającymi pierwiastki, wykładniki i tym podobne. Jest przydatny w rozwiązywaniu i ilustrowaniu problemów między innymi z zakresu probabilistyki i kombinatoryki. Program przydatny jest nie tylko na polach użyteczności dla matematyki ale także jego obszerna funkcjonalność przydatna jest również w statystyce czy fizyce.

SageMath dzięki wbudowanym pakietom jest w stanie obrazować grafiki 2D i 3D dające, z pomocą narzędzia jesteśmy także w stanie wykonywać proste animacje, czy tworzyć modele i

wykresy 3D. Program jest bardzo pomocnym narzędziem w praktycznie każdej z dziedzin współczesnej matematyki w tym między innymi: rachunki symboliczne, działania na wielomianach, algebrze abstrakcyjnej, kryptografii itp. dającym użytkownikowi szeroki wachlarz możliwości, bez potrzeby zakupu drogiej licencji, czy subskrypcji. Ponadto jest prosty w obsłudze i stale rozwijany przez środowisko developerskie. Szczególną zaletą jest tutaj wielofunkcyjność Sage, gdyż pozwala używać jednego zunifikowanego programu i środowiska zamiast nauki kilku różnych środowisk programistycznych, czy też uciążliwego transferu danych pomiędzy kilkoma narzędziami. Obecnie Sagemath dostępny jest w wersji do pobrania na komputery z systemami Linux czy Windows ,oraz w wersji online Sagemath Cell pozwalającej na wykonywanie zapytań z użyciem przeglądarki bez potrzeby instalacji programu na komputerze jednak posiada on dość ograniczone możliwości i z powodu bycia witryną internetową nie pozwala na zapis wyników wykonanych zapytań do pliku.

3.4 ALGEBRA I ANALIZA DANYCH

SageMath pozwala nam na szeroki zakres obliczeń analitycznych na wyrażeniach symbolicznych składających się z liczb, podstawowych operacji matematycznych(dodawanie, odejmowanie, mnożenie, dzielenie), zmiennych symbolicznych, oraz funkcji takich jak **sin**, **cos**, **sqrt**, **log** itp. Dla przykładu rozważmy następujące równanie, z niewiadomą X i parametrem α:

$$x^2 - \frac{3x}{cos\alpha} + \frac{2}{cos\,\alpha^2} - 5 = 0, gdzie\ \alpha \in \left[\frac{-\pi}{2}, \frac{\pi}{2}\right] \tag{3.1}$$

Kod zapisany w Sage:

```
sage: x,alpha = var('x, alpha')
sage: eq= x**2 - 3/cos(alpha)*x +2/cos(alpha)**2 -5 ==0;
sage: eq

out: x^2 - 3*x/cos(alpha) + 2/cos(alpha)^2 - 5 == 0

sage: solve(eq,x)

out: [x == -1/2*(sqrt(20*cos(alpha)^2 + 1) - 3)/cos(alpha),
x == 1/2*(sqrt(20*cos(alpha)^2 + 1) + 3)/cos(alpha)]
```

Powyżej zaprezentowane postaci wyników dla wielu osób będą niesatysfakcjonujące gdyż przez zaprezentowanie go w sposób ciągu znaków jest on trudny do odczytania i wymaga czasu na jego poprawną interpretację. Dla uzyskiwania bardziej klarownych do odczytania wyników, czy sprawdzenia poprawności zapisanego przez nas równania w zapisie matematycznym przydatna jest metoda **show** której użyjemy jej na przykładzie równania powyżej, oraz wygenerowanych wyników.

Kod zapisany w Sage:

```
sage: show(eq)
```

$$x^2 - \frac{3\,x}{\cos(\alpha)} + \frac{2}{\cos(\alpha)^2} - 5 = 0$$ out:

```
sage: show(solve(eq,x))
```

$$\left[x = -\frac{\sqrt{20\,\cos(\alpha)^2 + 1} - 3}{2\,\cos(\alpha)}, x = \frac{\sqrt{20\,\cos(\alpha)^2 + 1} + 3}{2\,\cos(\alpha)}\right]$$ out:

Tabela 3.1 Tabela przydatnych funkcji obliczeniowych programu SageMath

Wybrane solvery do rozwiązywania równań	
Rozwiązanie równania	**solve**
Znajdź pierwiastki wielomianu	**roots**
Pierwiastek numeryczny	**find_root**
Solvery dla równań wektorowych i funkcyjnych	
Rozwiąż równania liniowe	**solve_right, solve_left**
Rozwiąż równania różniczkowe	**desolve**
Rozwiąż rekurencje	**rsolve**

Sage pozwala także na używanie różnych przydatnych funkcji, użytecznych w analizie takich jak obliczanie sum, granic, równań nieliniowych czy różniczkowych (szerzej opisane w rozdziale 4 pracy). Przykładowo funkcja **sum** pozwala nam na obliczanie sum symbolicznych, poniżej zaprezentowano przykład obliczenia sumy **n** pierwszych dodatnich liczb całkowitych.

Kod zapisany w Sage:

```
sage: x, n = var('x, n')
sage: sum(x, x, 1, n).factor()

out: 1/2*(n+1)*n
```

Program pozwala nam także na łatwe obliczanie wyżej wspomnianych granic funkcji, do tego możemy użyć funkcji **limit,** lub krótszej **lim** która jest jej aliasem.

Kod zapisany w Sage:

```
sage: limit((x**(1/2)-5)/ ((x+5)**(1/3) -2), x = 8)

out: (2*sqrt(2) - 5)/(13^(1/3) - 2)
```

```
sage: f(x) = (cos(5*x)/cos(3*x))
sage: lim(f(x), x = pi/2)

out: -5/3
```

Tabela 3.2 Zestawienie funkcji i operatorów różniczkowych w programie SageMath

Wybrane funkcje i operatory przydatne w analizie matematycznej	
Pochodna	**diff(f(x), x)**
Pochodna n-tego rzędu	**diff(f(x), x, n)**
Funkcja pierwotna	**Integrate(f(x), x)**
Integracja numeryczna	**Integrate_numerical(f(x), a, b)**
Suma symboliczna	**sum(f(i), i, imin, imax)**
Granica	**limit(f(x), x=a)**
Rozwinięcie Taylora	**taylor(f(x), x, a, n)**
Rozwinięcie szeregu potęgowego	**f.series(x==a, n)**
Graf funkcji	**plot(f(x), x, a, b)**

Oprogramowanie umożliwia nam tworzenie macierzy i wykonywanie działań na zeń macierzach, z wykorzystaniem stworzonych do tego operatorów. Do stworzenia macierzy wykorzystuje się funkcję **matrix**, lub **MatrixSpace** do utworzenia przestrzeni macierzowej by wprowadzić do niej obiekt by ją wypełnić. Macierze pozwalają na działania z użyciem wektorów wierszowych wektory tworzymy z użyciem polecenia **vector**, lub tworząc przestrzeń wektorową z użyciem **VectorSpace** i wprowadzając do niej obiekt. Sage nie pozwala niestety obecnie na działania z wykorzystaniem wektorów kolumnowych gdyż taka klasa wektorów nie istnieje ale według oficjalnej dokumentacji Sage, dodanie takiej funkcji jest planowane w przyszłości. Poniżej zaprezentowano utworzenie przykładowej macierzy oraz wykonano dwie jednakowe macierze z użyciem wyżej wspomnianych funkcji **matrix** i **MatrixSpace**.

Kod zapisany w Sage:

```
sage:A = matrix(QQ, [[1,2],[3,4]]);
sage:A

out: [1 2]
     [3 4]

#utworzenie pustej przestrzeni macierzowej 2x2

sage:MS=MatrixSpace(QQ,2)

#wypełniamy przestrzeń macierzy listą liczb i zapisujemy utworzoną
```

```
macierz w zmiennej B

sage:L = [1,2,3,4]
sage:B=MS.matrix(L);
      sage: B

      out: [1 2]
           [3 4]
```

Tabela 3.3 Zestawienie funkcji i operatorów wykorzystywanych do operacji na macierzach

Wybrane funkcje i operatory używane do działań na macierzach	
Tworzenie macierzy	**matrix**
Tworzenie przestrzeni macierzowej	**MatrixSpace**
Rozwiązanie równania macierzowego	**solve_right, solve_left**
Generowana kolumnowo przestrzeń wektorowa	**column_space**
Generowana wierszowo przestrzeń wektorowa	**row_space**
Łączenie macierzy	**block_matrix**
Wartości własne macierzy	**eigenvalues**
Redukcja do postaci Jordana	**jordan_form**

3.5 PROGRAMOWANIE I STRUKTURY DANYCH

3.5.1 WPROWADZENIE

System algebry komputerowej Sage jest w rzeczywistości rozszerzeniem języka komputerowego Python, i pozwala na dowolne używanie możliwości języka programistycznego Python, z kilkoma odstępstwami o których wspomnę poniżej w tej części rozdziału. Komendy użyte w powszednich częściach rozdziału pokazują że biegła znajomość Pythona nie jest wymagana aby używać Sage, jednak konstrukty programistyczne języka Python potrafią być bardzo przydatne i bardzo poszerzają możliwości użytkowe programu na przykład jeśli chcemy wykonywać zapytania zawierające rozbudowane sekwencje instrukcji. W tej części rozdziału pobieżnie omówione zostaną najprzydatniejsze elementy składni języka Python i użyteczne dla nas struktury danych których pozwala nam używać.

Ważnym paradygmatem programowania strukturalnego jest to by napisany program był skończoną sekwencją instrukcji wykonanych w określonej kolejności. Zapisane instrukcje w programie możemy podzielić na proste i złożone:

- instrukcje proste – to instrukcje których rezultatem jest przypisanie wartości jakiejś zmiennej lub rezultat wyjściowy
- instrukcje złożone – to polecenia takie jak pętle, czy instrukcje warunkowe zawierające w sobie kilka instrukcji lub ich wyników.

3.5.2 FUNKCJE

Sage pozwala nam na wykonywanie zapytań funkcji oraz na tworzenie nowych. Funkcje pozwalają nam na między innymi na wykonywanie obliczeń na podanych w zapytaniu argumentach. Definiowanie funkcji pozwala nam na znaczne skrócenie kodu szczególnie przy wykonywaniu repetytywnych obliczeń lub akcji. Sage posiada także wiele funkcji przejętych z języka Python jak **print**, czy **cos**. Do kreowania nowych funkcji używamy komendy **def**, by zobrazować składnię definiowania funkcji wykonano poniższy przykład następującej funkcji:

$$(x, y) \to x^2 - y^2.$$

Kod zapisany w Sage:

```
sage: def foo(x,y):
return x**2 - y**2
sage: a = var('a'); foo(2*a**2,3*a)
out: 4*a^4 - 9*a^2
```

Powyżej zdefiniowana funkcja kończy się komendą **return**, z jej użyciem wyprowadzone argumenty są rezultatem wykonania funkcji. Domyślnie wszystkie zmienne zawarte w funkcji są uznawane za zmienne lokalne, tj. nie można się do nich odwołać czy użyć ich poza funkcją. Zmienne lokalne funkcji są kreowane przy każdym wywołaniu danej funkcji oraz przestają istnieć po jej zakończeniu, zmienne o tej samej nazwie zawarte w funkcji nie wchodzą w jakąkolwiek interakcję z wartościami globalnymi o tej samej nazwie wewnątrz funkcji, chyba że się do niej wcześniej odwołaliśmy z użyciem słowa kluczowego **global**. Wspomniane zachowanie prezentują poniższe dwa przykłady.

Kod zapisany w Sage:

```
sage: def foo(x):
y = x/2
return x+y

sage: x = 2; y = 5
sage: foo(4),x,y

out:(6, 2, 5)
```

```
sage:x = 2; y = 5;
sage: def foo2():
global x,y;
```

```
        y = x/2; x = 7;

        sage: foo2(); x,y

out: (7, 1)
```

3.5.3 PĘTLE WHILE I FOR

Jedną z bardziej uniwersalnych i przydatnych funkcji jakie przynosi ze sobą Python są pętle, często używane w matematyce komputerowej czy przy imporcie danych z zewnętrznych plików. Rozróżniamy pętle enumeracyjne **for,** oraz pętle **while**. Pętle for pozwalają nam na wykonanie określonych instrukcji dla każdej iteracji pętli dla ściśle określonego przedziału. Poniżej pokazano przykładowe użycie pętli **for**.

Kod zapisany w Sage:

```
        sage:for a in [1..5]:
        print(7*a)

out: 6
12
18
24
30
36
42
```

Pętle **while** natomiast różnią się tym od pętli **for** że nie podaje się w niej ściśle określonej ilości iteracji. Ilość iteracji pętli **while** zależy bowiem od spełnienia warunku zapisanego przy jej definicji, dopóki dany warunek pozostanie spełniony pętla będzie wykonywana, jak na przykładzie poniżej.

Kod zapisany w Sage:

```
        sage: k=0;x=0
        sage: while k<=10:
        x = x+k**2
        k = k+1

        sage: x

out: 385
```

3.5.4 INSTRUKCJE WARUNKOWE

Instrukcje warunkowe **if**, pozwalają nam na wykonywanie instrukcji w zależności od tego czy został spełniony sprecyzowany wcześniej warunek. Ze struktur instrukcji warunkowej możemy wyróżnić dwie podstawowe składnie:

if warunek : sekwencja instrukcji	**if** warunek : sekwencja instrukcji else: kolejna sekwencja instrukcji

Kod zapisany w Sage:

```
sage: k=0;x=0
sage: while k<=10:
if k%2==0:
x=x+2
else:
x=x+k

k = k+1

sage: x

out: 37
```

3.5.5 LISTY

Listy pozwalają na uporządkowane przechowywanie danych, każdy obiekt w liście posiada własny unikatowy numer zaczynając od 0, do którego w każdym momencie możemy się odwołać. Listę definiujemy na kilka sposobów: poprzez okraszenie jej elementów, oddzielonych przecinkami w nawiasach kwadratowych, lub poprzez utworzenie pustej listy i późniejsze dodawanie do niej nowych elementów. Elementy list możemy także poddać poddać edycji w przeciwieństwie do zestawów które są niezmienne, sprawiając że listy są jednymi z najbardziej użytecznych struktur danych dostępnych w Sage.

Kod zapisany w Sage:

```
sage:L = [1,2,3]
sage:L.append(4)

out: [1, 2, 3, 4]

sage:L[0]
```

```
out: 1
```

Odwoływanie się do ujemnych indeksów listy sprawia że otrzymamy wartości listy liczone od jej końca z czym L[-1] odnosi się do ostatniej wartości z listy.

Kod zapisany w Sage:

```
Sage:L = [1,2,3,4]
sage:L[-1]

out: 4
```

Tabela 3.4 Wybrane metody i operatory dla operacji na listach

Wybrane przydatne metody i operatory dla list	
Konkaktenacja	**+**
Konkaktenacja iteracyjna	*****
Zliczenie elementów listy	**count()**
Dodanie elementu do listy w podanym indeksie	**insert()**
Dodanie elementu do listy	**append()**
Dodanie grupy elementów	**extend()**
Odwrócenie kolejności elementów	**reverse()**
Usunięcie elementu o podanym indeksie	**pop()**
Usunięcie elementu o podanej wartości	**remove()**

3.5.6 KROTKI

Krotki [z ang. tuple] są używane do magazynowania wielu elementów wewnątrz jednej zmiennej. Krotka to uporządkowany zbiór elementów który różni się od list tym że po utworzeniu nie ma możliwości edycji, ani dodania do niego nowych elementów. Krotki definiujemy z pomocą konstruktora **tuple** albo z wykorzystaniem **()**.

Kod zapisany w Sage:

```
sage: tuple = (1,10,2,20);
sage: tuple, tuple[0]
```

```
out: ((1, 10, 2, 20), 1)
```

3.5.7 ZESTAWY

W przeciwieństwie do listy, gdzie wszystkie elementy są uporządkowane, zestaw danych[z ang. Set] jest strukturą nieuporządkowaną i śledzi tylko czy dany element już jest częścią danego zestawu, czy nie. Zestaw nie bierze przy tym uwagi na to w której pozycji zbioru znajduje się element ani na to ile razy się on w nim powtarza. Jesteśmy w stanie kreować zbiory elementów z użyciem funkcji **Set**.

Kod zapisany w Sage:

```
sage: E = Set([1, 2, 4, 8, 2, 2, 2]); F = Set([7, 5, 3, 1]); E, F

out: ({8, 1, 2, 4}, {1, 3, 5, 7})
```

Wykorzystując operator **in** jesteśmy w stanie sprawdzić czy dany zestaw zawiera element o który się pytamy. Sage pozwala także na sprawdzanie sum zbiorów z użyciem operatora **+** lub |, skrzyżowania z pomocą &, różnicy zestawów -, i różnicy symetrycznej wykorzystując **^^**.

Kod zapisany w Sage:

```
sage: E = Set([1, 2, 4, 8, 2, 2, 2]); F = Set([7, 5, 3, 1]);
sage:  5 in E, 5 in F, E + F == F | E

out: (False, True, True)

sage: E & F, E - F, E ^^ F

out: ({1}, {8, 2, 4}, {2, 3, 4, 5, 7, 8})
```

Należy pamiętać że w przeciwieństwie do list zestawy danych nie pozwalają na edycję elementów ani samego zbioru. Możliwe jest zatem utworzenie zestawu którego elementami są krotki czy inne zestawy, jednak nie możliwym jest utworzenie zestawu którego elementami będą listy.

3.5.8 SŁOWNIKI

Python a zatem i Sage pozwala nam także na tworzenie tak zwanych słowników[z ang. dictionary]. Słowniki pozwalają na powiązanie ze sobą wartości z przydzielonymi kluczami umożliwiając ich wywołanie, podobnie jak w książce telefonicznej, czy słowniku językowym. Kluczami słownika mogą być stałe zmienne każdego typu: ciągi znaków, liczby, krotki itp. Składnia jest podobna do tej używanej w przypadku list, do stworzenia słownika używamy **dict()** lub **{}**.

Kod zapisany w Sage:

```
sage: D={}; D['jeden']=1; D['dwa']=2; D['trzy']=3;
sage:  D['jeden'] + D['trzy']
```

```
out: 4
```

3.6 TWORZENIE GRAFIKI

3.6.1 WPROWADZENIE

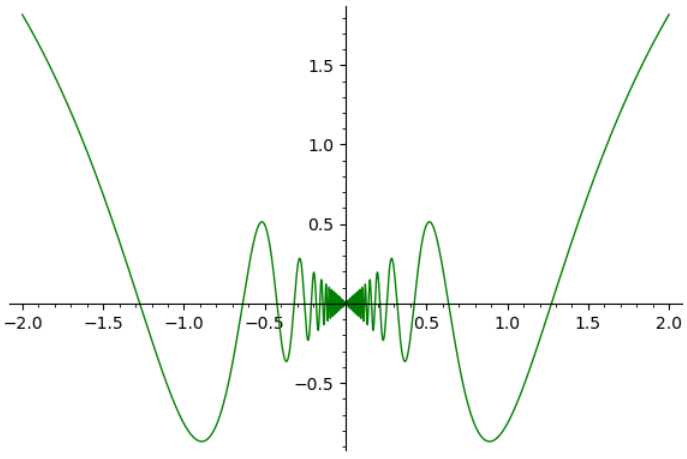
Rysowanie funkcji jednej lub kilku zmiennych z użyciem serii danych znacznie ułatwia zrozumienie różnych zjawisk fizycznych czy działań matematycznych. Wykreowane grafiki potrafią być nieocenione w analizie matematycznej i pomagają wyciągać wnioski. W tej części rozdziału zostanią krótko zilustrowane możliwości jakie daje nam Sage w kreowaniu grafiki z wykorzystaniem przykładów wykonanych w programie.

3.6.2 GRAFICZNA REPREZENTACJA FUNKCJI

Do wykonywania grafu funkcji w przedziale [a,b] używamy funkcji **plot** jej podstawowa struktura wygląda następująco: plot(f(x), a, b). Funkcja pozwala nam na definiowanie wielu opcji generowanych wykresów takich jak kolor, czy minimalna ilość obliczonych punktów wykresu. Poniżej przedstawiono przykładowy wykres funkcji wygenerowany z użyciem **plot**.

Kod zapisany w Sage:

```
sage: plot(x * sin(4/x), x, -2, 2,color="green", plot_points=600 )
```



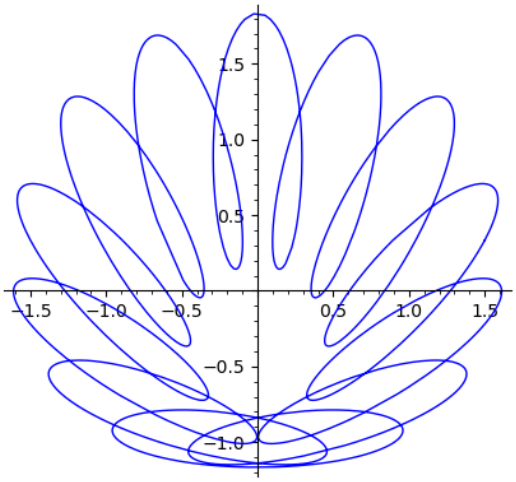
Rysunek 3.1Wykres funkcji x*sin(4/x).

Krzywe parametryczne jesteśmy w stanie wizualizować z użyciem komendy **parametric_plot(f(t), g(t), (t,a,b))**, z czego a i b są interwałem w którym mieści się parametr t. Poniżej przedstawiona została krzywa paramtetryczna zdefiniowana przez układ równań:

$$\begin{cases} x(t) = cos(t) + \frac{1}{2}cos(13t) + \frac{1}{3}sin(12t), \\ y(t) = sin(t) + \frac{1}{2}sin(13t) + \frac{1}{3}cos(12t). \end{cases} \tag{3.2}$$

Kod zapisany w Sage:

```
sage: t = var('t')
sage: x = cos(t) + cos(13*t)/2 + sin(12*t)/3
sage: y = sin(t) + sin(13*t)/2 + cos(12*t)/3
sage: f = parametric_plot((x, y), (t, 0, 2*pi))
sage:f.show(aspect_ratio=1)
```



Rysunek 3.2 Wykres krzywych parametrycznych.

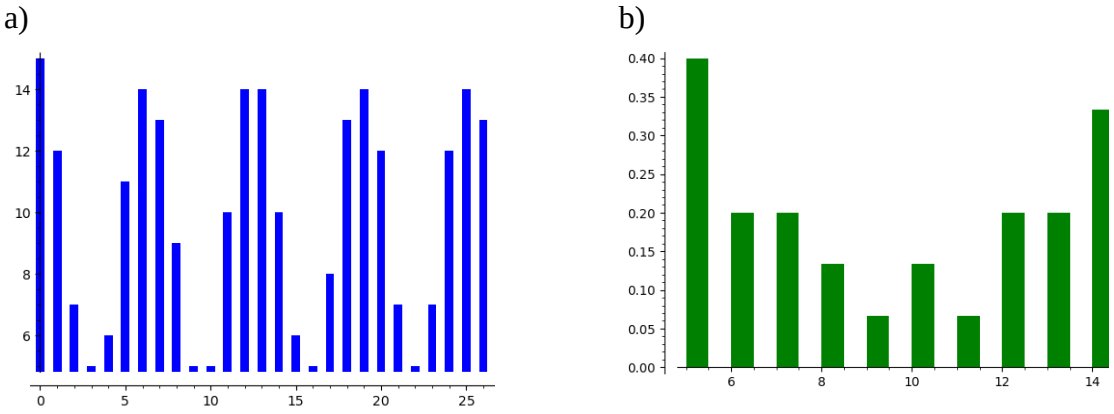
3.6.3 GRAFICZNA REPREZENTACJA DANYCH

Sage umożliwia nam nie tylko tworzenie wykresów liniowych ale także na generowanie wykresów słupkowych, używanych często do analizy danych statystycznych, czy serii pomiarów. By stworzyć wykres słupkowy używamy komendy **bar_chart**, która przyjmuje listę danych wejściowych i rysuje słupki których to wysokość zależy od wartości elementów listy. Poza wykresami słupkowymi SageMath umożliwia także tworzenie histogramów, wysokość każdego słupka histogramu jest zależna od liczby wartości zawartych w liście. Do tego używamy funkcji **plot_histogram** wartości listy zostają pogrupowane na przedziały, których liczba jest regulowana przez opcję **bins** i domyślnie wynosi 50.

Kod zapisany w Sage:

```
sage:list = [10 + floor(5*cos(i)) for i in range(30)]
sage:  bar_chart(list)
```

```
sage: finance.TimeSeries(list).plot_histogram(bins=20,
color="green")
```

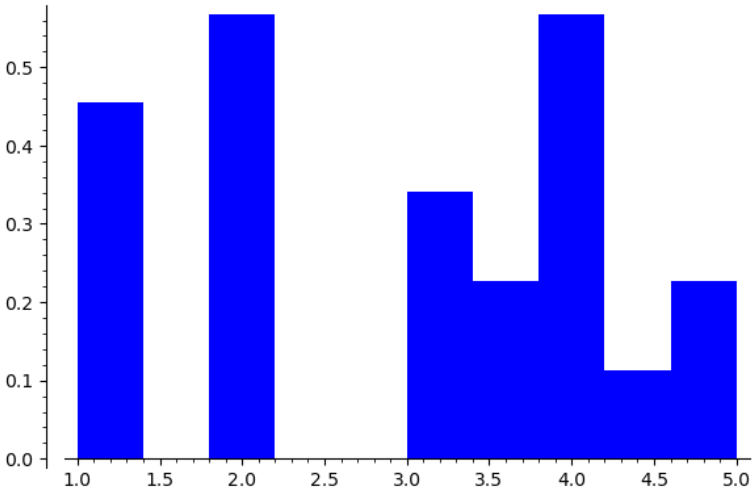


Rysunek 3.3 a) Przykład wykresu słupkowego b) przykład histogramu.

Do analizy danych statystycznych bardzo często używa się danych pochodzących z zewnętrznych źródeł. Bardzo często dane odczytujemy z plików w formacie csv na których to import pozwala nam język Python, w którym to języku istnieją różne pakiety pozwalające na łatwy odczyt danych z plików, takie jak Pandas czy csv. Dla przykładu, poniżej przedstawiony zostanie histogram ocen dla fikcyjnej klasy 2b, który zapisany został w pliku csv. Tabela posiada 3 kolumny: imię, nazwisko, ocena. By pozyskać tylko oceny użyto napisanego poniżej kodu, z wykorzystaniem pętli for, oraz składni try/except by uniknąć dodania do listy tekstu znajdującego się w pierwszym wierszu tabeli.

Kod zapisany w Sage:

```
Sage: import csv
sage: dane = csv.reader(open(„oceny_2b.csv”))
sage: oceny = []; lista_ocen = []
sage: for wiersz p dane:
oceny.append(wiersz[2])
for i p oceny:
try:
f = float(i)
except ValueError:
pass
else:
lista_ocen.append(f)
sage:finance.TimeSeries(lista_ocen).plot_histogram(bins=10)
```



Rysunek 3.4 Histogram ocen pobranych z zewnętrznego pliku

3.6.4 GRAFICZNA REPREZENTACJA ROZWIĄZAŃ DLA RÓWNAŃ RÓŻNICZKOWYCH

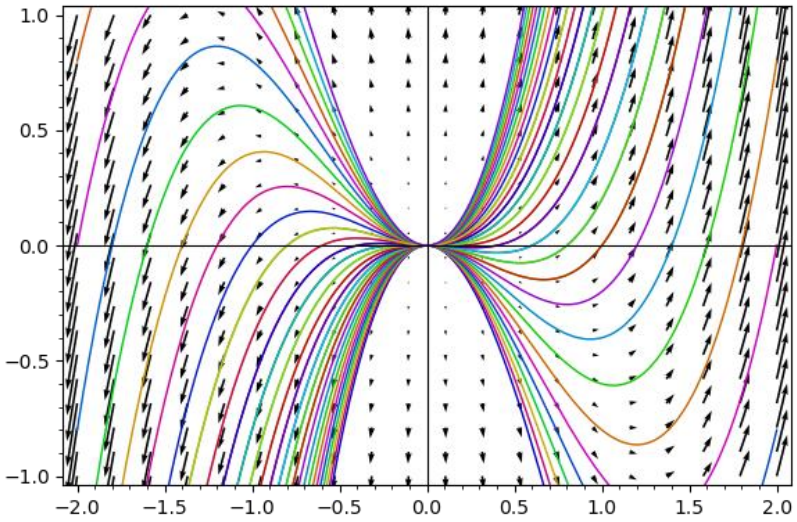
Dzięki wykorzystaniu odpowiednich kombinacji komend, z użyciem Sage jesteśmy w stanie obrazować wyniki dla równań różniczkowych zwyczajnych lub ich układów. Do obliczania równań różniczkowych zwyczajnych używamy komendy desolve. Jeżeli chcemy uzyskać wyniki numeryczne, mamy do wyboru kilka narzędzi to umożliwiających: odeint, desolve_rk4 i ode_solver. Dzięki temu że funkcje odeint oraz desolve_rk4 zwracają wynik jako listę punktów możemy w łatwy sposób rysować wykresy z użyciem funkcji line. Poniższy przykład przedstawia wyrysowanie krzywych całkowania równania różniczkowego z użyciem komendy desolve: $xy' - 2y = x^3$ przykłady pochodzą z literatury **[5]**.

Kod zapisany w Sage:

```
sage: x = var('x'); y = function('y')
sage: DE = x*diff(y(x), x) == 2*y(x) + x^3
sage: desolve(DE, [y(x),x])

out: (_C + x)*x^2

sage: sol = []
sage: for i in srange(-2, 2, 0.2):
sol.append(desolve(DE, [y(x), x], ics=[1, i]))
sol.append(desolve(DE, [y(x), x], ics=[-1, i]))
sage: g = plot(sol, x, -2, 2)
sage: y = var('y')
sage: g += plot_vector_field((x, 2*y+x^3), (x,-2,2), (y,-1,1))
sage: g.show(ymin=-1, ymax=1)
```



Rysunek 3.5Linie całkowania równania różniczkowego z wykorzystaniem funkcji desolve

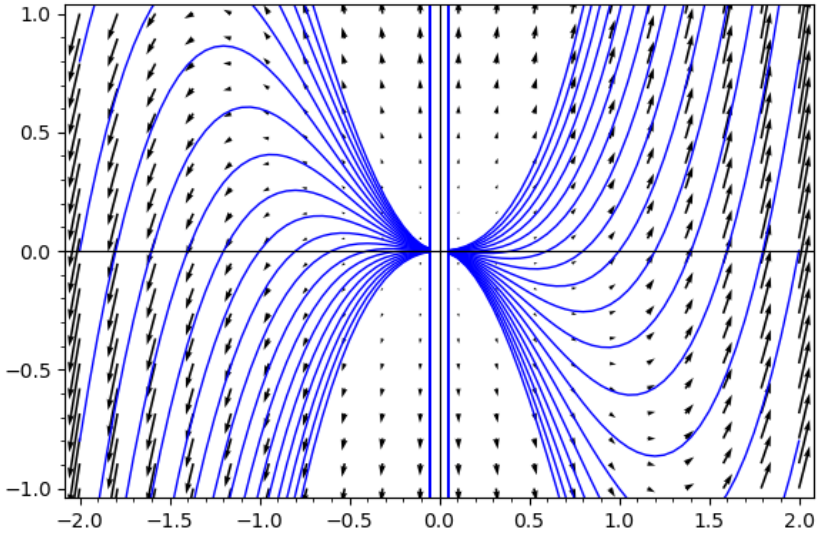
Aby skrócić czas obliczeń, zamiast obliczać równanie różniczkowe kilka razy dla różnych warunków początkowych wykorzystując desolve, możemy użyć desolve_rk4 by uzyskać wyniki numerycznie.

Kod zapisany w Sage:

```
sage: x = var('x'); y = function('y')
sage: DE = x*diff(y(x), x) == 2*y(x) + x^3
sage: desolve(DE, [y(x),x])

out: (_C + x)*x^2

sage: sol = []
sage: for I in srange(-2, 2, 0.2):
sol.append(desolve(DE, [y(x), x], ics=[1, i]))
sol.append(desolve(DE, [y(x), x], ics=[-1, i]))
sage: g = plot(sol, x, -2, 2)
sage: y = var('y')
sage: g += plot_vector_field((x, 2*y+x^3), (x,-2,2), (y,-1,1))
sage: g.show(ymin=-1, ymax=1)
```



Rysunek 3.6 Linie całkowania równania różniczkowego z wykorzystaniem funkcji desolve_rk4

Jak można było zauważyć przy wyprowadzeniu, funkcja desolve_rk4 jako argument wejściowy pobiera równanie różniczkowe, nazwę niewiadomej, warunki początkowe, krok i granice przedziału w którym szukamy rozwiązań. Z wykresów na rysunkach 3.5 i 3.6 widać że wyniki wykonane numerycznie różnią się nieco od tych teoretycznych czego można się było spodziewać, bardzo rzadko bowiem używając na jednym równaniu obu metod otrzymamy nieodróżnialne wyniki.

Kod zapisany w Sage:

```
sage: import scipy; from scipy import integrate
sage: f = lambda y, t: - cos(y * t)
sage: t = srange(0, 5, 0.1); p = Graphics()
sage: for k in srange(0, 10, 0.15):
y = integrate.odeint(f, k, t)
p += line(zip(t, flatten(y)))
sage: t = srange(0, -5, -0.1); q = Graphics()
sage: for k in srange(0, 10, 0.15):
y = integrate.odeint(f, k, t)
q += line(zip(t, flatten(y)))
sage: y = var('y')
sage: v = plot_vector_field((1, -cos(x*y)), (x,-5,5), (y,-2,11))
sage: g = p + q + v; g.show()
```

Tabela 3.5 Wybrane funkcje do tworzenia grafik 2D

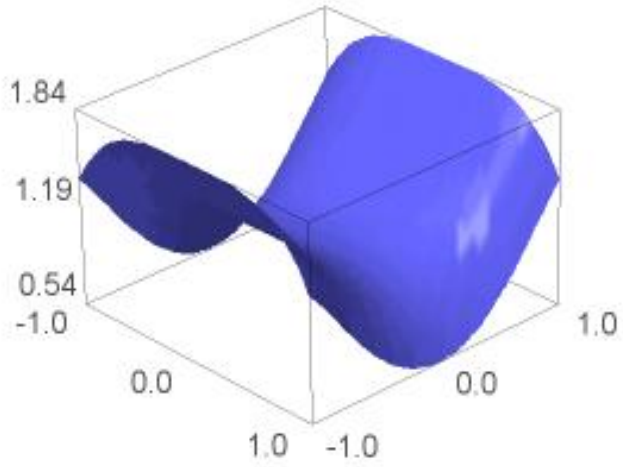
Wybrane funkcje używane do tworzenia wykresów	
Graf funkcji	**plot**
Pusty obiekt graficzny	**Graphics**
Krzywa parametryczna	**parametric_plot**
Krzywa zdefiniowana przez równanie biegunowe	**polar_plot**
Krzywa zdefiniowana przez równanie niejawne	**implicit_plot**
Zestawienie poziomów funkcji złożonej	**complex_plot**
Generowanie linii całkowych równań różniczkowych	**desolve_rk4, odeint**
Histogram danych statystycznych	**plot_histogram**
Graf słupkowy	**bar_chart**
Linia łamana	**line**

3.6.5 TWORZENIE GRAFIKI 3D

SageMath pozwala nam na kreowanie grafów trójwymiarowych dzięki funkcji **plot3d(f(x,y),(x,a,b),(y,c,d))**. Grafiki trójwymiarowe często są bardzo pomocne w analizie statystycznej niektórych zjawisk, przydatne są także np. przy badaniu funkcji wielu zmiennych. Na rysunku 3.7 widoczna jest przykładowa wykreowana z użyciem funkcji **plot3d** powierzchnia parametryczna.

Kod zapisany w Sage:

```
sage: x, y = var('x, y')
sage: h = lambda x,y: cos(x^2) + sin(y^2)
sage: plot3d(h, (x,-1,1), (y,-1,1), aspect_ratio=[1,1,1])
```

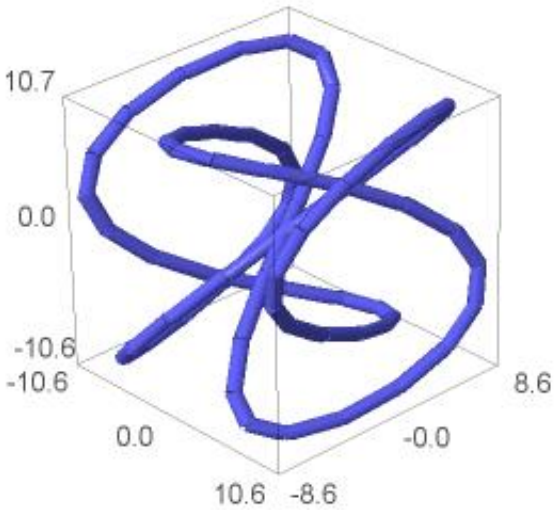


Rysunek 3.7 Powierzchnia parametryczna wykreowana z użyciem plot3d

Poniżej przedstawiono utworzenie przykładowej trójwymiarowej krzywej wykreowanej z użyciem komendy **line3d**, efekt wykonanej funkcji w Sage można zobaczyć na rysunku 3.8.

Kod zapisany w Sage:

```
sage: line3d([(10*sin(2*t), 8*sin(5*t), 10*sin(3*t)),\
for t in srange(0,6.4,.1)],radius=.5)
```

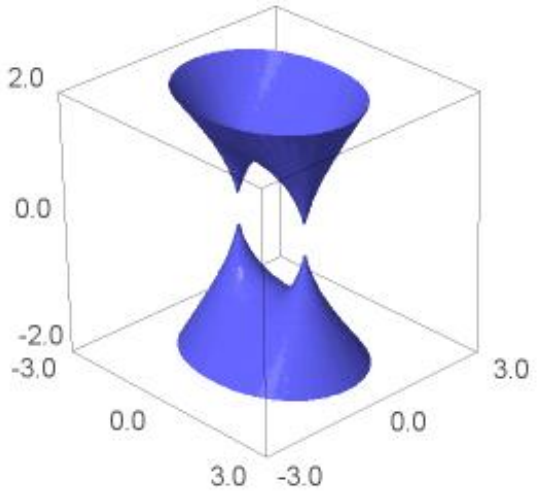


Rysunek 3.8 Trójwymiarowy węzeł wykreowany z użyciem line3d

Komenda **implicit_plot3d** pozwala nam na wyświetlanie powierzchni zdefiniowanych przez równania niejawne o formie f(x,y,z) = 0. Na rysunku 3.9 Przedstawiona została powierzchnia Cassiniego zdefiniowana równaniem $(a|2 + x^2 + y^2)^2 = 4a^2x^2 + z^4$ przykład pochodzi z literatury [5].

Kod zapisany w Sage:

```
sage: x, y, z = var('x, y, z'); a = 1
sage: h = lambda x, y, z:(a^2 + x^2 + y^2)^2 - 4*a^2*x^2-z^4
sage: implicit_plot_3d=implicit_plot3d(h, (x,-3,3), (y,-3,3),\
(z,2,2),plot_points=100)
```



Rysunek 3.9 Powierzchnia Cassiniego stworzona poleceniem implicit_plot3d

4. ROZWIĄZYWANIE RÓWNAŃ RÓŻNICZKOWYCH W SYSTEMIE SAGEMATH (OPRACOWAŁ MICHAŁ WIELGAT)

4.1 WPROWADZENIE

Program SageMath posiada wiele różnych funkcji służących do rozwiązywania równań różniczkowych. W naszym przypadku szukamy funkcji, która po otrzymaniu systemu równań różniczkowych zwyczajnych pierwszego stopnia rozwiąże je numerycznie, a wyniki poda w postaci listy punktów.

4.2 WYBRANE FUNKCJE PROGRAMU SAGEMATH

W programnie SageMath do numerycznego rozwiązywania równań różniczkowych zwyczajnych możemy użyć funkcje: **desolve_system_rk4, desolve_odeint.**

Funkcja: **desolve_system_rk4** – funkcja używająca algorytmu Rungego-Kutty, metody 4. rzędu. Jako wejścia wymaga ona prawe strony równań, zmienne zależne, oraz wartości początkowe. Dodatkowo należy podać zakres lub punkt zakończenia całkowania oraz długość użytego kroku. Działa w oparciu o system algebry komputerowej Maxima [11].

Funkcja: **desolve_odeint** – bazuje na funkcji odeint z modułów całkujących biblioteki SciPy. Jako wejść wymaga prawe strony równań, listę wartości początkowych, listę z punktami czasu oraz zmienne zależne. Można też podać zmienną niezależną. Zakres czasu oraz długość kroku symulacji można ustalić tworząc odpowiednią listę punktów czasu [11].

W tej pracy użyto funkcję desolve_odeint. Jednym z powodów tego wyboru jest to, że oparta o Maximę funkcja desolve_system_rk4 jest ograniczona precyzją sprzętowych liczb zmiennoprzecinkowych. Tak więc zastosowanie funkcji desolve_system_rk4 mogłoby skutkować otrzymaniem niedokładnych wyników [5].

4.3 UŻYCIE FUNKCJI DESOLVE_ODEINT W PROGRAMIE

W tym przykładzie użyto model 4 segmentów, który został dokładniej opisany w rozdziale 6. Użyto uzyskany w tym rozdziale system 8 równań różniczkowych zwyczajnych pierwszego stopnia (6.6):

```
F1 = vu
F2 = vs
F3 = vc
F4 = vh
F5 = ( -kt*( xu-x(t))+ks*( xs-xu ) +cs*( vs-vu ) )/mu -g
F6 = ( -ks*( xs-xu ) -cs*( vs-vu ) +kc*( xc-xs ) +cc*( vc-vs ) )/ms -g
F7 = ( -kc*( xc-xs ) -cc*( vc-vs ) +kb*( xh-xc ) +cb*( vh-vc ) )/mc -g
F8 = ( -kb*( xh-xc ) -cb*( vh-vc ) )/mc -g
```

Podstawienie tych równań do funkcji desolve_odeint wymaga zawarcia ich w jednej zmiennej. W tym celu użyto funkcji vector(), dzięki której uzyskano wektor wierszowy zawierający prawe strony równań użytego układu. Wektor ten zawarto w zmiennej „eqs”:

```
eqs = vector([F1,F2,F3,F4,F5,F6,F7,F8])
```

Następnie utworzono listę zawierającą zmienne zależne. Powinny one być zapisane w odpowiedniej kolejności względem układu równań. Zapisano je jako listę w zmiennej programowej o nazwie „vrs”:

```
vrs = [xu, xs, xc, xh, vu, vs, vc, vh]
```

Wymagana też jest lista wartości początkowych. Funkcja desolve_odeint przypisze wszystkie wartości listy zmiennym zależnym o kolejności ustalonej w liscie „vrs”. W tutaj przedstawionym przypadku nie było potrzeby dbania o kolejność, ponieważ wszystkim zmiennym przypisano wartość początkową 0:

```
ics = [0,0,0,0,0,0,0,0]
```

Ostatnią wymaganą lista jest sekwencja punktów czasowych użytych w symulacji. Zawiera ona wartości zmiennej „t”, zaczynając od 0, z każdą kolejną wartością rosnąc o interwał symulacji i kończąc się na ustalonym czasie końca badania. Do otrzymania takiej listy przydaje się funkcja srange. W ten sposób można generować sekwencję punktów czasowych podając początek i koniec przedziału oraz interwał kroków symulacji. Na potrzeby tworzonego programu górny przedział symulacji zapisano w zmiennej „tend”:

```
tend = 40
times = srange(0, tend, 0.01)
```

Oprócz przygotowania wszystkich wymaganych przez funkcję desolve_odeint wejść należy także zdefiniować funkcję nierówności drogi. Opisuje ona przemieszczenie pionowe działające w danym czasie na koło samochodu. Jako prosty przykład utworzono funkcję sinusoidalną x(t) o amplitudzie i częstotliwością modyfikowaną zmiennymi „b” i „om”:

```
x(t) = b*sin(om*t)
```

Parametry symulacji, to jest masy elementów oraz współczynniki sprężyn i amortyzatorów, przypisano z tabeli (NUMER), która to znajduje się na początku rozdziału 8. Dla zmiennych funkcji drogi przyjęto wartości *b=0.1* oraz *om=2*.

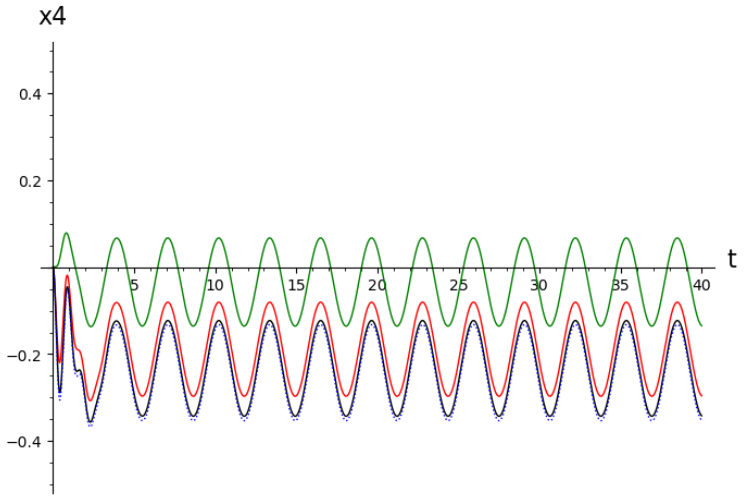
Wyniki funkcji desolve_odeint przypisano do zmiennej sol, za zmienną niezależną przyjęto „t”, a wszystkie wymagane listy dodano w następujący sposób:

```
sol = desolve_odeint( eqs, ics, times, vrs, ivar=t )
```

Po uruchomieniu w zmiennej sol ukazuje się lista z uzyskanymi przemieszczeniami dla każdej zmiennej zależnej. Aby ułatwić odczytanie i weryfikację poprawności tych danych zastosowano funkcję plot. W przypadku pierwszej listy z wartościami przemieszczenia, to jest listy dla zmiennej xu, wykres uzyskano w następujący sposób:

```
x1_sol = sol[:,0]
P1 = list_plot(list(zip(times, x1_sol)),plotjoined=True, color='green', figsize=7, axes_labels=['t','x1'])
```

Podobne wykresy utworzono dla zmiennych xs, xc i xh. Wynikiem połączenia tych czterech wykresów jest rysunek 4.1:



Rysunek 4.1 Przykładowe wyniki funkcji desolve_odeint

Wszystkie symulowane segmenty modelu wytworzyły poprawne krzywe ruchu. Wygenerowany obraz wymaga jedynie zmiany podpisów osi do uzyskania użytecznego wykresu. Funkcja desolve_odeint zadziałała poprawnie.

5. Model matematyczny układu zawieszenia w pojeździe (opracował Łukasz Lechna)

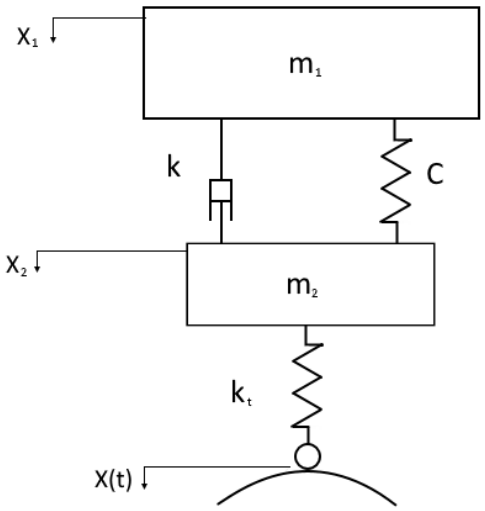
5.1 Wprowadzenie

Model matematyczny jest opisem układu w języku matematycznym. Do opisu zależności występujących w opisywanym układzie posłużymy się układem równań różniczkowych. Model matematyczny posłuży tutaj do opisu oraz analizy własności układu zawieszenia oraz stanie się podstawą dla uzyskania późniejszych wyników w programie Sagemath.

5.2 Model graficzny oraz matematyczny modelu

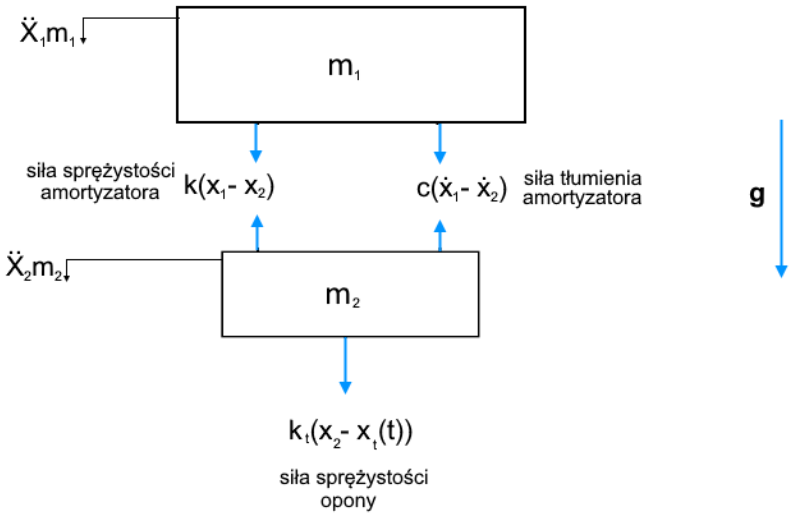
Poniżej znajduje się graficzne przedstawienie zawieszenia ćwiartki pojazdu samochodowego(Rys. 5.1).Opisywany w tym rozdziale model został zaczerpnięty z literatury **[2].** Składa się on z masy resorowanej(pojazd) m_1 oraz nieresorowanej(koło, opona) m_2. Ruch mas został opisany z użyciem współrzędnych X_1 i X_2. Między masami znajdują się elementy opisujące właściwości zawieszenia pojazdu: amortyzator samochodowy o współczynniku tłumienia C oraz

sprężyna o sztywności K. Pomiędzy masą nieresorowaną a zmienną opisującą nierówność drogi $X_t(t)$ w badanym modelu widoczna jest sprężyna o sztywności K_t opisująca sztywność opony auta.



Rysunek 5.1 Model zawieszenia ćwiartki pojazdu samochodowego.

Przyjmując założenia i uwzględniając prawa dynamiki Newtona siły oddziaływujące na układ mechaniczny przedstawiony na rysunku powyżej możemy przedstawić w następujący sposób w postaci graficznej zaprezentowanej na Rys 5.2.



Rysunek 5.2 Model zawieszenia ćwiartki pojazdu z rozrysowaniem sił działających w układzie.

Na rysunku 5.2 przyjęto następujące oznaczenia:

X_1, X_2 - to przemieszczenia mas,

$\dot{X}_1, \dot{X}_2$ – pochodne pierwszego rzędu współrzędnych mas odpowiadają prędkościom ich przemieszczeń, oraz

$\ddot{X}_1, \ddot{X}_2$ - pochodne drugiego rzędu współrzędnych mas odpowiadają ich przyspieszeniom,

g – siła grawitacji.

Opierając się na założeniach drugiego prawa dynamiki Newtona otrzymujemy równanie ruchu dla przemieszczenia X_1:

$$-C(\dot{X}_1 - \dot{X}_2) - K(X_1 - X_2) - m_1 g = m_1 \ddot{X}_1. \tag{5.1}$$

Natomiast równanie ruchu dla przemieszczenia X_2 wygląda następująco:

$$K(X_1 - X_2) + C(\dot{X}_1 - \dot{X}_2) - K_t\big(X_2 - X_t(t)\big) - m_2 g = m_2 \ddot{X}_2. \tag{5.2}$$

Ponadto przyjęto jako funkcję zmian kształtu drogi (jej wyboistość) w czasie funkcję:

$$X_t(t) = A \cdot sin(\omega \cdot t). \tag{5.3}$$

Po uporządkowaniu równań (5.1) i (5.2) otrzymujemy poszukiwany model matematyczny układu mechanicznego w postaci układu dwóch równań różniczkowych niejednorodnych liniowych drugiego rzędu:

$$\begin{cases} \frac{d^2X_1}{dt^2} = \ddot{X_1} = \frac{-C(\dot{X_1}-\dot{X_2})-K(X_1-X_2)}{m_1} - g \\ \frac{d^2X_2}{dt^2} = \ddot{X_2} = \frac{C(\dot{X_1}-\dot{X_2})+K(X_1-X_2)-K_t(X_2-X_t(t))}{m_2} - g \end{cases}. \tag{5.4}$$

Każde równanie należące powyższego układu równań w rozpatrywanym przypadku możemy przedstawić w postaci układu dwóch równań różniczkowych liniowych pierwszego rzędu. W tym celu posłużymy się podstawieniami:

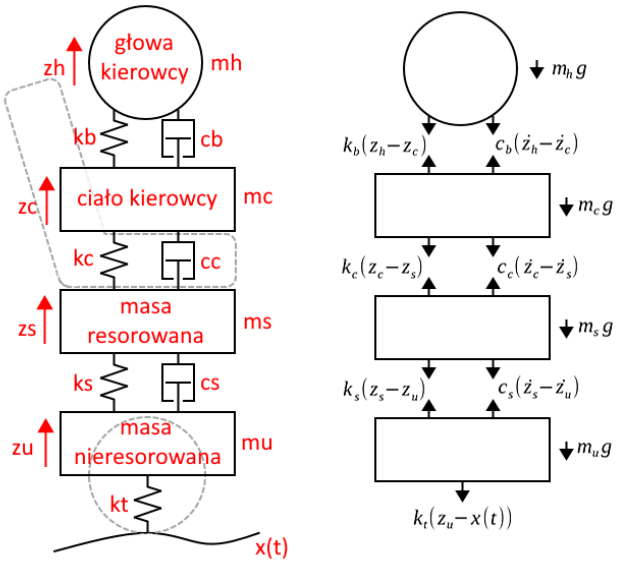
$$\begin{cases} \dot{X_1} = X_3 \\ \dot{X_2} = X_4 \end{cases} \tag{5.5}$$

Układ dwóch równań drugiego rzędu przekształcono do układu czterech równań pierwszego rzędu. Poniżej przedstawiono układ czterech równań liniowych różniczkowych pierwszego rzędu:

$$\begin{cases} \frac{dX_1}{dt} = \dot{X_1} = X_3 \\ \frac{dX_2}{dt} = \dot{X_2} = X_4 \\ \frac{dX_3}{dt} = \dot{X_3} = \frac{-C(X_3-X_4)-K(X_1-X_2)}{m_1} - g \\ \frac{dX_4}{dt} = \dot{X_4} = \frac{K(X_1-X_2)+C(X_3-X_4)-K_t(X_2-X_t(t))}{m_2} - g \end{cases} \tag{5.6}$$

6. Model matematyczny układu zawieszenia w pojeździe z pasażerem (opracował Michał Wielgat)

Symulacja oddziaływania zawieszenia na pasażera wymaga wzbogacenia modelu matematycznego o dwie nowe masy. Pierwsza z nich to tułowie, wspierane przez siedzenie o określonych parametrach sprężystości i tłumienia. Druga masa to głowa, której ruch też został opisany używając prostego modelu sprężyny i amortyzatora. Model ten został zapożyczony z literatury [9].



Rysunek 6.1 Złożony z 4 segmentów model ćwiartkowy układu zawieszenia z kierowcą.

Ze względu na zwiększoną liczbę elementów modelu zastosowano zmienione nazwy zmiennych. Indeksy dolne zostały zmienione z liczb na litery związane z konkretną masą (**r**oad, **u**nsprung mass, **s**prung mass, **c**hair, **h**ead).

Podobnie jak w przypadku modelu z rozdziału piątego, oddziaływania między elementami zostały zapisane w postaci wzorów opisujących zależność wywieranych sił od zmian położenia. Uwzględniono także siłę przyciągania, która to zależy od przyspieszenia grawitacyjnego oraz poszczególnych mas segmentów. Po uwzględnieniu każdej siły działającej na dany element uzyskano oparte o drugie prawo dynamiki Newtona równanie ruchu. Ze względu na to, że w modelu występują cztery masy, to powstał układ czterech równań (6.1):

$$\begin{aligned} m_u\ddot{z}_u &= -k_t\big(z_u - x(t)\big) + k_s(z_s - z_u) + c_s(\dot{z}_s - \dot{z}_u) - m_u g \\ m_s\ddot{z}_s &= -k_s(z_s - z_u) - c_s(\dot{z}_s - \dot{z}_u) + k_c(z_c - z_s) + c_c(\dot{z}_c - \dot{z}_s) - m_s g \\ m_c\ddot{z}_c &= -k_c(z_c - z_s) - c_c(\dot{z}_c - \dot{z}_s) + k_b(z_h - z_c) + c_b(\dot{z}_h - \dot{z}_c) - m_c g \\ m_h\ddot{z}_h &= -k_b(z_h - z_c) - c_b(\dot{z}_h - \dot{z}_c) - m_h g \end{aligned} \tag{6.1}$$

Równania (6.1) zostały następnie przekształcone do formy (6.2), gdzie po ich lewej stronie znajduje się jedynie druga pochodna pozycji opisywanego przez równanie elementu:

$$\begin{cases} \ddot{z_u} = \frac{-k_t(z_u - x(t)) + k_s(z_s - z_u) + c_s(\dot{z_s} - \dot{z_u})}{m_u} - g \\ \ddot{z_s} = \frac{-k_s(z_s - z_u) - c_s(\dot{z_s} - \dot{z_u}) + k_c(z_c - z_s) + c_c(\dot{z_c} - \dot{z_s})}{m_s} - g \\ \ddot{z_c} = \frac{-k_c(z_c - z_s) - c_c(\dot{z_c} - \dot{z_s}) + k_b(z_h - z_c) + c_b(\dot{z_h} - \dot{z_c})}{m_c} - g \\ \ddot{z_h} = \frac{-k_b(z_h - z_c) - c_b(\dot{z_h} - \dot{z_c})}{m_h} - g \end{cases} \quad (6.2)$$

Wymieniona w rozdziale czwartym funkcja programu SageMath do rozwiązywania równań różniczkowych zwyczajnych wymaga układu równań różniczkowych pierwszego rzędu. W celu uzyskania takiego układu użyto podstawianie:

$$\begin{aligned} \dot{z_u} &= v_u \\ \dot{z_s} &= v_s \\ \dot{z_c} &= v_c \\ \dot{z_h} &= v_h \end{aligned} \quad (6.3)$$

które pozwoliło uzyskać stosowny układ równań różniczkowych.

Układ czterech równań różniczkowych drugiego rzędu (6.2) przekształcono w układ ośmiu równań pierwszego rzędu (6.4).

$$\begin{cases} \dot{z_u} = v_u \\ \dot{z_s} = v_s \\ \dot{z_c} = v_c \\ \dot{z_h} = v_h \\ \dot{v_u} = \frac{-k_t(z_u - x(t)) + k_s(z_s - z_u) + c_s(\dot{z_s} - \dot{z_u})}{m_u} - g \\ \dot{v_s} = \frac{-k_s(z_s - z_u) - c_s(\dot{z_s} - \dot{z_u}) + k_c(z_c - z_s) + c_c(\dot{z_c} - \dot{z_s})}{m_s} - g \\ \dot{v_c} = \frac{-k_c(z_c - z_s) - c_c(\dot{z_c} - \dot{z_s}) + k_b(z_h - z_c) + c_b(\dot{z_h} - \dot{z_c})}{m_c} - g \\ \dot{v_h} = \frac{-k_b(z_h - z_c) - c_b(\dot{z_h} - \dot{z_c})}{m_h} - g \end{cases} \quad (6.4)$$

W kodzie programu zamieniono nazwy zmiennych w równaniach (6.4) na takie bez indeksów dolnych (6.5):

$$\begin{aligned} z_u &= xu \\ z_s &= xs \\ z_c &= xc \\ z_h &= xh \\ \dot{z_u} = v_u &= vu \\ \dot{z_s} = v_s &= vs \\ \dot{z_c} = v_c &= vc \\ \dot{z_h} = v_h &= vh \end{aligned} \tag{6.5}$$

Układ równań (6.4) następnie przekształcono na formę (6.6) kompatybilną z programem SageMath:

```
F1 = vu
F2 = vs
F3 = vc
F4 = vh
F5 = ( -kt*( xu-x(t))+ks*( xs-xu ) +cs*( vs-vu ) )/mu -g
F6 = ( -ks*( xs-xu ) -cs*( vs-vu ) +kc*( xc-xs ) +cc*( vc-vs ) )/ms -g
F7 = ( -kc*( xc-xs ) -cc*( vc-vs ) +kb*( xh-xc ) +cb*( vh-vc ) )/mc -g
F8 = ( -kb*( xh-xc ) -cb*( vh-vc ) )/mc -g
```

(6.6)

Funkcja drogi x(t) będzie zmieniana na potrzeby danego badania. Użyte wzory matematyczne zostały podane w rozdziale 8.

7. WYNIKI SYMULACJI DLA MODELU 1 (OPRACOWAŁ ŁUKASZ LECHNA)

7.1 WPROWADZENIE

W tym rozdziale omówione są wyniki symulacji dla pierwszego modelu matematycznego który wyprowadzono, oraz szerzej opisano w rozdziale 5 pracy. Ćwiartkę pojazdu poddano kilku różnym próbom, które polegały między na nagłych zmianach warunków drogowych po której poruszał się pojazd i obserwacji zachowania zawieszenia. W tabeli 7.1 zamieszczono dane fizyczne dla badanego modelu układu które wprowadzono do programu w celu przeprowadzenia badań. Dane te pochodzą z literatury [2] z której zapożyczony został wyprowadzony model. Na początku wykonana zostanie analiza wyników z wykorzystaniem dwóch metod badania układu: z uwzględnieniem grawitacji oraz bez, oraz przeanalizowanie wpływu grawitacji na działanie badanego w pracy układu, na której podstawie wybrana zostanie metoda dalszej analizy.

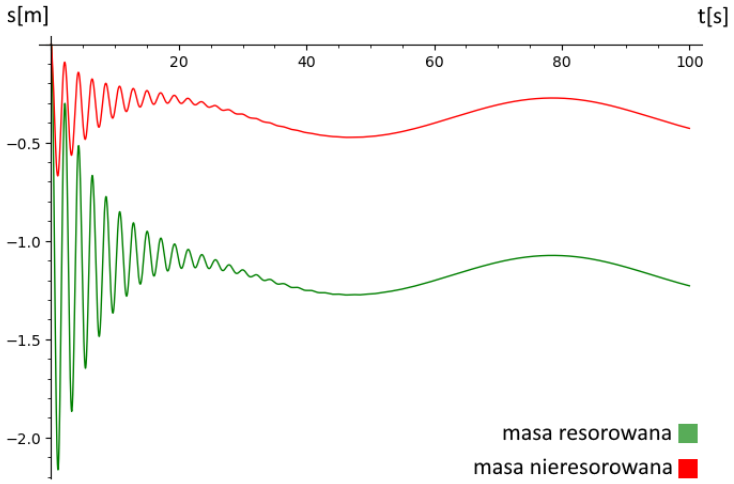
Tabela 7.1 Dane fizyczne dla badanego modelu

Wielkość fizyczna	Wartość [jednostka]
Masa resorowana	465 [kg]
Masa nieresorowana	50 [kg]
Sztywność sprężyny	5700 [N/m]
Sztywność opony	135 [kN/m]
Współczynnik tłumienia amortyzatora	290 [N/m]
Przyspieszenie ziemskie	9.81 [m/s^2]

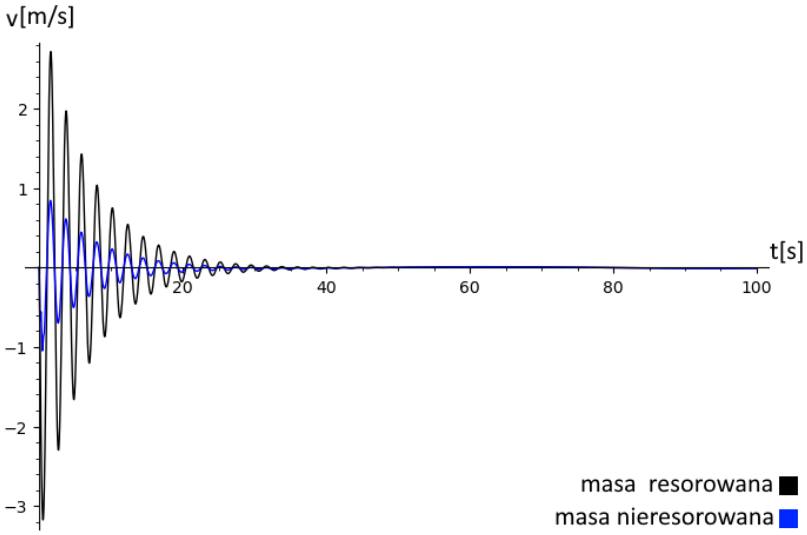
7.2 WYNIKI BADANIA DLA MODELU 1 PODCZAS JAZDY PO WYBOISTEJ DRODZE O FUNKCJI OKRESOWEJ O STAŁEJ AMPLITUDZIE I OKRESIE, ORAZ OKREŚLENIE ZNACZENIA WPŁYWU GRAWITACJI NA BADANY UKŁAD

Na rysunkach 7.1 obserwujemy zachowanie pracy badanej ćwiartki układu zawieszenia z wpływem siły grawitacji na układ. Na początku z wykresów przemieszczeń7.1a oraz prędkości obserwujemy wystąpienie silnych wibracji w układzie wywołanych jego początkowym „układaniem się". Na początku bowiem w układzie występuje imbalans działających na zeń układ sił, co powoduje że badane masy wpadają w wibracje które z czasem zostają wytłumione. Dodatkowo obserwujemy przemieszczenie się mas do wartości ujemnych, zarówno jeśli mowa o masie resorowanej jak i nieresorowanej. Różne wartości przemieszczeń dla obu mas wynikają z ich różnicy w masie, na masę resorowaną grawitacja działa znacznie silniej gdyż jej ciężar jest większy niż dziewięciokrotność masy nieresorowanej, co przekłada się na jej większy spadek.

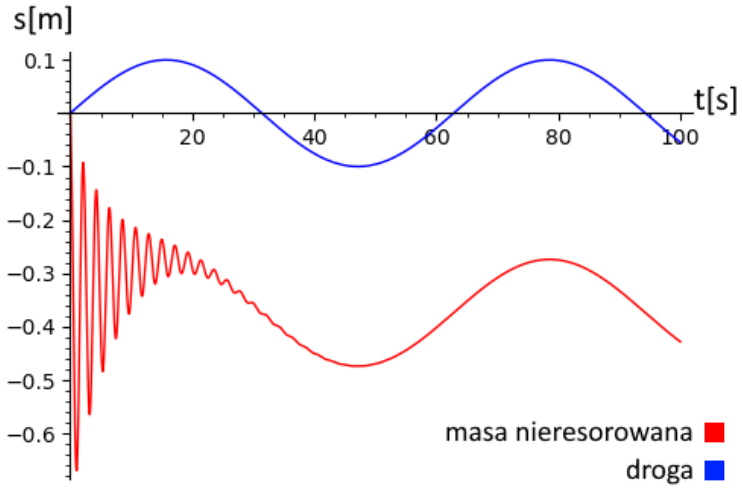
a)



b)



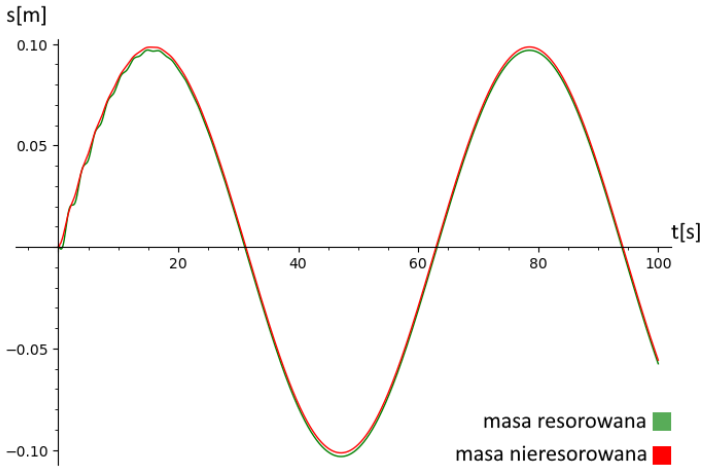
c)



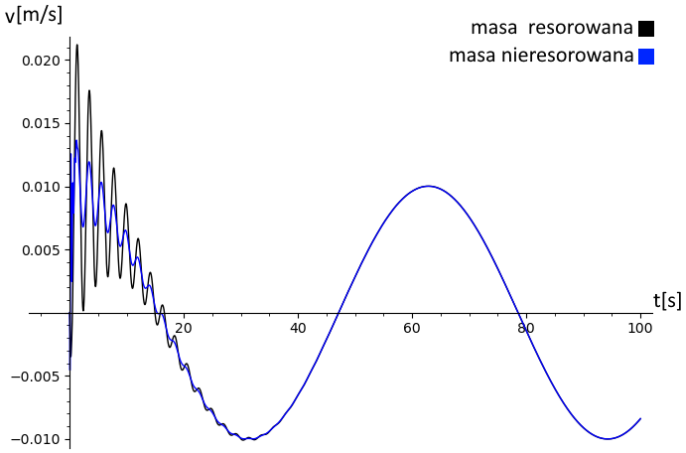
Rysunek 7.1 Wyniki obliczeń przedstawiające: a) przemieszczenia mas b) prędkości mas c) zależność ruchu masy resorowanej od drogi.

Rysunki 7.2 przedstawiają uzyskane wyniki dla układu bez udziału grawitacji. Po wprowadzeniu danych i rozpisaniu odpowiednio równań w programie uzyskaliśmy poniżej widoczne wykresy funkcji opisujące przemieszczenia masy resorowanej jak i nieresorowanej (rys 7.2a), model porusza się po wyboistej drodze o funkcji x(t), która jest funkcją sinusoidalną, o stałej amplitudzie i częstotliwości. Widoczne na wszystkich wykresach 7.2 początkowe silne wahania drgań wywołane są tym że badany model musi wpierw się „ułożyć", początkowy imbalans układu sprawia że wpada on w wibracje które stopniowo ulegają wytłumieniu przez układ zawieszenia. Biorąc pod uwagę że takie zjawisko nie występowałoby podczas badania układu rzeczywistego chyba że układ zostałby wpierw upuszczony z wysokości, w kolejnych badaniach zjawisko początkowego „układania się" układu nie zostanie uwzględnione, oraz nie będzie brane pod uwagę w trakcie badania zachowania układu zawieszenia. Wystąpienie takiego artefaktu podczas wykonywania badań uznałem jednak jako interesującą ciekawostkę, stąd postanowiłem zawrzeć go w pierwszym najprostszym badanym przykładzie, gdzie amplituda drgań drogi wynosi 0.1m a częstotliwość wynosi 0.1 Hz.

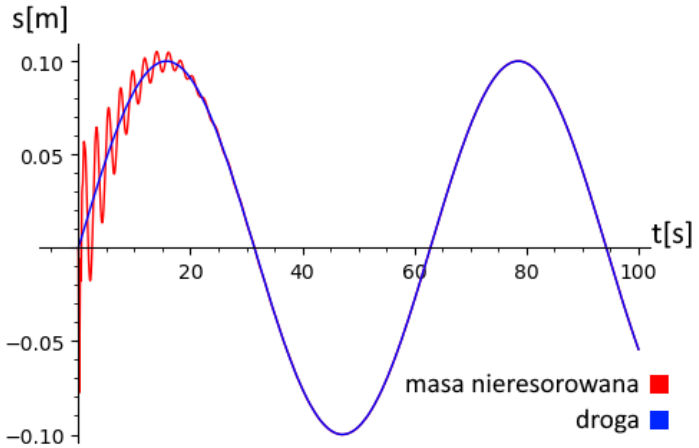
a)



b)



c)

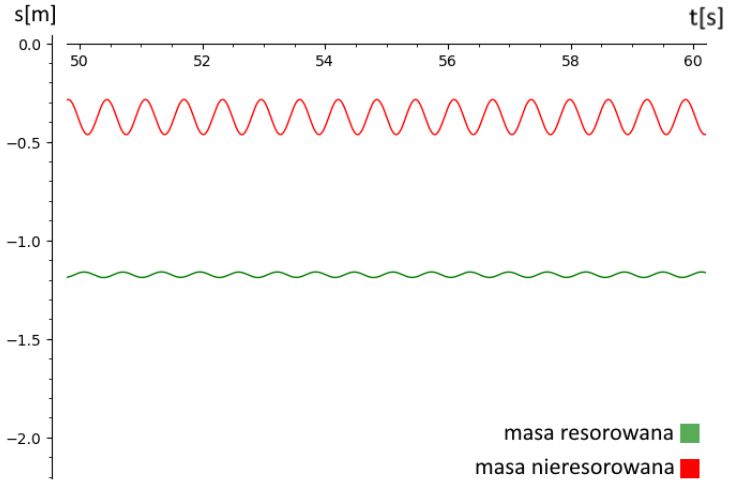


Rysunek 7.2 Wyniki obliczeń przedstawiające: a) przemieszczenia mas b) prędkości mas c) zależność ruchu masy resorowanej od drogi masy resorowanej od drogi.

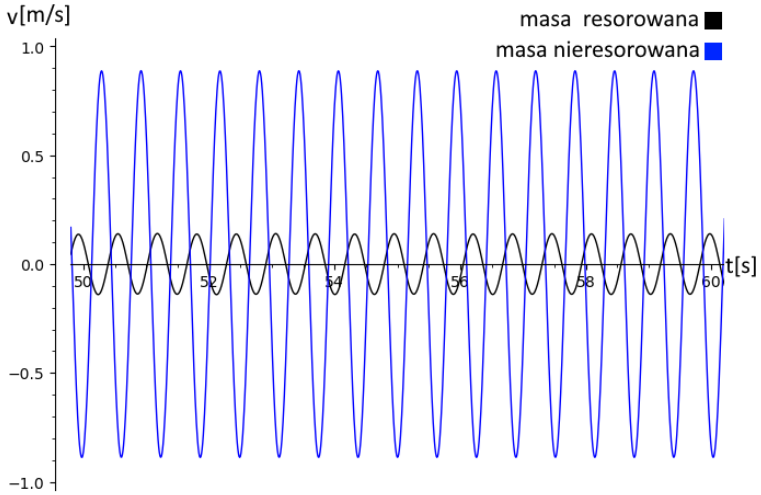
Wykresy przedstawione na rysunkach 7.2 a-c pokazują zachowanie zawieszenia samochodowego dla warunków w których badane zawieszenie nie tłumiło wibracji wywołanych wyboistością drogi, powodem tego jest zbyt niska częstotliwość drgań wymuszonych przez drogę. Aby uzyskać bardziej miarodajne wyniki, w kolejnym badaniu zwiększona została częstotliwość okresowa drogi po której porusza się pojazd na ok. 2 Hz, amplituda drgań pozostała bez zmian na poziomie 0.1 metra.

Na rysunkach 7.3 przedstawiono wyniki badania wpływu okresowej drogi o wysokiej, stałej częstotliwości wymuszeń. Z wykresu 7.4a odczytać możemy że drgania masy resorowanej są znacznie mniejsze niż w przypadku badania dla częstotliwości 0.1 Hz i amplitudy odchyleń wynoszą ok. 0.01[m], amplitudy przemieszczeń masy nieresorowanej utrzymują się w okolicach 0.10[m] i są zbliżone do drgań drogi co obrazuje wykres 7.3c. Prędkości drgań masy nieresorowanej (rys 7.3b) zwiększyły się za to prawie dziewięciokrotnie, osiągając amplitudy w granicach 0.9 [m/s]. Na wykresie 7.3a widać także że masa resorowana drga z nieco mniejszą częstotliwością od masy resorującej.

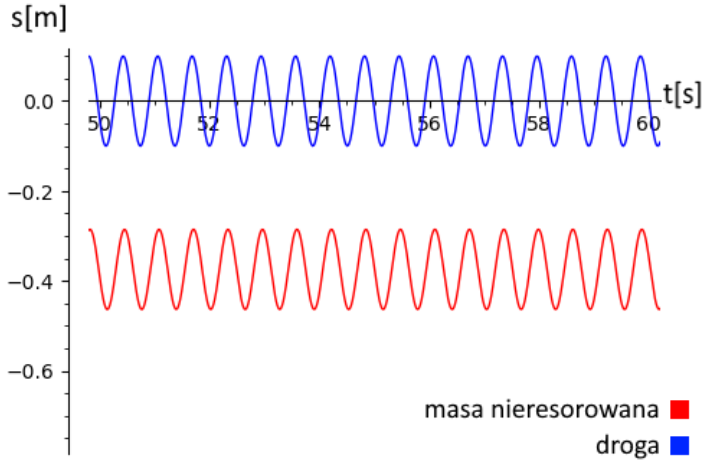
a)



b)



c)

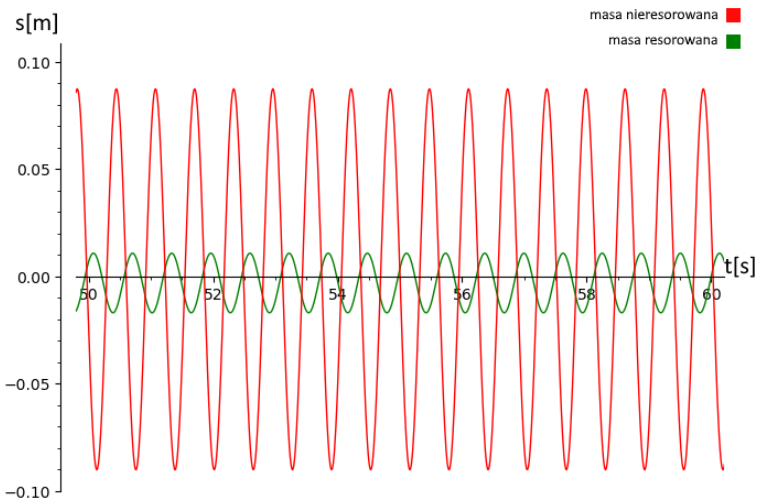


Rysunek 7.3 Wyniki obliczeń przedstawiające: a) przemieszczenia mas b) prędkości mas c) zależność ruchu masy resorowanej od drogi.

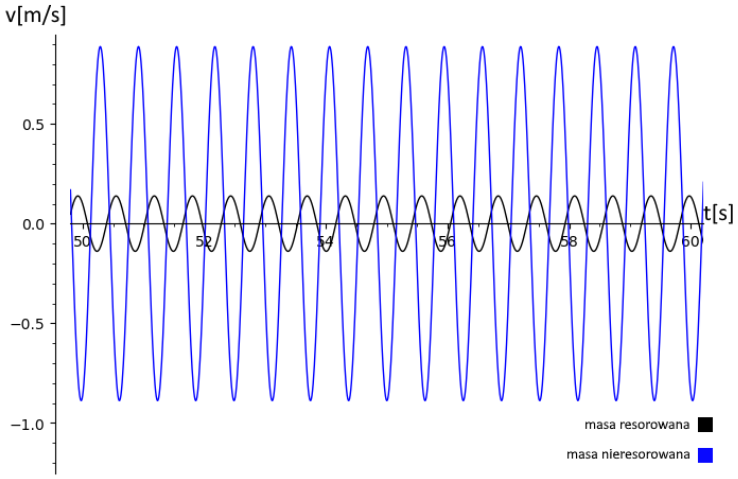
Przez znaczne zwiększenie częstotliwości okresowej drogi próby czasowe wykresów na rysunkach 7.4 zostały zmniejszone do 10 sekund dla zachowania czytelności i przejrzystości zawartych na nich danych. Z wykresów (7.4a) przemieszczeń możemy łatwo odczytać że zawieszenie w istocie tłumi drgania masy resorowanej poruszającej się po drodze o charakterystyce funkcji okresowej. Drgania masy resorowanej posiadają widocznie mniejszą amplitudę, podczas gdy drgania masy resorującej oddają w dużej mierze kształt drogi. Przemieszczenia masy resorującej na wykresie (7.4b) posiadają nieznacznie mniejsze amplitudy od drogi, powodem takiego zjawiska jest opona badanego modelu która posiada wysoki współczynnik elastyczności k_t równy 135 kN/m, zatem w domyśle nieznaczne zmniejszenie przemieszczeń masy nieresorowanej jest spowodowane zmianą kształtu opony co oznacza że jest ona wciąż w stałym kontakcie z drogą i zapewnia odpowiednią stabilność podczas przejazdu.

Wyniki symulacji zaprezentowane na rysunkach 7.4 ukazują że efektywność tłumienia wibracji badanego układu jest na zachwycającym poziomie na rysunku 7.4a obserwujemy bowiem nie tylko spadek amplitudy drgań względem drogi o prawie 90%, a względem masy nieresorowanej o 86,5% ale także zmniejszenie się częstotliwości drgań masy resorowanej do ok. 1,5 Hz, co względem drgań drogi wynoszących 2 Hz daje nam blisko 25% spadek częstotliwości drgań, co może pozytywnie przełożyć się na komfort jazdy. Bardzo pozytywne wyniki najprawdopodobniej są spowodowane stałą okresowością wymuszeń drogi, w przypadku niestałości kształtu drogi czy okresowości jej wymuszeń układ znacznie gorzej tłumiłby drgania.

a)



b)

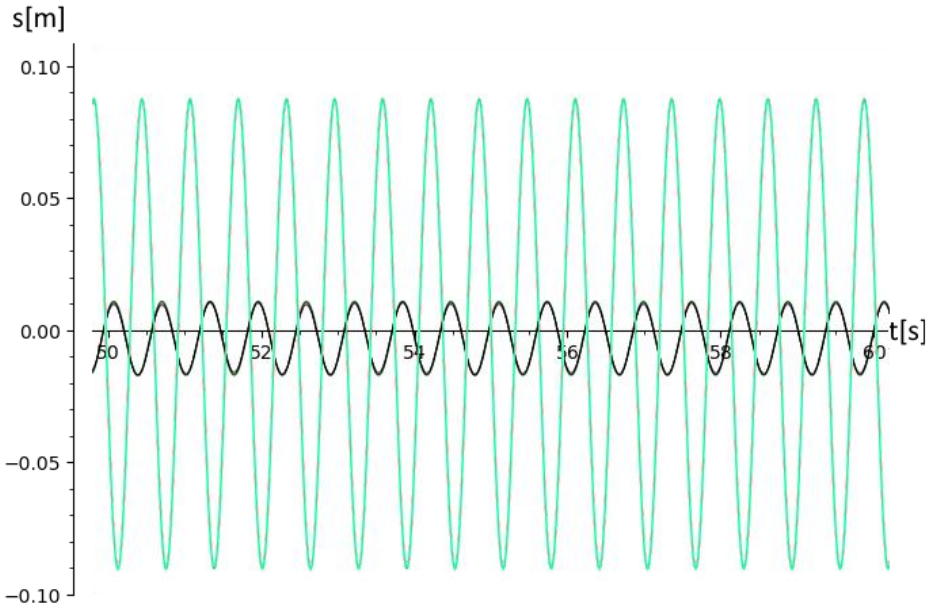


c)

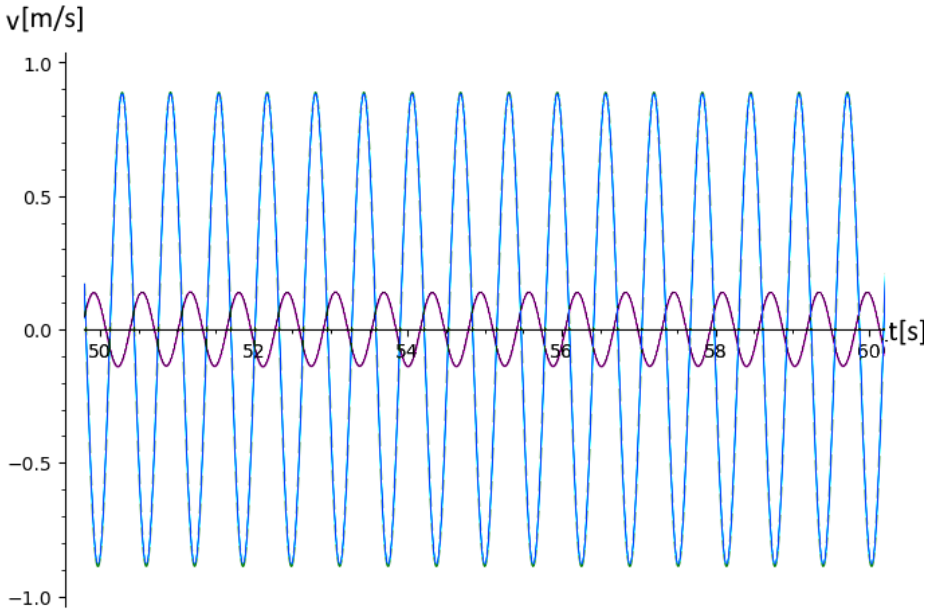
Rysunek 7.4 Wyniki obliczeń przedstawiające: a) przemieszczenia mas b) prędkości mas c) zależność ruchu masy resorowanej od drogi.

Po porównaniu wyników uzyskanych dla obu metod badania z wykresów 7.3 i 7.4 zauważamy że wyniki które uzyskaliśmy, są identyczne jeśli nie weźmiemy pod uwagę początkowej nierównowagi układu którą ukazano na wykresach 7.1 i 7.2. Obie linie przemieszczeń w układzie nakładają się na siebie po nałożeniu obu wykresów na siebie z wykorzystaniem programu graficznego i wcześniejszym ujednoliceniem skali eksportowanych wykresów. Jednolitość uzyskiwanych linii wykresów powtarzała się w przypadku innych przeprowadzonych przez nas badań. Oznacza to że udział grawitacji w układzie nie ma tu znaczącego przełożenia na interesujące nas aspekty i pracę układu tj. wielkości amplitud, czy częstotliwość, gdyż jej wpływ na układ po uzyskaniu przez zeń równowagi, ogranicza się tu tylko do stałego przemieszczenia otrzymywanych wyników w dół osi o stałą wartość. W dalszych badaniach postanowiliśmy więc zmienić wartość siły grawitacji na 0 dla reprezentowanych wykresów, dla zwiększenia klarowności uzyskanych wykresów i ich prostszego odczytu.

a)



b)



Rysunek 7.5 Wyniki przedstawiające: a) nałożone przemieszczenia mas z rys 7.3a i 7.4a b) nałożone wykresy prędkości z rys 7.3b i 7.4b.

7.3 WYNIKI BADANIA DLA MODELU 1 PODCZAS PRZEJAZDU PO DRODZE CZĘSTOTLIWOŚCI OKRESOWEJ ZMIENIAJĄCEJ SIĘ W CZASIE

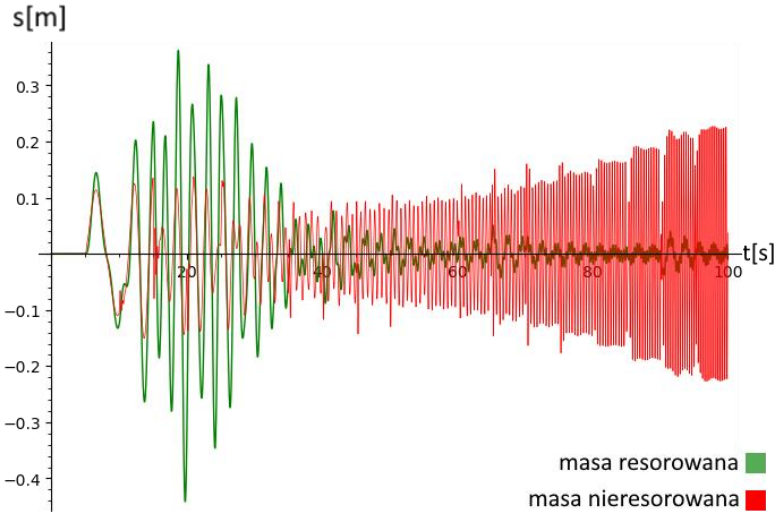
By zbadać zachowanie zawieszenia podczas jazdy na różnych rodzajach wyboistej nawierzchni poddano go symulacji drogi której to parametry zmieniają się wraz z czasem przejazdu. W ten sposób będziemy w stanie dokładniej określić działanie i sprawność badanego przez nas układu zawieszenia dla różnych częstotliwości drgań układu, oraz zbadanie reakcji układu na zmiany częstotliwości drgań. Użyta dla tego badania funkcja zmian kształtu drogi:

$$X_t(t) = A \cdot sin(\omega \cdot \theta), \tag{7.1}$$

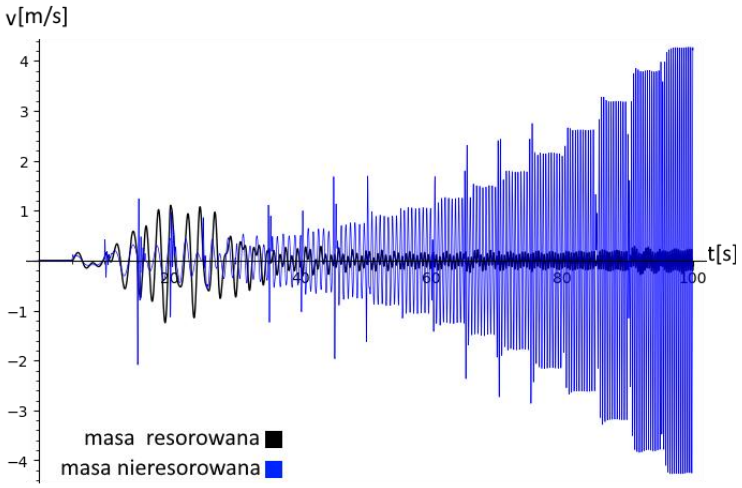
$$\theta = floor\left(\frac{t}{5}\right) \cdot (t - 5), \tag{7.2}$$

wywołuje nagłe zmiany częstotliwości drgań układu w pięciosekundowych odstępach czasu, w których to częstotliwość pozostaje stała. Amplituda drogi wynosi 0.1[m] i pozostaje niezmienna. Widoczne na wykresie 7.6a przemieszczenia masy nieresorowanej oraz resorowanej ukazują ciekawą zależność zmiany amplitud drgań w układzie wraz ze zmieniającymi się stopniowo częstościami drgań. Widzimy tutaj 7.6a że dla niskich częstotliwości drgań drogi amplituda drgań masy resorowanej jest znacznie większa, szczytowe odchylenie drgającego zawieszenia to 0.42 metra. Wraz ze zwiększającą się stopniowo częstotliwością drgań amplituda drgań masy resorowanej zaczęła się zmniejszać po 20 sekundzie badania. Obserwujemy także stopniowe zwiększanie się przemieszczeń masy resorującej, oraz znacznego zwiększania się prędkości drgań tej masy 7.6b. Aby dokonać dokładniejszej analizy otrzymanych wyników dane widoczne na rysunkach 7.6 podzielono na dwa przedziały czasowe jednakowej długości, dla uzyskania lepszej czytelności wyników i umożliwienia ich dokładniejszej analizy.

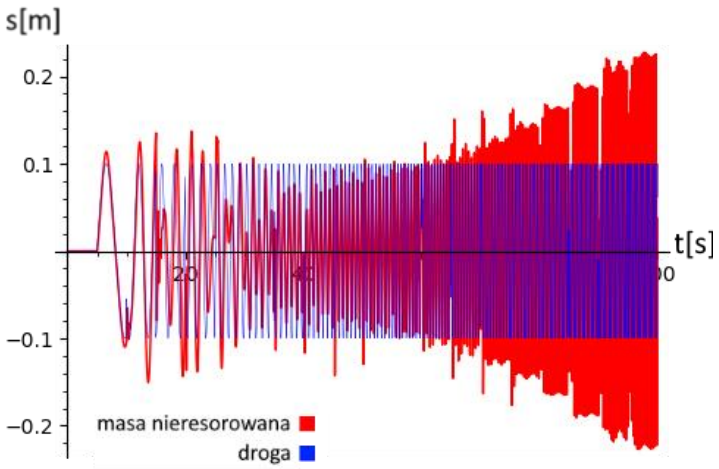
a)



b)



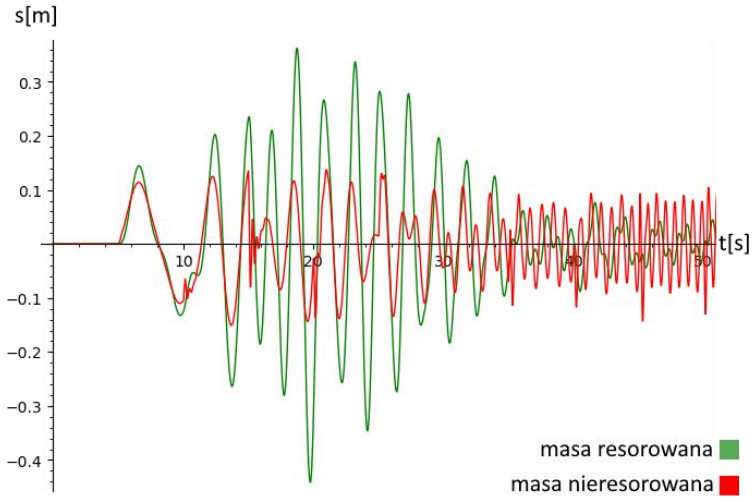
c)



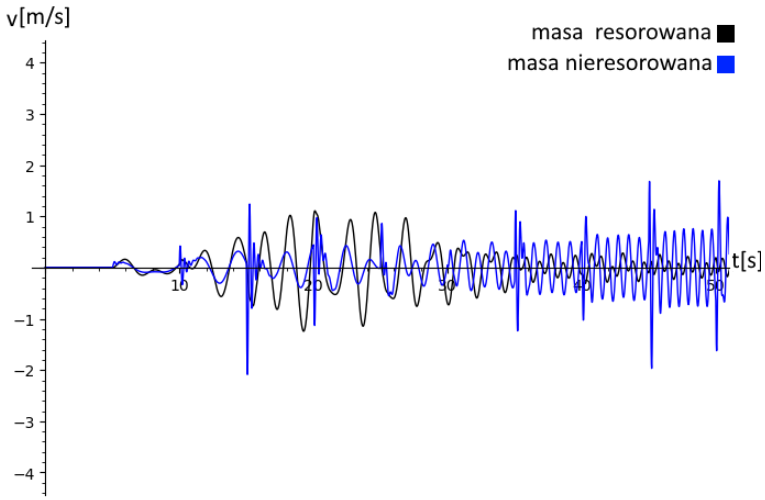
Rysunek 7.6 Wyniki obliczeń przedstawiające: a) przemieszczenia mas b) prędkości mas c) zależność ruchu masy resorowanej od drogi.

Z wykresu 7.7a możemy wywnioskować że w zakresie czasowym 15-20[s] masa resorowana wpada w rezonans przy częstotliwości 0,5 Hz. W tym samym czasie obserwujemy że masa resorująca drga nieharmonicznie co wywołane jest nie tylko zmieniającym się profilem drogi ale także wymuszeniem na niej tłumienia drgań masy resorowanej. Uskoki widoczne w wykresach prędkości 7.7b dla masy nieresorowanej wywołane są gwałtownymi zmianami drogi lepiej widocznymi na rysunku 7.7c z każdym zwiększeniem częstotliwości wymuszeń, obserwujemy swego rodzaju anomalie w funkcji drogi wywołane działaniem użytej funkcji, przekładające się na reakcję masy nieresorowanej. Masa resorowana znacznie zmniejsza wielkość amplitud drgań po około 30 sekundach od rozpoczęcia badania gdy częstość drgań wymuszonych drogi jest w okolicach 0.9 Hz. Można więc założyć że 0.9 Hz jest częstotliwością graniczną dla badanego układu po której przekroczeniu znacząco wzrasta efektywność tłumienia drgań wymuszonych.

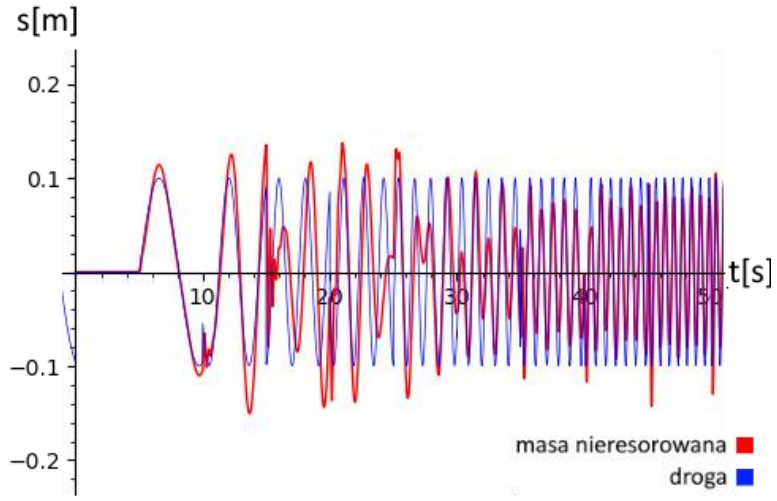
a)



b)



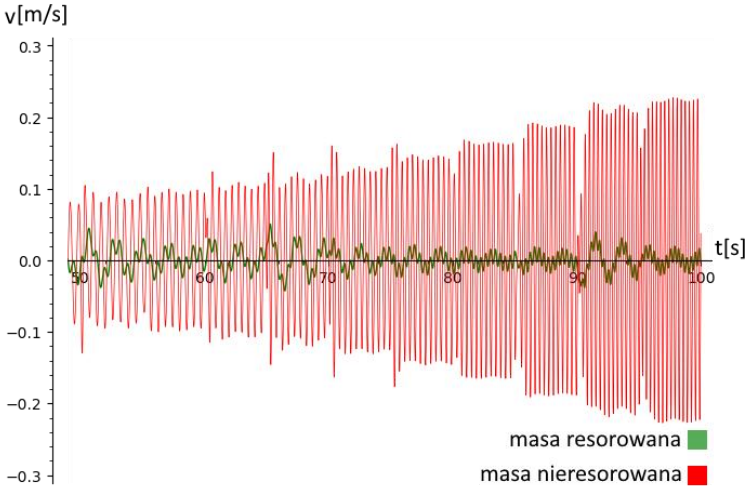
c)



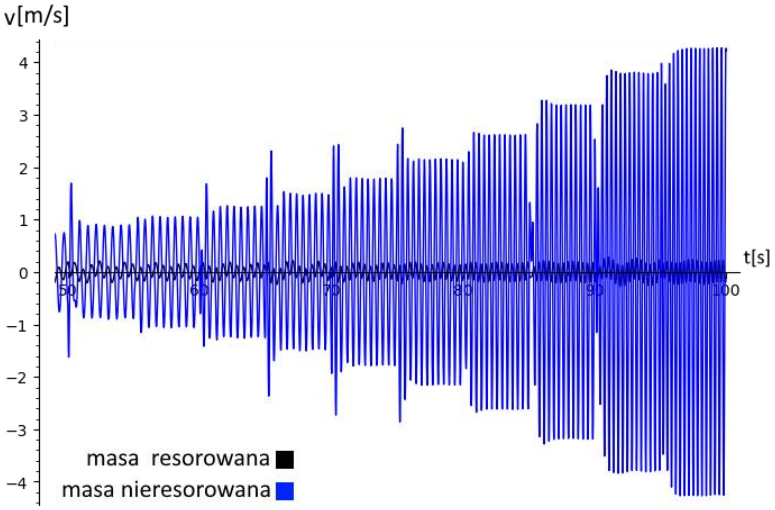
Rysunek 7.7 Wyniki obliczeń przedstawiające: a) przemieszczenia mas b) prędkości mas c) zależność ruchu masy resorowanej od drogi.

Wykresy 7.8 przedstawiają zachowanie badanego układu w przedziale czasowym od 50 do 100 [s]. Na wykresie 7.8a obserwujemy że masa resorowana krótkie drgania, tworząc jednocześnie linię przemieszczeń na kształt sinusoidy, dla przemieszczeń masy resorowanej prędkość drgania pozostaje na podobnym poziomie i utrzymuje się w okolicach 0.2 [m/s], podczas gdy prędkość drgania masy nieresorowanej zwiększa się z każdym wzrostem częstotliwości wymuszeń drogi (7.8b), wraz z amplitudami jej przemieszczeń w sposób przypominający rosnącą funkcję wykładniczą.

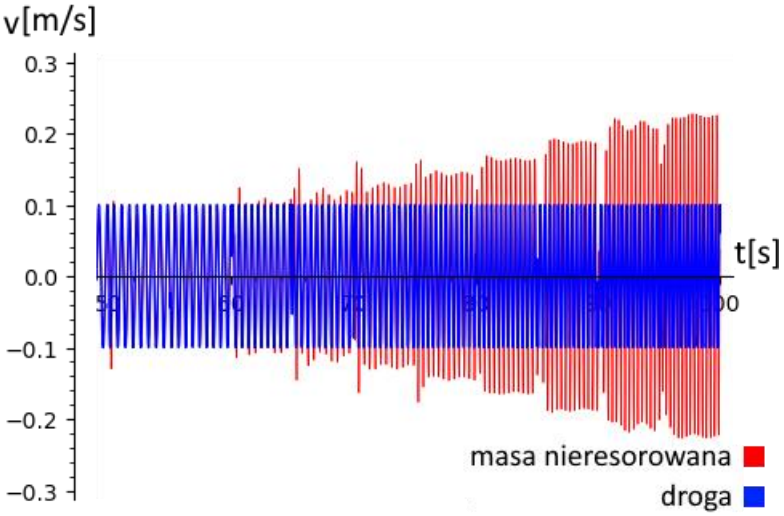
a)



b)



c)



Rysunek 7.8 Wyniki obliczeń przedstawiające: a) przemieszczenia mas b) prędkości mas c) zależność ruchu masy resorowanej od drogi.

8. WYNIKI SYMULACJI DLA MODELU 2 (OPRACOWAŁ MICHAŁ WIELGAT)

8.1 WPROWADZENIE

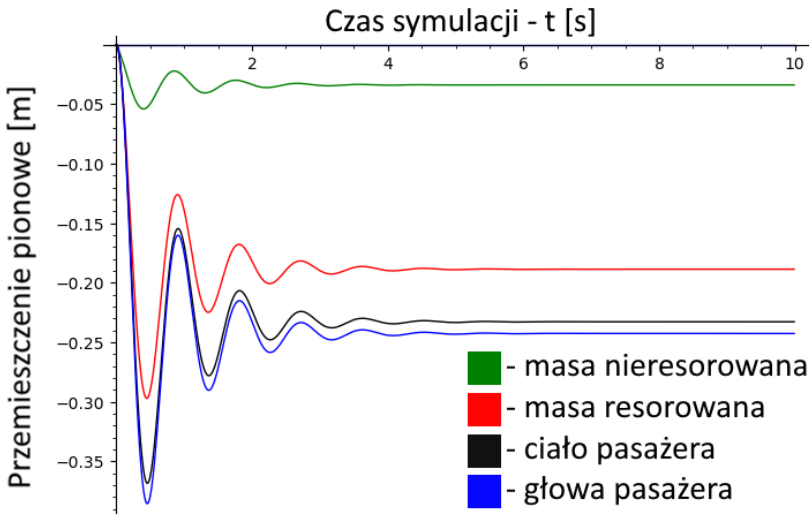
W tym rozdziale zostały omówione wyniki symulacji z użyciem modelu matematycznego z rozdziału 6, to jest modelu ćwiartkowego samochodu złożonego z czterech mas – koła, nadwozia, ciała oraz głowy kierowcy. Poszczególne parametry symulacji, takie jak wartości mas obiektów, sztywności sprężyn oraz współczynniki tłumienia zostały umieszczone w tabeli 8.1. Dane te zostały zapożyczone z literatury [9].

Tabela 8.1 Domyślne parametry symulacji użyte w modelu 4 segmentów.

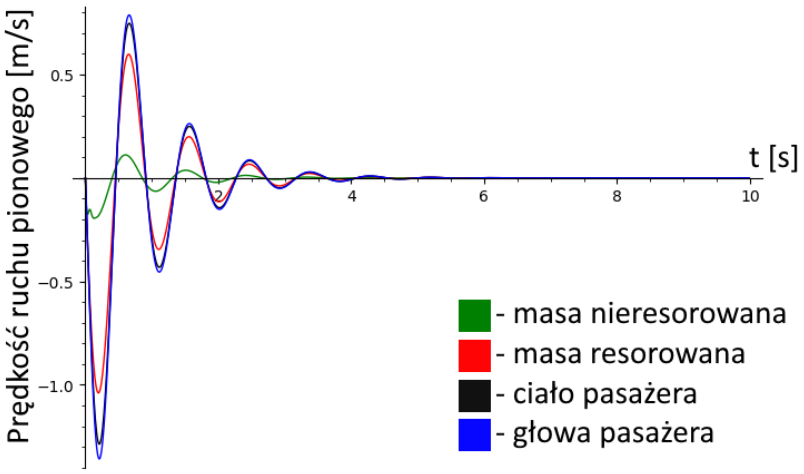
Wielkość fizyczna	Wartość [jednostka]
Masa resorowana	241.65 [kg]
Mass nieresorowana	21.46 [kg]
Współczynnik sztywności sprężyny	21027.63 [N/m]
Współczynnik tłumienia amortyzatora	1183.2 [Ns/m]
Współczynnik sztywności opony	102017.2 [N/m]
Masa głowy pasażera	20 [kg]
Masa ciała pasażera	45 [kg]
Współczynnik sztywności ciała pasażera	45000 [N/m]
Współczynnik sztywności fotela	20000 [N/m]
Współczynnik tłumienia ciała pasażera	1360 [Ns/m]
Współczynnik tłumienia fotela	1650 [Ns/m]
Przyspieszenie grawitacyjne	9.81 [m/s^2]

8.2 BADANIE MODELU 2 PODCZAS JAZDY PO RÓWNEJ DRODZE

Badanie to miało na celu potwierdzenie funkcjonalności programu oraz obserwację zachowania modelowanego zawieszenia przy braku zakłóceń w postaci oscylujących wybrzuszeń drogi. Efekt ten uzyskano poprzez ustawienie wartości amplitudy funkcji x(t) na 0. Czas symulacji ustawiono na 10 sekund. Wszystkie pozostałe współczynniki pozostały niezmienione, z wartościami takimi jak w tabeli 8.1.



Rysunek 8.1 Przemieszczenie poszczególnych elementów modelu w czasie, równa droga, domyślne wartości współczynników.



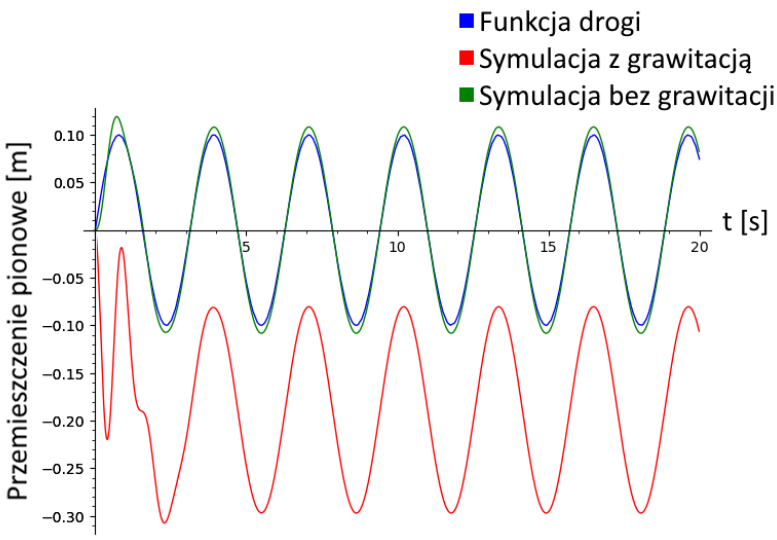
Rysunek 8.2 Prędkość ruchu pionowego poszczególnych elementów modelu w czasie, równa droga, domyślne wartości współczynników.

Mimo braku nierówności nawierzchni to powstały pewne oscylacje przemieszczeń na wykresie 8.1. Po rozpoczęciu symulacji siła przyciągania wywołała ruch elementów systemu. Po zmianie ich względnych przemieszczeń symulowane sprężyny zadziałały przeciwległą siłą, ograniczając ruch zawieszenia. Wprowadziło to układ w oscylacje, które to po upływie około pięciu sekund zostały wytłumione przez symulowane amortyzatory. Po ustaleniu pozycji wszystkich elementów nie zaobserwowano dalszych zmian.

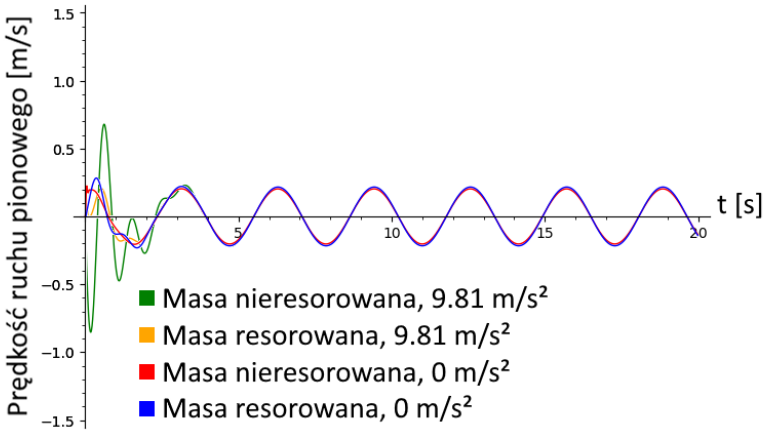
8.3 BADANIE WPŁYWU PRZYCIĄGANIA NA WYNIKI UZYSKANE Z MODELU 2

W badaniu 8.2 każdy z segmentów ustalił się na innej wysokości, co może utrudniać wizualne porównywanie ich ruchu. Dlatego też podobnie jak w rozdziale 7 sprawdzono wpływ grawitacji na wyniki symulacji. W tym celu porównano uzyskane w dwóch różnych symulacjach prędkości ruchu pionowego wszystkich elementów układu. Jedyna różnica między tymi symulacjami to współczynnik przyspieszenia grawitacyjnego – w pierwszej użyto wartości 9.81 m/s^2, a w drugiej użyto 0 m/s^2. Jako funkcję drogi zastosowano sinusoidę o stałej częstotliwości i amplitudzie, zapisaną wzorem 8.1. Współczynnik b ustala wysokość nierówności, a Ω zmienia częstotliwość funkcji.

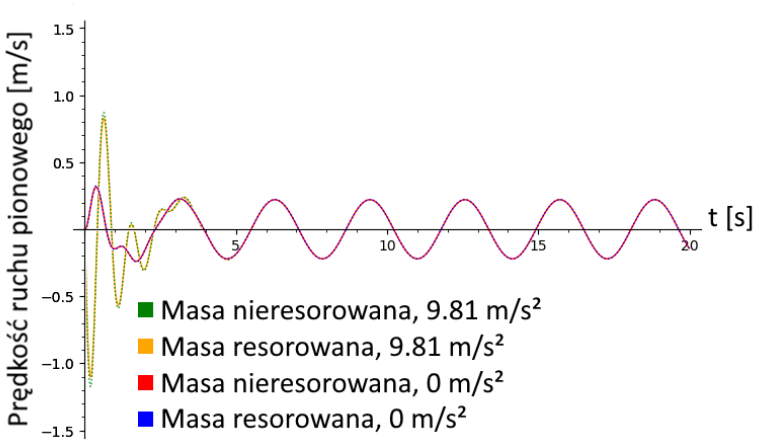
$$x(t) = bsin(\Omega t) \tag{8.1}$$



Rysunek 8.3 Przemieszczenie pionowe w czasie masy resorowanej dla symulacji z grawitacją oraz symulacji bez grawitacji.



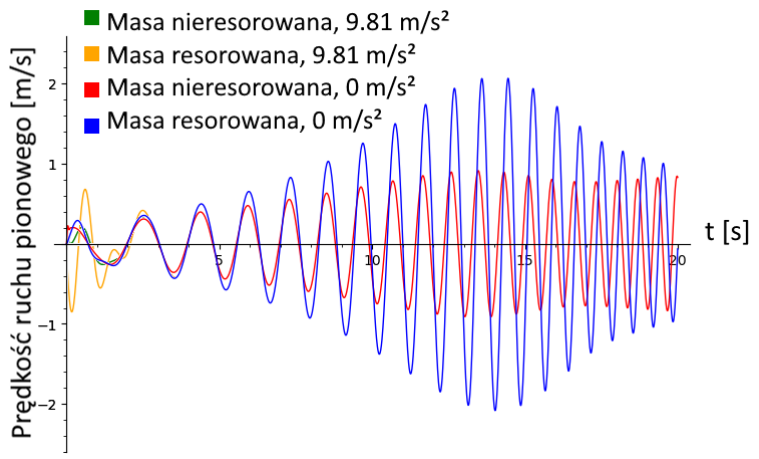
Rysunek 8.4 Porównanie prędkości ruchu pionowego koła i nadwozia samochodu w symulacjach z grawitacją oraz bez grawitacji, droga to sinusoida o stałych parametrach.



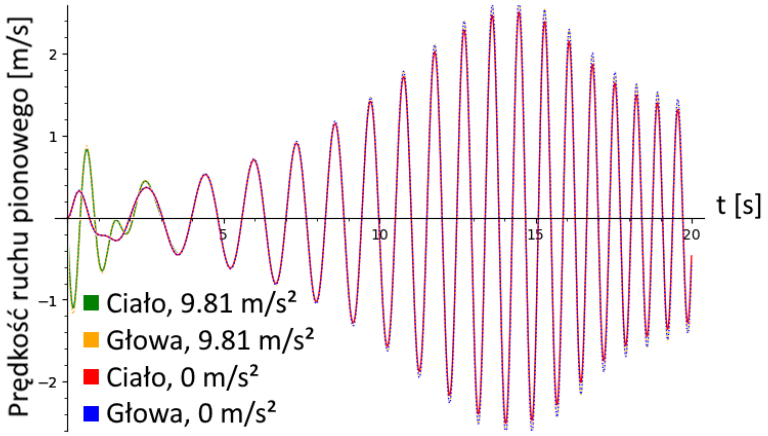
Rysunek 8.5 Porównanie prędkości ruchu pionowego ciała i głowy kierowcy w symulacjach z grawitacją oraz bez grawitacji, droga to sinusoida o stałych parametrach.

Zauważalne różnice pomiędzy wynikami symulacji występują tylko do około czwartej sekundy, po czym prędkości ruchów pionowych zaczynają się ze sobą pokrywać. W celu potwierdzenia tego wyniku zostały przeprowadzone dalsze symulacje na zmodyfikowanej funkcji drogi. Użyto zapisaną wzorem (8.2) funkcję sinusoidalną o stałej amplitudzie oraz zwiększającej się z czasem częstotliwością.

$$x(t) = bsin\left(\left(\frac{t}{10} + 1\right) \cdot \Omega t\right) \tag{8.2}$$



Rysunek 8.6 Porównanie prędkości ruchu pionowego koła i nadwozia samochodu w symulacjach z grawitacją oraz bez grawitacji, droga to sinusoida o zmiennej częstotliwości.



Rysunek 8.7 Porównanie prędkości ruchu pionowego ciała i głowy kierowcy w symulacjach z grawitacją oraz bez grawitacji, droga to sinusoida o zmiennej częstotliwości.

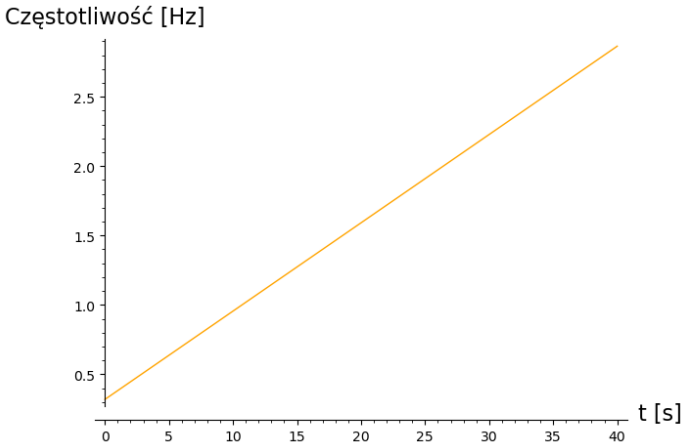
Podobnie jak w badaniu z użyciem wzoru (8.1), prędkości ruchów pionowych w symulacji z grawitacją oraz bez grawitacji różnią się jedynie przez kilka pierwszych sekund. Oznacza to, że grawitacja nie ma zauważalnego wpływu na uzyskane wyniki. Dlatego też w następnych badaniach wartość współczynnika przyspieszenia grawitacyjnego została ustawiona na 0 m/s².

8.4 WYNIKI BADANIA MODELU 2 NA FUNKCJI DROGI O ZMIENNEJ CZĘSTOTLIWOŚCI

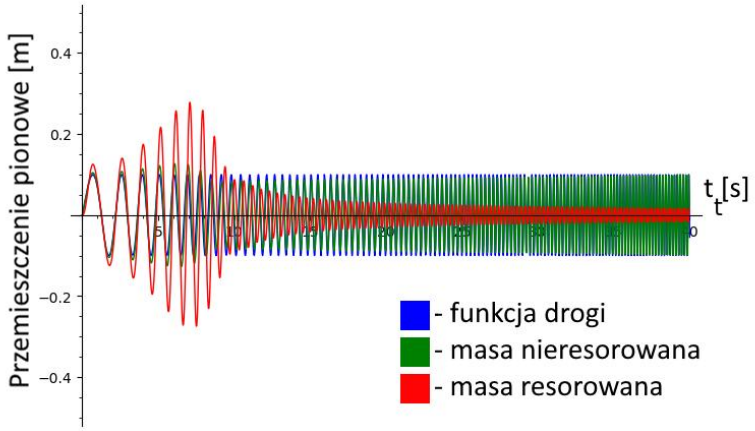
W badaniu 8.3 oprócz głównego celu, jakim było wykluczenie wpływu grawitacji na wyniki, można zauważyć ciekawą zależność zmiany prędkości ruchu pionowego od wzrostu częstotliwości nierówności na drodze. Aby dogłębniej przeanalizować to zjawisko zwiększono czas symulacji do 50 sekund, oraz zmodyfikowano równanie drogi (8.2) w celu zwiększenia szybkości wzrostu częstotliwości występowania nierówności. Nowa funkcja drogi przedstawiona jest we wzorze (8.3). Dodatkowo utworzono nową funkcję (8.4) do rysowania wykresów zależności częstotliwości tych nierówności od czasu, która lepiej przedstawia zmiany występujące na drodze symulowanego pojazdu.

$$x(t) = bsin\left(\left(\frac{t}{5} + 1\right) \cdot \Omega t\right) \tag{8.3}$$

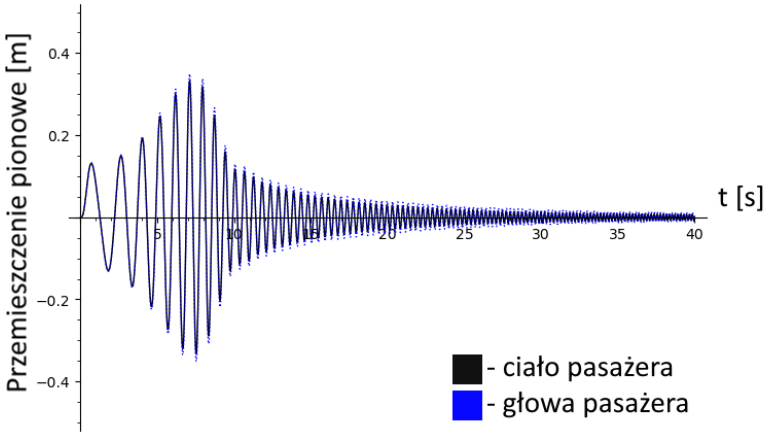
$$f(t) = \frac{\left(\frac{t}{5}+1\right)\cdot\Omega}{2\pi} \tag{8.4}$$



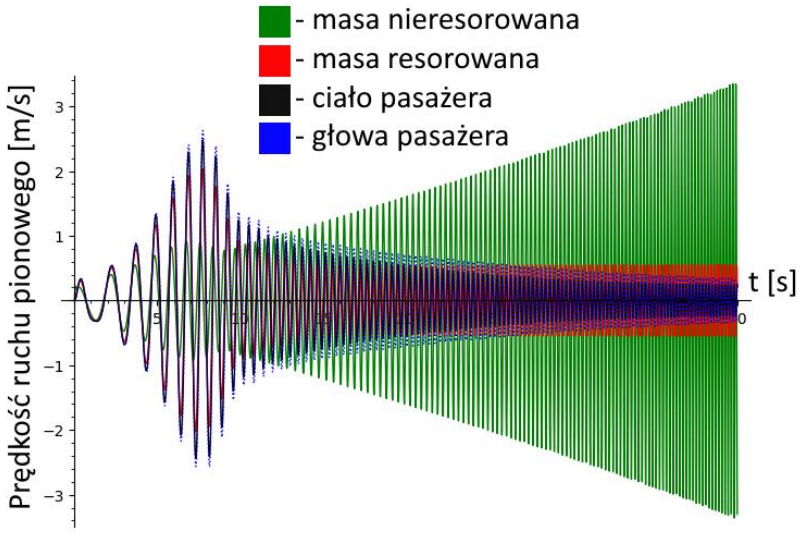
Rysunek 8.8 Częstotliwość nierówności na drodze w zależności od czasu symulacji.



Rysunek 8.9 Przemieszczenie pionowe masy resorowanej oraz masy nieresorowanej w czasie symulacji, funkcja drogi o zmieniającej się częstotliwości



Rysunek 8.10 Przemieszczenie pionowe ciała oraz głowy pasażera w czasie symulacji, funkcja drogi o zwiększającej się częstotliwości.



Rysunek 8.11 Prędkość ruchu pionowego segmentów modelu w czasie, funkcja drogi o zwiększającej się częstotliwości.

Przez pierwsze kilka sekund symulacji amplitudy przemieszczeń wszystkich elementów modelu wzrastały wraz ze wzrostem częstotliwości funkcji drogi. Po upływie około siedmiu i pół sekundy otrzymano największe przemieszczenia. Częstotliwość nierówności w tym punkcie wyniosła 0,8 Hz. Jest to prawdopodobnie częstotliwość rezonansu układu. Przez resztę symulacji maksymalne przemieszczenia wszystkich segmentów malały z czasem. Na wykresie 8.11 ciało oraz głowa pasażera wykazuje coraz to mniejsze wartości prędkości ruchu pionowego. Od około 20 sekundy amplituda oscylacji masy resorowanej przestała znacząco maleć, stabilizując się na wartości 0,5 m/s. Maksymalne prędkości pionowe masy nieresorowanej wzrastają razem z częstotliwością nierówności, co można wyjaśnić stwierdzeniem, że koło symulowanego pojazdu porusza się zgodnie z kształtem drogi.

Mimo zmniejszającej się amplitudy drgań ciała kierowcy to należy pamiętać, że przy odpowiednio dużych częstotliwościach drgania zawieszenia, o zakresach których wspomniano w rozdziale 2.3, może zacząć się pojawiać dyskomfort oraz pogorszenie sprawności wykonywania zadań przez człowieka.

9. PODSUMOWANIE I WNIOSKI (OPRACOWALI ŁUKASZ LECHNA, MICHAŁ WIELGAT)

Praca ta przedstawia możliwości zastosowania analizy numerycznej dla badań zawieszenia samochodowego. W pracy przedstawiono graficzny i matematyczny model pojazdu, oraz uwzględniono proces kreowania modelu matematycznego przez wyprowadzanie równań różniczkowych użytych w pracy, a następnie poddano analizie uzyskane wyniki po odpowiednim wprowadzeniu ich do programu obliczeniowego. Zaletą zastosowanego sposobu badania układów zawieszenia poprzez ich analizę numeryczną, jest powtarzalność wyników, które będą jednakowe niezależnie od tego ilekroć wykonamy ponowne badanie dla danych parametrów układu. Dzięki temu jesteśmy w stanie w łatwy sposób odtworzyć uzyskane wyniki np. w przypadku ich utraty. Kolejne ważne zalety tego typu analizy to jej niski koszt, oraz szybkość przeprowadzenia, czy zmiany parametrów badanego modelu.

Użyty tutaj program posiada też duże możliwości dalszego rozwoju. W tej pracy opisano stosunkowo prosty przykład ćwiartkowego modelu pojazdu. Równania różniczkowe mogą być użyte do opisania bardziej skomplikowanych i dokładniejszych modeli. Jednym z możliwych kierunków rozbudowy tej symulacji byłoby, na przykład, stworzenie modelu opisującego ruch oraz wzajemne interakcje przedniej i tylnej osi pojazdu. Opisany w tej pracy przykład stanowi dobrą podstawę do stworzenia symulacji innych układów fizycznych.

LITERATURA

1. A. MITURA, MODELOWANIE DRGAŃ NIELINIOWEGO ZAWIESZENIA POJAZDU SAMOCHODOWEGO TŁUMIENIEM MAGNETO REOLOGICZNYM, LUBLIN 2010.

2. TESFAYE O. TEREFE , HIRPA G. LEMU, SOLUTION APPROACHES TO DIFFERENTIAL EQUATIONS OF MECHANICAL SYSTEM DYNAMICS: A CASE STUDY OF CAR SUSPENSION SYSTEM, ADVANCES IN SCIENCE AND TECHNOLOGY RESEARCH JOURNAL, VOLUME 12 NO.2, STRONY 266-273, 2018.

3. FERDEK U., ŁUCZKO J., MODELING AND ANALYSIS OF A TWIN-TUBE HYDRAULIC SHOCK ABSORBER, JOURNAL OF THEORETICAL AND APPLIED MECHANICS VOL. 50,2, PP.627-638, WARSZAWA 2012.

4. TOMASZ NABAGŁO, SYNTEZA UKŁADU STEROWANIE SEMIAKTYWNEGO ZAWIESZENIA SAMOCHODU Z ELEMENTAMI MEGNETOREOLOGICZNYMI, ROZPRAWA DOKTORSKA, KRAKÓW 2006.

5. P. ZIMMERMANN, A. CASAMAYOU, N. COHEN, G. CONNAN, T. DUMONT, L. FOUSSE, F. MALTEY, M. MEULIEN, M. MEZZAROBBA, C. PERNET, N. M. THIÉRY, E. BRAY, J. CREMONA, M. FORETS, A. GHITZA, H. THOMAS, COMPUTATIONAL MATHEMATICS WITH SAGEMATH , 2018.

6. PROBLEMS IN PHYSICS (WITH SAGEMATH), MARCIN KOSTUR, JERZY ŁUCZKA, ŁUKASZ MACHURA, INSTYTUT FIZYKI UNIWERSYTETU ŚLĄSKIEGO, 2019.

7. JÓZEF GIERGIEL, DRGANIA UKŁADÓW MECHANICZNYCH – WYDANIE DRUGIE ZMIENIONE, KRAKÓW 1986.

8. THE ORIGINS OF SAGEMATH – CREATING A VIABLE OPEN SOURCE ALTERNATIVE TO MAGMA, MAPLE, MATHEMATICA, AND MATLAB, WILLIAM STEIN, SAGEMATH, INC., AND UNIVERSITY OF WASHINGTON, 2016.

9. ANIRBAN. C. MITRAA, GOURAV. J. DESAIB, SAAISH. R. PATWARDHANC, PARAG H. SHIRKED,WASEEM M. H. KURNEE, NILOTPAL BANERJEEF, OPTIMIZATION OF PASSIVE VEHICLE SUSPENSION SYSTEM BY GENETIC ALGORITHM, PROCEDIA ENGINEERING 144, STRONY 1158 – 1166, 2016 .

10. ZALEWSKI J., WPŁYW WYBRANYCH PARAMETRÓW RUCHU NA NIEKTÓRE CECHY EKSPLOATACYJNE SAMOCHODU, RADOM, ITEE-PIB W RADOMIU, 2018.

11. DOKUMENTACJA PROGRAMU SAGEMATH, **https://doc.sagemath.org/** [DOSTĘP DNIA: 23.01.2023].

SPIS RYSUNKÓW

SPIS TABEL

ZAŁĄCZNIK NR 1 – KOD ŹRÓDŁOWY PROGRAMU SYMULACJI PIERWSZEGO MODELU

```
reset()
var('t x1 x2 x3 x4')

# parameters of mechanical system
m1=465   #[kg]
m2=50    #[kg]
k=5700   #[N/m]
kt=13500 #[N/m]
c=290    #[N/m]


g=9.81   #[m/s^2]


# the displacement input due to the surface irregularity of the road

#regular oscilatory road input
#b=0.1
#om=10          # om=0.1, 0.2, 0.5, 1, 2, 3.
#xt=b*sin(om*t)


#irregular oscilatory road input with rising oscilation speed
b=0.1
om=1          # om=0.1, 0.2, 0.5, 1, 2, 3.
xt= b* sin(om* floor(t/5)* (t-5) )


# parameters of equations
#a1=F/m1
#a2=k/m1
#a3=c/m1
#a4=F/m2
#a5=k/m2
#a6=c/m2
#a7=kt/m2

# initial conditions
ics=[0,0,0,0]
```

```
# right sides of ODE
F1 = x3
F2 = x4
F3 = ((-k*(x1-x2)-c*(x3-x4))/m1)
F4 = ((k*(x1-x2)+c*(x3-x4)-kt*(x2-xt))/m2)

#F3 = (F-k*(x1-x2)-c*(x3-x4))/m1
#F4 = (-F+k*(x1-x2)+c*(x3-x4)-kt*(x2-x(t)))/m2

eqs = vector([F1,F2,F3,F4])

vrs=[x1,x2,x3,x4]
ics=[0,0,0,0]
tend=100
times = srange(0,tend,0.01)

# numerical solution of ODEs
sol = desolve_odeint(eqs, ics, times, vrs, ivar=t)
sol

[len(sol[:]),len(sol[1,:])]

sol[0,:]

sol[10000-1,:]

[sol[0,0], sol[1,0], sol[1,1], sol[2,0]]

sol[:,1]

# plotting results

times;

x1_sol = sol[:,0]
x2_sol = sol[:,1]
x3_sol = sol[:,2]
x4_sol = sol[:,3]

drg=plot(xt, (t, 0, tend),plot_points=1000,thickness=0.3 ,color='blue',figsize=5,
axes_labels=['t','x(t)']);drg

P1=list_plot(list(zip(times, x1_sol)),plotjoined=True, color='green',figsize=7, axes_labels=['t','x1'])
```

```
P2=list_plot(list(zip(times, x2_sol)),plotjoined=True, color='red',figsize=7,thickness=0.5,
axes_labels=['t','x2'])
P3=list_plot(list(zip(times, x3_sol)),plotjoined=True, color='black',figsize=7, axes_labels=['t','x3'])
P4=list_plot(list(zip(times, x4_sol)),plotjoined=True, color='blue',figsize=7, axes_labels=['t','x4'])

#movement of mass
P12=P1+P2
#mass speed
P34=P3+P4
#relation of road and unsprung mass
P2d= P2+drg
```

ZAŁĄCZNIK NR 2 – KOD ŹRÓDŁOWY PROGRAMU SYMULACJI DRUGIEGO MODELU

```
reset()
var('xu xs xc xh vu vs vc vh t')

#parameters of mechanical system

mh=20
mc=45
ms=241.65
mu=21.46
kb=45000
kc=20000
ks=21027.63
kt=102017.2
cb=1360
cc=1650
cs=1183.2
g=9.81

#the displacement input due to the surface irregularity of the road

b=0.1
om=2          # om=0.1, 0.2, 0.5, 1, 2, 3.
#x(t)= b*sin(om*floor(t/5)*(t-5) )
#x(t)= b*sin(t*om*sin(t))
#x(t) = b*(2*floor(om*t)-floor(2*om*t))+b
#x(t) = b+b*sign(t-om)
#b=0.1
#om=5          # om=0.1, 0.2, 0.5, 1, 2, 3.
#x(t) = b*sin((t/5 + 1)*om*t)
```

```
x(t) = b*sin(om*t)

freq(t) = ((t/5 + 1)*om)/6.2831853

#initial conditions
ics=[0,0,0,0,0,0,0,0]

#Right sides of ODE
F1 = vu
F2 = vs
F3 = vc
F4 = vh
F5 = ( -kt*( xu-x(t))+ks*( xs-xu ) +cs*( vs-vu ) )/mu -g
F6 = ( -ks*( xs-xu ) -cs*( vs-vu ) +kc*( xc-xs ) +cc*( vc-vs ) )/ms -g
F7 = ( -kc*( xc-xs ) -cc*( vc-vs ) +kb*( xh-xc ) +cb*( vh-vc ) )/mc -g
F8 = ( -kb*( xh-xc ) -cb*( vh-vc ) )/mc -g

eqs =vector([F1,F2,F3,F4,F5,F6,F7,F8])

vrs = [xu, xs, xc, xh, vu, vs, vc, vh]
ics=[0,0,0,0,0,0,0,0]
tend=40
times = srange(0,tend,0.01)

#numerical solution of ODEs
sol = desolve_odeint(eqs,ics,times,vrs,ivar=t)

#plotting results
x1_sol = sol[:,0]
x2_sol = sol[:,1]
x3_sol = sol[:,2]
x4_sol = sol[:,3]
x5_sol = sol[:,4]
x6_sol = sol[:,5]
x7_sol = sol[:,6]
x8_sol = sol[:,7]

PT = plot(x(t), (t, 0, tend), color='orange',figsize=5, axes_labels=['t','x(t)'])
PF = plot(freq(t), (t, 0, tend), color='orange',figsize=7, axes_labels=['t [s]','Częstotliwość [Hz]'])
PT.show()
PF.show()
```

```
P1=list_plot(list(zip(times, x1_sol)),plotjoined=True, color='green',figsize=7,ymax=0.5, ymin=-0.5,
axes_labels=['t','x1'])
P2=list_plot(list(zip(times, x2_sol)),plotjoined=True, color='red',figsize=7,ymax=0.5, ymin=-0.5,
axes_labels=['t','x2'])
P3=list_plot(list(zip(times, x3_sol)),plotjoined=True, color='black',figsize=7,ymax=0.5, ymin=-0.5,
axes_labels=['t','x3'])
P4=list_plot(list(zip(times, x4_sol)),plotjoined=True, color='blue',figsize=7,ymax=0.5, ymin=-0.5,
axes_labels=['t','x4'], linestyle='dotted')
P5=list_plot(list(zip(times, x5_sol)),plotjoined=True, color='green',figsize=7, axes_labels=['t','x5'])
P6=list_plot(list(zip(times, x6_sol)),plotjoined=True, color='red',figsize=7, axes_labels=['t','x6'])
P7=list_plot(list(zip(times, x7_sol)),plotjoined=True, color='black',figsize=7, axes_labels=['t','x7'])
P8=list_plot(list(zip(times, x8_sol)),plotjoined=True, color='blue',figsize=7, axes_labels=['t','x8'],
linestyle='dotted')

(PT+P1+P2).show()
(PT+P3+P4).show()
(P5+P6+P7+P8).show()
(P1+P2+P3+P4).show()

P12=P1+P2
P12.show(axes_labels=['t','x1,x2 (wykres 1)'])
```