



**AKADEMIA GÓRNICZO-HUTNICZA
im. Stanisława Staszica w Krakowie
WYDZIAŁ INŻYNIERII
MECHANICZNEJ I ROBOTYKI**

Praca dyplomowa magisterska

**Katarzyna Rugełło,
Bartłomiej Piwowarczyk**

Imię i nazwisko

Automatyka i Robotyka

Kierunek studiów

**Opracowanie aplikacji do symulacji propagacji
fal prowadzonych w prętach stalowych**

Temat pracy dyplomowej

prof. dr hab. inż. Tadeusz Stepinski

Promotor Pracy

.....

Ocena

Kraków, rok 2017/2018

Kraków, 1 września 2018

Imię i nazwisko: Katarzyna Rugełło
Nr albumu: 269475
Kierunek studiów: Automatyka i Robotyka
Profil dyplomowania: Automatyka i Metrologia

OŚWIADCZENIE

Uprzedzony o odpowiedzialności karnej na podstawie art. 115 ust 1 i 2 ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (tj. Dz. U. z 2006 r. Nr 90, poz. 631 z późn. zm.): „Kto przywłaszcza sobie autorstwo albo wprowadza w błąd co do autorstwa całości lub części cudzego utworu albo artystycznego wykonania, podlega grzywnie, karze ograniczenia wolności albo pozbawienia wolności do lat 3. Tej samej karze podlega, kto rozpowszechnia bez podania nazwiska lub pseudonimu twórcy cudzy utwór w wersji oryginalnej albo w postaci opracowania, artystyczne wykonanie albo publicznie zniekształca taki utwór, artystyczne wykonanie, fonogram, videogram lub nadanie”, a także uprzedzony o odpowiedzialności dyscyplinarnej na podstawie art. 211 ust. 1 ustawy z dnia 27 lipca 2005 r. Prawo o szkolnictwie wyższym (tj. Dz. U. z 2012 r. poz. 572, z późn. zm.) „Za naruszenie przepisów obowiązujących w uczelni oraz za czyny uchybiające godności student ponosi odpowiedzialność dyscyplinarną przed komisją dyscyplinarną albo przed sądem koleżeńskim samorządu studenckiego, zwanym dalej „sądem koleżeńskim”, oświadczam, że niniejszą pracę dyplomową wykonałem osobiście i samodzielnie i że nie korzystałem ze źródeł innych niż wymienione w pracy”.

.....
podpis dyplomanta

Kraków, 1 września 2018

Imię i nazwisko: Bartłomiej Piwowarczyk
Nr albumu: 269466
Kierunek studiów: Automatyka i Robotyka
Profil dyplomowania: Automatyka i Metrologia

OŚWIADCZENIE

Uprzedzony o odpowiedzialności karnej na podstawie art. 115 ust 1 i 2 ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (tj. Dz. U. z 2006 r. Nr 90, poz. 631 z późn. zm.): „Kto przywłaszcza sobie autorstwo albo wprowadza w błąd co do autorstwa całości lub części cudzego utworu albo artystycznego wykonania, podlega grzywnie, karze ograniczenia wolności albo pozbawienia wolności do lat 3. Tej samej karze podlega, kto rozpowszechnia bez podania nazwiska lub pseudonimu twórcy cudzy utwór w wersji oryginalnej albo w postaci opracowania, artystyczne wykonanie albo publicznie zniekształca taki utwór, artystyczne wykonanie, fonogram, videogram lub nadanie”, a także uprzedzony o odpowiedzialności dyscyplinarnej na podstawie art. 211 ust. 1 ustawy z dnia 27 lipca 2005 r. Prawo o szkolnictwie wyższym (tj. Dz. U. z 2012 r. poz. 572, z późn. zm.) „Za naruszenie przepisów obowiązujących w uczelni oraz za czyny uchybiające godności student ponosi odpowiedzialność dyscyplinarną przed komisją dyscyplinarną albo przed sądem koleżeńskim samorządu studenckiego, zwanym dalej „sądem koleżeńskim”, oświadczam, że niniejszą pracę dyplomową wykonałem osobiście i samodzielnie i że nie korzystałem ze źródeł innych niż wymienione w pracy”.

.....
podpis dyplomanta

Kraków, 1 września 2018

Imię i nazwisko: Katarzyna Rugełło
Nr. albumu: 269475
Kierunek studiów: Automatyka i Robotyka
Specjalność: Automatyka i Metrologia

OŚWIADCZENIE

Świadomy odpowiedzialności karnej za poświadczanie nieprawdy oświadczam, że niniejszą magisterską pracę dyplomową wykonałem osobiście i samodzielnie oraz nie korzystałem ze źródeł innych niż wymienione w pracy.

Jednocześnie oświadczam, że dokumentacja pracy nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 roku o prawie autorskim i prawach pokrewnych (Dz. U. z 2006 r. Nr 90 poz. 631 z późniejszymi zmianami) oraz dóbr osobistych chronionych prawem cywilnym. Nie zawiera ona również danych i informacji, które uzykałem w sposób niedozwolony. Wersja dokumentacji dołączona przeze mnie na nośniku elektronicznym jest w pełni zgodna z wydrukiem przedstawionym do recenzji.

Zaświadczam także, że niniejsza magisterska praca dyplomowa nie była wcześniej podstawą żadnej innej urzędowej procedury związanej z nadawaniem dyplomów wyższej uczelni lub tytułów zawodowych.

.....
podpis dyplomanta

Kraków, 1 września 2018

Imię i nazwisko: Bartłomiej Piwowarczyk

Nr. albumu: 269466

Kierunek studiów: Automatyka i Robotyka

Specjalność: Automatyka i Metrologia

OŚWIADCZENIE

Świadomy odpowiedzialności karnej za poświadczanie nieprawdy oświadczam, że niniejszą magisterską pracę dyplomową wykonałem osobiście i samodzielnie oraz nie korzystałem ze źródeł innych niż wymienione w pracy.

Jednocześnie oświadczam, że dokumentacja pracy nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 roku o prawie autorskim i prawach pokrewnych (Dz. U. z 2006 r. Nr 90 poz. 631 z późniejszymi zmianami) oraz dóbr osobistych chronionych prawem cywilnym. Nie zawiera ona również danych i informacji, które uzy skałem w sposób niedozwolony. Wersja dokumentacji dołączona przeze mnie na nośniku elektronicznym jest w pełni zgodna z wydrukiem przedstawionym do recenzji.

Zaświadczam także, że niniejsza magisterska praca dyplomowa nie była wcześniej podstawą żadnej innej urzędowej procedury związanej z nadawaniem dyplomów wyższej uczelni lub tytułów zawodowych.

.....

podpis dyplomanta

Kraków, 1 września 2018

Imię i nazwisko:	Katarzyna Rugełło
Adres korespondencyjny:	ul. Adama Mickiewicza 17/38, 38-400 Krosno
Temat pracy dyplomowej magisterskiej:	Opracowanie aplikacji do symulacji propagacji fal prowadzonych w prętach stalowych
Rok ukończenia:	2018
Nr. albumu:	269475
Kierunek studiów:	Automatyka i Robotyka
Specjalność:	Automatyka i Metrologia

OŚWIADCZENIE

Niniejszym oświadczam, że zachowując moje prawa autorskie, udzielam Akademii Górniczo-Hutniczej im. S. Staszica w Krakowie nieograniczonej w czasie nieodpłatnej licencji niewyłącznej do korzystania z przedstawionej dokumentacji magisterskiej pracy dyplomowej, w zakresie publicznego udostępniania i rozpowszechniania w wersji drukowanej i elektronicznej¹.

Publikacja ta może nastąpić po ewentualnym zgłoszeniu do ochrony prawnej wynalazków, wzorów użytkowych, wzorów przemysłowych będących wynikiem pracy inżynierskiej².

Kraków,

data *podpis dyplomanta*

¹Na podstawie Ustawy z dnia 27 lipca 2005 r. o prawie o szkolnictwie wyższym (Dz.U. z 2005 Nr 164, poz. 1365) Art. 239 oraz Ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (Dz.U. 2000 r. Nr 80, poz. 904, z późn. zm.) Art. 15a: "Uczelni w rozumieniu przepisów o szkolnictwie wyższym przysługuje pierwszeństwo w opublikowaniu pracy dyplomowej studenta. Jeżeli uczelnia nie opublikowała pracy dyplomowej w ciągu 6 miesięcy od jej obrony, student, który ją przygotował, może ją opublikować, chyba że praca dyplomowa jest częścią utworu zbiorowego."

²Ustawa z dnia 30 czerwca 2000 r. – prawo własności przemysłowej (Dz.U. z 2003 r. Nr 119, poz. 1117, z późn. zm.), a także rozporządzenie Prezesa Rady Ministrów z dnia 17 września 2001 r. w sprawie dokonywania i rozpatrywania zgłoszeń wynalazków i wzorów użytkowych (Dz.U. z 2001 r. Nr 102, poz. 1119 oraz z 2005 r. Nr 109, poz. 910).

Kraków, 1 września 2018

Imię i nazwisko:	Bartłomiej Piwowarczyk
Adres korespondencyjny:	Stryszowa 27, 32-420 Gdów
Temat pracy dyplomowej magisterskiej:	Opracowanie aplikacji do symulacji propagacji fal prowadzonych w prętach stalowych
Rok ukończenia:	2018
Nr. albumu:	269466
Kierunek studiów:	Automatyka i Robotyka
Specjalność:	Automatyka i Metrologia

OŚWIADCZENIE

Niniejszym oświadczam, że zachowując moje prawa autorskie, udzielam Akademii Górnictwo-Hutniczej im. S. Staszica w Krakowie nieograniczonej w czasie nieodpłatnej licencji niewyłącznej do korzystania z przedstawionej dokumentacji magisterskiej pracy dyplomowej, w zakresie publicznego udostępniania i rozpowszechniania w wersji drukowanej i elektronicznej³.

Publikacja ta może nastąpić po ewentualnym zgłoszeniu do ochrony prawnej wynalazków, wzorów użytkowych, wzorów przemysłowych będących wynikiem pracy inżynierskiej⁴.

Kraków,

data *podpis dyplomanta*

³Na podstawie Ustawy z dnia 27 lipca 2005 r. o prawie o szkolnictwie wyższym (Dz.U. z 2005 Nr 164, poz. 1365) Art. 239 oraz Ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (Dz.U. 2000 r. Nr 80, poz. 904, z późn. zm.) Art. 15a: "Uczelni w rozumieniu przepisów o szkolnictwie wyższym przysługuje pierwszeństwo w opublikowaniu pracy dyplomowej studenta. Jeżeli uczelnia nie opublikowała pracy dyplomowej w ciągu 6 miesięcy od jej obrony, student, który ją przygotował, może ją opublikować, chyba że praca dyplomowa jest częścią utworu zbiorowego."

⁴Ustawa z dnia 30 czerwca 2000 r. – prawo własności przemysłowej (Dz.U. z 2003 r. Nr 119, poz. 1117, z późn. zm.), a także rozporządzenie Prezesa Rady Ministrów z dnia 17 września 2001 r. w sprawie dokonywania i rozpatrywania zgłoszeń wynalazków i wzorów użytkowych (Dz.U. z 2001 r. Nr 102, poz. 1119 oraz z 2005 r. Nr 109, poz. 910).

Kraków, 1 września 2018

AKADEMIA GÓRNICZO-HUTNICZA WYDZIAŁ INŻYNIERII MECHANICZNEJ I ROBOTYKI

TEMATYKA PRACY DYPLOMOWEJ MAGISTERSKIEJ

dla studenta II roku studiów stacjonarnych

**Katarzyna Rugełło,
Bartłomiej Piwowarczyk
*imię i nazwisko studenta***

TEMAT PRACY DYPLOMOWEJ MAGISTERSKIEJ: Opracowanie aplikacji do symulacji propagacji fal prowadzonych w prętach stalowych

Promotor pracy: prof. dr hab. inż. Tadeusz Stepinski

Recenzent pracy: dr hab. inż. Paweł Paćko podpis dziekana

PLAN PRACY DYPLOMOWEJ:

1. Omówienie tematu pracy i sposobu realizacji z promotorem.
 2. Zebranie i opracowanie literatury dotyczącej tematu pracy.
 3. Zebranie i opracowanie wyników badań.
 4. Analiza wyników badań, ich omówienie i zatwierdzenie przez promotorą.
 5. Opracowanie redakcyjne.

Kraków,
data podpis dyplomanta

TERMIN ODDANIA DO DZIEKANATU: **20** r.

.....
podpis promotora

Kraków, 1 września 2018

Akademia Górnictwo-Hutnicza im. Stanisława Staszica
Wydział Inżynierii Mechanicznej i Robotyki

Kierunek: Automatyka i Robotyka

Profil dyplomowania: Automatyka i Metrologia

Katarzyna Rugełło

Bartłomiej Piwowarczyk

Praca dyplomowa magisterska

Opracowanie aplikacji do symulacji propagacji fal prowadzonych w prętach stalowych

Opiekun: prof. dr hab. inż. Tadeusz Stepiński

STRESZCZENIE

Celem niniejszej pracy dyplomowej było stworzenie aplikacji, pozwalającej na symulację badania metalowych prętów przy pomocy fal prowadzonych. Do tworzenia aplikacji wybrany został obiektowy język programowania Python w wersji 3.x. Stworzona aplikacja pozwala użytkownikowi na wybór sygnału wejściowego, wyznaczenie modelu matematycznego symulowanego pręta, wyznaczenie sygnału zwońskiego z uwzględnieniem parametrów pręta, wyznaczenie krzywych dyspersji dla pręta o zadanych właściwościach materiałowych a także kompensację dyspersji sygnału wyjściowego trzema wybranymi metodami. Praca składa się z sześciu rozdziałów. Pierwszy z nich stanowi wstęp do niniejszej pracy. W tym rozdziale zdefiniowany został cel pracy oraz przedstawiona geneza powstania wybranego tematu. Rozdział drugi stanowi wstęp teoretyczny, zawierający podstawowe zagadnienia propagacji fal w ośrodku sprężystym. Omówione w nim zostały zagadnienia związane z liniową teorią sprężystości, zjawiskiem dyspersji a także rodzajem fal sprężystych. W kolejnym rozdziale zaprezentowane zostały zagadnienia związane z metodą elementów skończonych. Omówione zostały wszystkie etapy tworzenia solvera zaimplementowanego w ramach tej pracy. Rozdział czwarty poświęcony jest zjawisku dyspersji, oraz metodom kompensacji tego zjawiska. W rozdziale tym zostały szczegółowo omówione trzy wybrane metody kompensacji. Omówiona została ich implementacja numeryczna oraz zaprezentowane wyniki uzyskane drogą symulacji w stworzonej aplikacji. Rozdział kolejny stanowi dokumentację kodu stworzonej aplikacji. Zostały w nim zawarte informacje pozwalające na instalację oraz konfigurację środowiska, szczegółowe opisy stworzonych funkcji, oraz przykłady użycia ich w kontekście symulacji przy pomocy aplikacji. Dodatkowym elementem tego rozdziału jest opis stworzonego w ramach pracy graficznego interfejsu użytkownika wraz z przykładem przeprowadzenia symulacji przy jego pomocy. Ostatni rozdział stanowi podsumowanie całej pracy.

Cracow, September 1, 2018

AGH University of Science and Technology
Faculty of Mechanical Engineering and Robotics

Field of Study: Automatics and Robotics

Specialisations: Automation and Metrology

Katarzyna Rugełło

Bartłomiej Piwowarczyk

Master Diploma Thesis

Development of software application for the simulation of guided wave in steel bars

Supervisor: prof. dr hab. inż. Tadeusz Stepinski

SUMMARY

[Summary text]

Składamy najserdeczniejsze podziękowania dla Promotora pracy, prof. dr hab. inż. Tadeusza Stepińskiego, oraz recenzenta dr hab. inż. Pawła Paćko, za wskazówki udzielone nam podczas pisania pracy, za cenne uwagi przekazywane w dużym spokoju, za wszechstronną pomoc, która była dużym wsparciem podczas pisania tej pracy, a także za cały poświęcony czas.

Spis treści

1. Wstęp.....	11
1.1. Cele pracy.....	12
1.2. Plan pracy	13
2. Wybrane zagadnienia propagacji fal w ośrodku sprężystym.....	14
BARTŁOMIEJ PIWOWARCZYK	
2.1. Liniowa teoria sprężystości	14
2.1.1. Naprężenie	14
2.1.2. Odkształcenie i prawko Hooke'a	15
2.2. Rodzaje fal sprężystych.....	17
2.2.1. Fale podłużne	17
2.2.2. Fale poprzeczne.....	18
2.2.3. Fale Rayleigha i Löva.....	18
2.2.4. Fale Lamba.....	19
2.3. Zjawisko dyspersji	20
2.3.1. Prędkość fazowa i grupowa.....	22
2.3.2. Analityczne wyznaczanie krzywych dyspersji.....	24
2.3.3. Numeryczne wyznaczanie krzywych dyspersji.....	28
2.3.4. Eksperymentalne wyznaczanie krzywych dyspersji	35
3. Wykorzystywane zagadnienia metody elementów skończonych	37
BARTŁOMIEJ PIWOWARCZYK	
3.1. Rozwiązywanie zadań z pomocą MES.....	37
3.2. Funkcje kształtu.....	38
3.3. Wyznaczanie macierzy sztywności i mas elementów	40
3.4. Agregacja globalnych macierzy mas i sztywności	44
3.5. Rozwiązywanie wyznaczonego równania macierzowego.....	47
3.6. Zbieżność metody i błędy rozwiązania	48
4. Metody kompensacji dyspersji.....	51
KATARZYNA RUGIELŁO	
4.1. Cel kompensacji	51
4.1.1. Zastosowanie fal Lamba w nieniszczacych testach.....	51
4.1.2. Sygnał stosowany w symulacji.....	54
4.2. Agregacja krzywych dyspersji uzyskanych z zaimplementowanego solvera ..	56
4.2.1. Cel agregacji	56
4.2.2. Algorytm agregacji	56

4.3. Metoda odwracania sygnału w czasie	58
4.3.1. Podstawy teoretyczne	58
4.3.2. Implementacja numeryczna	60
4.3.3. Wybrane wyniki z symulacji	61
4.4. Metoda mapowania liniowego przy pomocy rozwinięcia w szereg Taylora ..	67
4.4.1. Podstawy teoretyczne	67
4.4.2. Implementacja numeryczna	70
4.4.3. Wybrane wyniki symulacji	73
4.5. Metoda mapowania sygnału z dziedziny czasu na dziedzinę odległości	76
4.5.1. Podstawy teoretyczne	76
4.5.2. Implementacja numeryczna	79
4.5.3. Wybrane wyniki symulacji	81
4.5.4. Przykład zastosowania na podstawie symulacji wybranego pręta	84
4.6. Porównanie opracowanych metod kompensacji	88
4.6.1. Metoda odwracania sygnału w czasie	88
4.6.2. Metoda mapowania liniowego przy pomocy rozwinięcia w szereg Taylora	89
4.6.3. Metoda mapowania sygnału z dziedziny czasu na dziedzinę odległości	90
5. Aplikacja do obliczeń numerycznych	91
KATARZYNA RUGIELŁO	
BARTŁOMIEJ PIWOWARCZYK	
5.1. Instalacja i konfiguracja środowiska	91
BARTŁOMIEJ PIWOWARCZYK	
5.1.1. Python	91
5.1.2. PyCharm	96
5.2. Potrzebne biblioteki	97
KATARZYNA RUGIELŁO	
5.3. Obliczenia MES i wyznaczanie krzywych dyspersji oraz wzbudzalności	102
BARTŁOMIEJ PIWOWARCZYK	
5.3.1. Elementy czworościenne	103
5.3.2. Elementy sześciennne	108
5.3.3. Pozostałe moduły katalogu MES_dir	111
5.3.4. Moduł main	114
5.3.5. Wyniki - krzywe dyspersji	117
5.4. Segregacja krzywych dyspersji	119
KATARZYNA RUGIELŁO	
5.5. Metody kompensacji dyspersji - Katarzyna Rugełło	124
KATARZYNA RUGIELŁO	
5.6. Graficzny interfejs użytkownika - Katarzyna Rugełło	128
KATARZYNA RUGIELŁO	
5.6.1. Opis dostępnych funkcji	128

5.6.2. Przykład zastosowania do wygenerowania danych dla zadanego pręta.....	136
6. Podsumowanie	139
KATARZYNA RUGIEŁŁO	
Bibliografia	141

1. Wstęp

W dzisiejszym czasach z każdej strony otaczają nas różnego rodzaj konstrukcje mechaniczne. Wysokie koszty materiałów skłaniają inżynierów do szukania sposobów na mniejsze ich zużycie oraz zwiększenie żywotności konstrukcji. Dodatkowo pojawia się problem bieżącego sprawdzania stanu konstrukcji w trakcie jej eksploatacji. Jednym ze sposobów badania stanu konstrukcji jest zastosowanie fal mechanicznych. Fale te stały się przedmiotem zainteresowania badaczy już w XIX w. i zaczęły powstawać pierwsze prace takich inżynierów i naukowców jak Poisson, Stokes, Rayleigh. Badania te stawały się popularniejsze i w kolejnych dziesięcioleciach powstawały nowe poważne prace na ten temat. W ostatnich latach pojawiło się duże zainteresowanie wokół tematu fal prowadzonych, które dają duże możliwości badania konstrukcji, ale ich opis jest trudny i najczęściej problemy z nimi związane nie są możliwe do rozwiązania w sposób analityczny.

Na gruncie badań nad falami wyłoniły się dwa obszary ich zastosowań tj. badania nieniszczące oraz monitorowanie zdrowia konstrukcji. W pierwszym przypadku celem jest wyszukiwanie uszkodzeń konstrukcji przy pomocy wyspecjalizowanej, przenośnej aparatury poprzez korzystanie z wysokoczęstotliwościowych fal (fal objętościowych). Konstrukcję bada się w różnych punktach, szukając uszkodzenia w bliskim otoczeniu punktu badawczego. Utrudnieniem jest brak odniesienia do wyników poprawnie działającej konstrukcji. Powoduje to, że badania może być bardzo czasochłonne, co podnosi dodatkowo koszty.

Monitorowanie zdrowia konstrukcji skupia się wokół sprawdzania przydatności konstrukcji do użycia i pozostałego czasu, w którym może być ona używana. Możliwe jest to dzięki badaniu z jednego punktu dużego obszaru konstrukcji za pomocą fal prowadzonych. O takich falach mówimy kiedy ich długość staje się porównywalna do wymiarów geometrycznych konstrukcji. Zastosowanie takiego rozwiązania pozwala zautomatyzować proces testowania i zmniejszyć koszta. Aby to było możliwe niezbędne są dane zebrane na prawidłowo działającej konstrukcji. Dodatkowo wymagane jest zastosowanie metod kompensacji zjawisk fizycznych jak dyspersja. Na rysunku 1.1 przedstawiony jest sposób badania konstrukcji przy pomocy fal objętościowych oraz fal prowadzonych.



Rys. 1.1. Przykład badania przy pomocy fal objętościowych (góry rysunek) oraz fal prowadzonych (dolny rysunek) [1]

1.1. Cele pracy

Praca skupia się wokół symulacji badania metalowych prętów przy pomocy fal prowadzonych. Symulacja ma za zadanie wyznaczyć sygnał zwrotny, dla nadanego sygnału za pomocą np. przetwornika piezoelektrycznego, w przypadku badania puls-echo. Fale prowadzone podlegają dyspersji, więc sygnał powrotny jest silnie zniekształcony w stosunku do nadanego. Algorytmy, które pozwolą skompensować dyspersję są kolejnym aspektem branym pod uwagę przy symulowaniu przebiegu sygnału.

Pomysł na tego typu symulację wziął się z problemu badania kotw stropowych przedstawionego w [2]. Jeśli wykorzystamy metalowe pręty aby wzmacnić strop tunelu i mamy do nich dostęp wyłącznie od strony czoła, to możliwości sprawdzania stanu tych prętów jest mocno ograniczony. Rysunek 4.1 przedstawia widok tunelu i kotw stropowych. Sposobem na diagnostykę kotw, może być właśnie zastosowanie fal prowadzonych.

Aby dać możliwość symulacji badania pręta, program musi zapewniać następujące funkcjonalności:

1. Możliwość wyboru sygnału wejściowego
2. Możliwość wyznaczenia modelu matematycznego pręta
3. Możliwość wyznaczenia sygnału zwrotnego z uwzględnieniem parametrów pręta
4. Możliwość wyznaczenia krzywych dyspersji dla pręta o zadanych właściwościach materiałowych
5. Możliwość kompensacji dyspersji sygnału wyjściowego kilkoma, wybranymi metodami.

1.2. Plan pracy

Praca podzielona jest na 6 rozdziałów. W rozdziale 1 przedstawiono krótki wstęp oraz założenia projektowe. W rozdziale 2 opisane są podstawowe pojęcia dotyczące propagacji fal w środowisku sprężystym. Podano definicję naprężenia oraz odkształcenia oraz opisano rodzaje fal sprężystych. Następnie przytoczone są informację o zjawisku dyspersji. Opisano przykład kiedy możliwe jest analityczne wyznaczenie krzywych dyspersji oraz przykład gdzie nie jest to już możliwe. Na kolejnych stronach znajdują się metody numerycznego wyznaczania tych krzywych oraz możliwości wyznaczenia ich w badaniach doświadczalnych. Rozdział 3 przedstawia zagadnienia metody elementów skończonych, które zostały wykorzystane w aplikacji do obliczania modelu pręta. Opisane są kolejno funkcje kształtu, sposoby ich wyznaczania, obliczanie macierzy mas i sztywności elementu oraz agregacja macierzy globalnych. Dla wyznaczonego modelu przedstawiono kilka sposobów wyznaczenia rozwiązania w postaci przemieszczeń, oraz sposoby estymacji błędów rozwiązania i sprawdzenia czy rozwiązanie jest zbieżne. W rozdziale 4 przedstawiono trzy wybrane metody kompensacji dyspersji, które zostały zaimplementowane w programie. Każda metoda opisana została zarówno od strony teoretycznej jak i praktycznej implementacji numerycznej. Ostatni podrzdział stanowi podsumowanie, porównujące wszystkie omawiane metody. Opisane są tam wady i zalety każdej z nich oraz ich ograniczenia. Rozdział 5 stanowi opis zbudowanej aplikacji oraz opis środowiska programistycznego. Początek rozdziału zawiera instrukcję instalacji i konfiguracji interpreta Python programu PyCharm który został wykorzystany w projekcie. Znajduje się tam również lista wszystkich bibliotek niezbędnych do uruchomienia stworzonej aplikacji. Dalsza część rozdziału zawiera pełny opis działania programu wraz z przytoczonymi przykładami wykorzystania. Opisane są kolejne kroki obliczeń dwóch głównych modułów aplikacji tj. budowania modelu wraz z wyznaczaniem krzywych dyspersji oraz kompensowania dyspersji z użyciem wybranej metody. Dodatkowym elementem jest możliwość korzystania z zaimplementowanego graficznego interfejsu użytkownika. Rozdział 6 jest podsumowaniem efektów pracy przy projekcie. Opisano stan zrealizowania założeń oraz podano możliwości rozwoju projektu.

2. Wybrane zagadnienia propagacji fal w ośrodku sprężystym

BARTŁOMIEJ PIWOWARCZYK

W rozdziale przedstawiono zagadnienia dotyczące mechaniki ośrodka sprężystego wykorzystywane w projekcie. Założenia liniowej teorii sprężystości oraz definicje naprężenia i odkształcenia przytaczane są w [3]. Opis typów fal sprężystych znaleźć można w [1] i [4]. Zjawisko dyspersji opisane jest między innymi w [1], [5], [6], [7], [8]. Przetworniki piezoelektryczne były analizowane na podstawie danych dostępnych na stronie jednego z producentów [13].

2.1. Liniowa teoria sprężystości

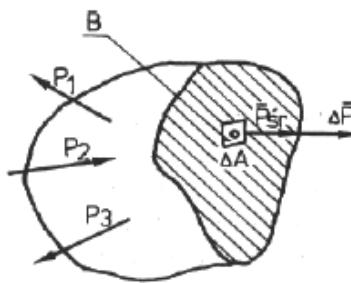
Liniowa teoria sprężystości jest mechaniką ciała stałego opartą na następujących założeniach:

- ciało jest wypełnione materią w sposób ciągły zarówno przed jak i po odkształceniu
- odkształcenia i przemieszczenia są bardzo małe
- spełniona jest zasada superpozycji
- ośrodek zachowuje się zgodnie z prawem Hooke'a
- siły działają na ciało w taki sam sposób przed odkształceniem jak i po odkształceniu

2.1.1. Naprężenie

Naprężenie można zdefiniować jako miara sił wewnętrznych ciała w punkcie. Weźmy ciało przedstawione na rysunku 2.1, na które działają siły zewnętrzne P_1 i P_2 . W płaszczyźnie przekroju wybieramy punkt B, w którego otoczeniu określamy pole dA . Stosunek sił jakimi oddziałują na siebie połówki ciała w tym punkcie przekroju, do pola dA , nazywamy naprężeniem ciała w punkcie B. Kierunek naprężenia będzie zgodny z kierunkiem działania siły przekrojowej w punkcie.

Stana naprężenia w punkcie B oznacza ogólną naprężenie, które otrzymamy dla wszystkich możliwych przekrojów ciała przez ten punkt. W przypadku trójwymiarowego stanu naprężenia, dla każdego przekroju wektor naprężenia będzie miał inny kierunek. Stan



Rys. 2.1. Ciało pod wpływem sił zewnętrznych [3]

naprężenia można opisać przy pomocy tensora naprężzeń, dla układu kartezjańskiego danego wzorem:

$$\sigma = \begin{bmatrix} \sigma_{xx} & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \sigma_{yy} & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \sigma_{zz} \end{bmatrix} \quad (2.1)$$

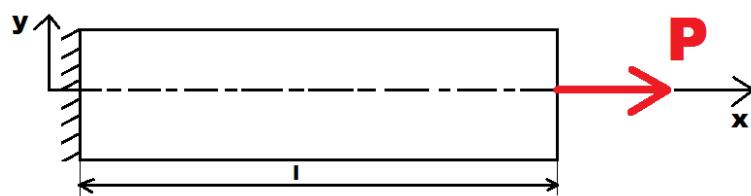
gdzie

σ_{ii} – naprężenie normalne, $i=(x, y, z)$

τ_{ij} – naprężenie styczne, $i,j=(x, y, z)$, $i \neq j$, $\tau_{ij} = \tau_{ji}$

2.1.2. Odkształcenie i prawko Hooke'a

Pod wpływem naprężzeń ciało sprężyste ulega odkształceniu. Możemy wyróżnić odkształcenia liniowe ε_i oraz odkształcenia postaciowe γ_{ij} . Najprostszym przypadkiem odkształcenia liniowego jest rozciąganie pręta. Poniżej znajduje się rysunek pręta rozciąganej siłą P .



Rys. 2.2. Rozciąganie pręta

W takim przypadku odkształceniem wzdłużnym będziemy nazywać stosunek wydłużenia pręta do jego początkowej długości:

$$\varepsilon_x = \frac{\Delta l}{l}. \quad (2.2)$$

Odkształceniem poprzecznym(postaciowym) nazywać będziemy stosunek zmiany średnicy przekroju do jego początkowej średnicy:

$$\varepsilon_y = \frac{\Delta d}{d}. \quad (2.3)$$

Odkształcenia te związane są zależnością:

$$\varepsilon_x = -\nu \varepsilon_y \quad (2.4)$$

gdzie

ν – współczynnik Poissona.

Współczynnik Poissona jest wielkością bezwymiarową. Określa on sposób w jaki odkształca się ciało i przyjmuje wartości z przedziału [-1, 1]. Dla popularnych w mechanice stopów metali przyjmuje zwykle wartości z przedziału [0.2, 0.4]. Wartości ujemne przyjmuje dla tak zwanych materiałów odwrotnych, które pod wpływem naprężenia zwiększą swoją objętość.

Dla przypadku prostego rozciągania zachodzi jeszcze jedna zależność. Opisuje ona związek pomiędzy naprężeniem i odkształceniem i nazywana jest prawem Hooke'a.

$$\sigma = E \varepsilon \quad (2.5)$$

gdzie

E – moduł Younga.

Moduł Younga określa zależność odkształcenia liniowego i przyłożonego naprężenia. Jednostką moduły Younga jest paskal, a wartości podaje się w gigapasklach (GPa). Wartości dla stali wynoszą około 200 GPa, a dla aluminium około 70 GPa.

W przypadku trójwymiarowego rozkładu oksztalceń stosuje się zapis tensorowy. Tensor odkształcenia znajduje się poniżej:

$$\boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_{xx} & \gamma_{xy} & \gamma_{xz} \\ \gamma_{yx} & \varepsilon_{yy} & \gamma_{yz} \\ \gamma_{zx} & \gamma_{zy} & \varepsilon_{zz} \end{bmatrix} \quad (2.6)$$

gdzie

ε_{ii} – odkształcenie liniowe, $i=(x, y, z)$

γ_{ij} – odkształcenie postaciowe, $i,j=(x, y, z)$, $i \neq j$, $\tau_{ij} = \tau_{ji}$

Dla takiego przypadku prawo Hooke'a przybiera bardziej skomplikowaną postać:

$$\varepsilon_{xx} = \frac{1}{E}(\sigma_{xx} - \nu(\sigma_{yy} + \sigma_{zz})), \gamma_{xy} = \frac{\tau_{xy}}{G} \quad (2.7)$$

$$\varepsilon_y y = \frac{1}{E}(\sigma_y y - \nu(\sigma_x x + \sigma_z z)), \gamma_{xz} = \frac{\tau_{xz}}{G} \quad (2.8)$$

$$\varepsilon_z z = \frac{1}{E}(\sigma_z z - \nu(\sigma_x x + \sigma_y y)), \gamma_{yz} = \frac{\tau_{yz}}{G} \quad (2.9)$$

G – moduł Kirchhoffa

Moduł Kirchhoffa opisuje zależność odkształcenia postaciowego od naprężenia stycznego występującego w materiale. Jednostką tego współczynnika jest pascal, a typowymi wartościami są dla stali 80 Gpa, dla aluminium 25,5 GPa.

Opisane wcześniej stałe materiałowe są powiązane równaniem:

$$E = 2G(\nu + 1) \quad (2.10)$$

2.2. Rodzaje fal sprężystych

Fale można podzielić ze względu na sposób w jaki propagują w materiale. Mogą to być fale podłużne lub poprzeczne. Fale takie propagują jeśli długość fali jest mniejsza lub bliska wymiarom ośrodka. Fale o długości przekraczającej wyraźnie przynajmniej jeden z wymiarów falowodu nazywamy falami prowadzonymi i mają one zupełnie inne własności. W takim przypadku występują dodatkowe rodzaje fal jak fale Rayleigha czy Lamba. Fala Rayleigha propagują na powierzchniach, stąd ośrodek musi być ograniczony przynajmniej jedną płaszczyzną. Fala Lamba propagują na strukturach cienkościennych.

2.2.1. Fale podłużne

Fale podłużne to fale, w których kierunek propagacji jest równoległy z kierunkiem drgania cząstek. Tego typu fale mogą rozchodzić się w każdym ośrodku materialnym. Przykładem fali podłużnej jest fala dźwiękowa rozchodząca się w powietrzu, pokazana na rysunku 2.3. Prędkość fali podłużnej w nieograniczonym ośrodku zależy od parametrów materiałowych ośrodka i dana jest wzorem:

$$c_L = \sqrt{\frac{\lambda + 2\mu}{\rho}} \quad (2.11)$$

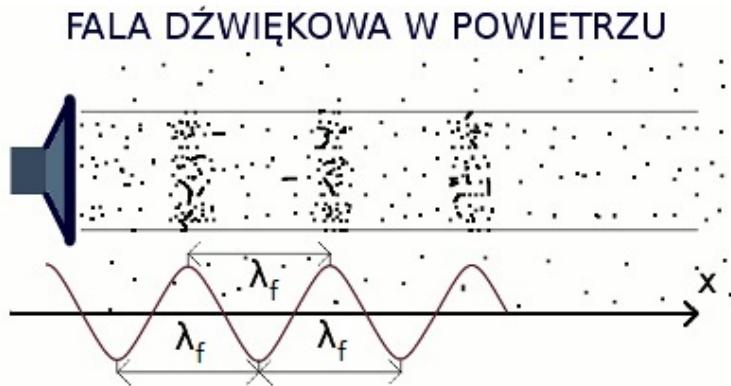
gdzie

c_L – prędkość fali podłużnej

λ, μ – stałe Lamégo

ρ – gęstość ośrodka

Długością fali λ_f nazywamy odległość między dwoma maksimami (lub minimami), które oznaczają maksymalne zagęszczenie (rozrzedzenie) cząstek w materiale.



Rys. 2.3. Przykład fali podłużnej [14]

2.2.2. Fale poprzeczne

Fale poprzeczne propagują w kierunku prostopadłym do kierunku drgań cząstek. Przykładami takich fal są fale elektromagnetyczne, fale propagacji naprężeń w materiale stałym, czy fala na zamocowanym jednostronnie sznurze, pokazana na rysunku 2.4. Prędkość fali poprzecznej w nieograniczonym ośrodku wyraża się wzorem:

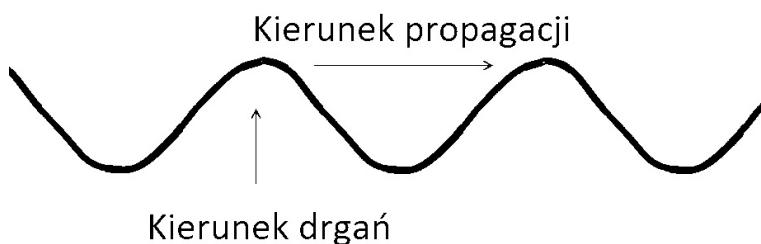
$$c_T = \sqrt{\frac{\mu}{\rho}} \quad (2.12)$$

gdzie

c_T – prędkość fali poprzecznej

μ – stała Lamégo

ρ – gęstość ośrodka



Rys. 2.4. Przykład fali poprzecznej

2.2.3. Fale Rayleigha i Löva

Opisane wcześniej fale propagują w nieograniczonych mediach. Fale Rayleigha oraz Löva są falami propagującymi w obszarze powierzchni ciał stałych. Dla fal Rayleigha drgania cząstek odbywają się równolegle oraz prostopadle do kierunku rozchodzenia się fali, zataczając elipsy. Kierunek prostopadły jest normalny do płaszczyzny, na której fala

propaguje. Prędkość fal Rayleigha dla metalu można wyrazić za pomocą współczynnika Poissona i prędkości fal poprzecznych:

$$c_R = \frac{0.87 + 1.12 \cdot \nu}{1 + \nu} \cdot c_T \quad (2.13)$$

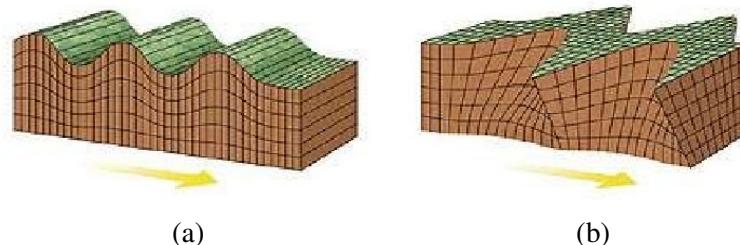
gdzie

c_R – prędkość fali Rayleigha

c_T – prędkość fali poprzecznej

ν – współczynnik Poissona

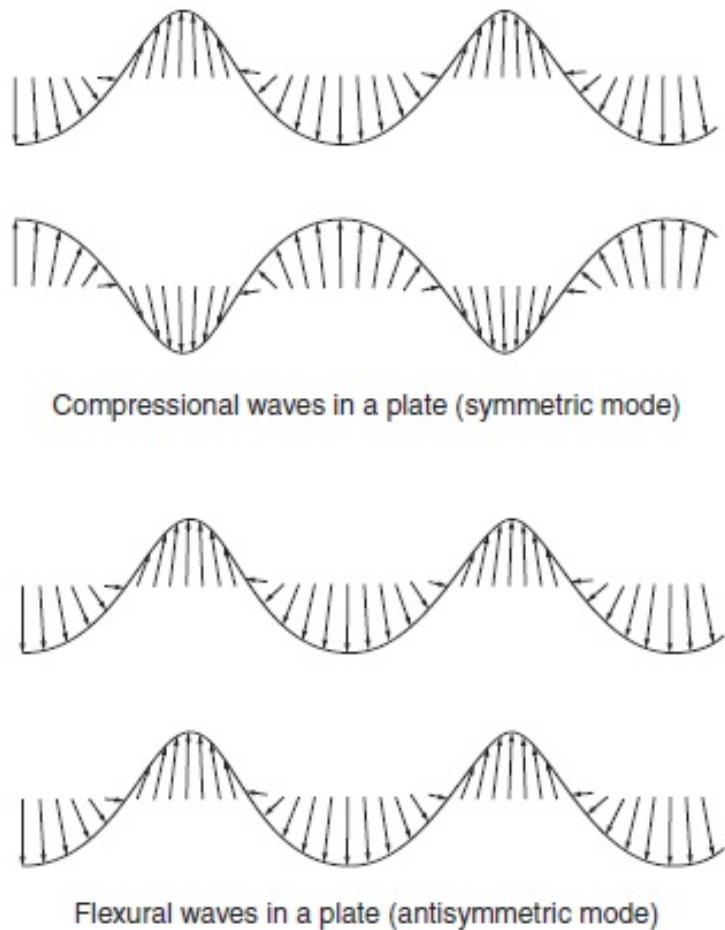
Fale Löva to fale, w których drgania zachodzą w kierunku prostopadłym do kierunku, ale równolegle do płaszczyzny propagacji. Takie fale są wykorzystywane przy badaniu układów wielowarstwowych. Są silnie dyspersyjne, co oznacza, że ich prędkość jest funkcją częstotliwości. Fala Rayleigha i Löva są zilustrowane na rysunku 2.5.



Rys. 2.5. Fale powierzchniowe: (a) fala Rayleigha, (b) fala Löva [15]

2.2.4. Fale Lamba

Ważnym typem fal, z racji na szerokie zastosowanie, są fale Lamba. Propagują one na cienkościennych elementach jak płyty, czy rury. Tego typu fale powstają na skutek złożenia dwóch fal Rayleigha, propagujących na płaszczyznach po obu stronach obiektu. Fale Lamba można podzielić na fale symetryczne, kiedy obie fale składowe propagują w tej samej fazie i antysymetryczne kiedy propagują w przeciwnych fazach. Prędkość fal Lamba zależna jest od częstotliwości, ze względu na dyspersyjny charakter. Rysunek 2.6 przedstawia podstaci fal Lamba.



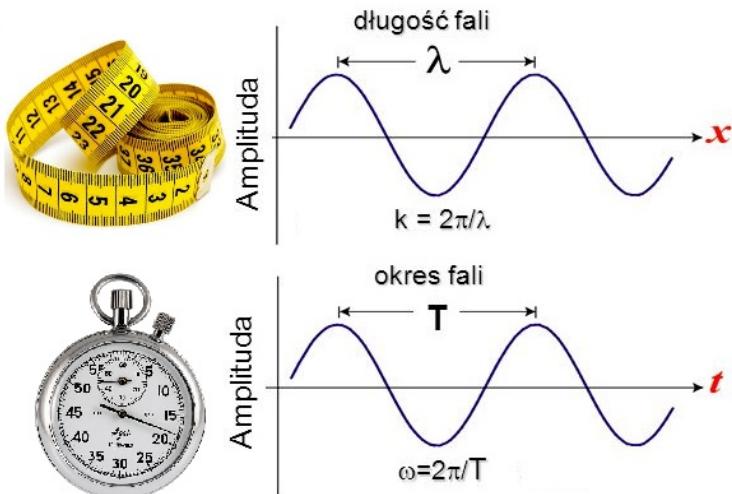
Rys. 2.6. Fale Lamba [1]

2.3. Zjawisko dyspersji

Dispersja jest zjawiskiem zależności prędkości propagacji fali od jej częstotliwości. Częstotliwością fali nazywamy liczbę pełnych przebiegów (np. od maximum do maximum) w jednostce czasu. Jednostką częstotliwości jest herc (Hz). Wielkością równie często używaną jest częstość kołowa dana wzorem $\omega = 2\pi f$. Jej jednostką jest radian na sekundę [rad/s]. Odpowiednikiem tego parametru dla wymiarów geometrycznych jest liczba falowa. Określa ona jak wiele długości fali zawiera fala w jednostce odległości. Jednostką liczby falowej jest radian na metr [rad/m].

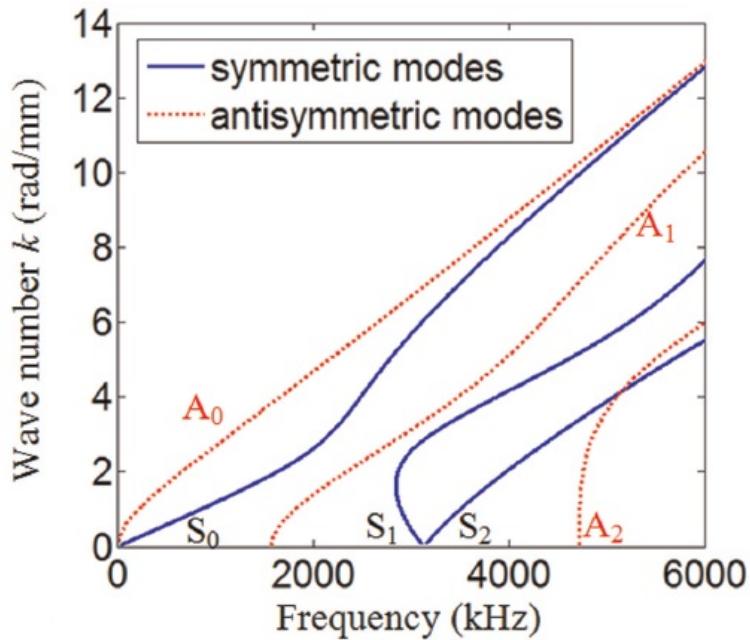
Kiedy mówimy o krzywych dyspersji, możemy mieć na myśli zależności liczby falowej od częstotliwości, prędkości fazowej od częstotliwości lub prędkości grupowej od częstotliwości. Na rysunku 2.8 znajdują się przykładowe charakterystyki $k(\omega)$ postaci fal Lamba dla aluminiowej płyty o grubości 1mm.

Na wykresach widać jedną rzeczą, ważną dla zastosowań fal prowadzonych - są one multimodalne. Z wykresów wynika, że w najkorzystniejszym do analizy przypadku propagują dwa mody, jeden symetryczny i jeden antysymetryczny. Wraz ze wzrostem



Rys. 2.7. Częstotliwość ω i liczba falowa k

częstotliwości propagującej fali pojawiają się dodatkowe jej postaci, co znacznie utrudnia analizę danych z przebiegów. Jednym ze sposobów ograniczania liczby modów, jest wprowadzanie wymuszeń o niskich częstotliwościach.

Rys. 2.8. Krzywe $k(\omega)$ [7]

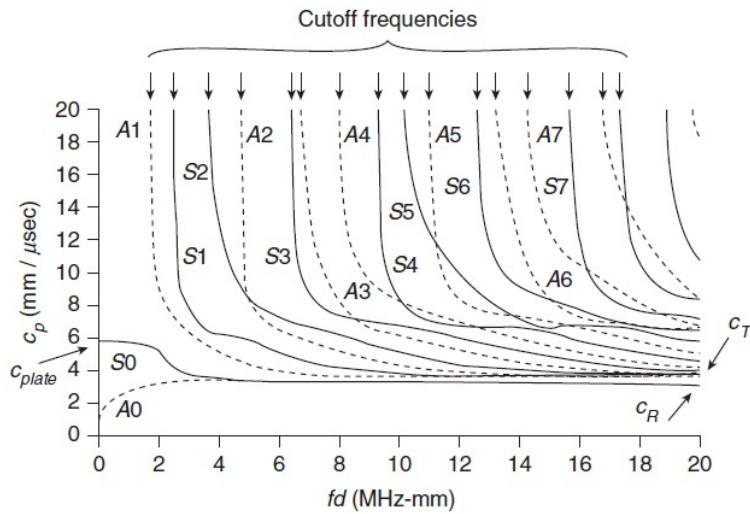
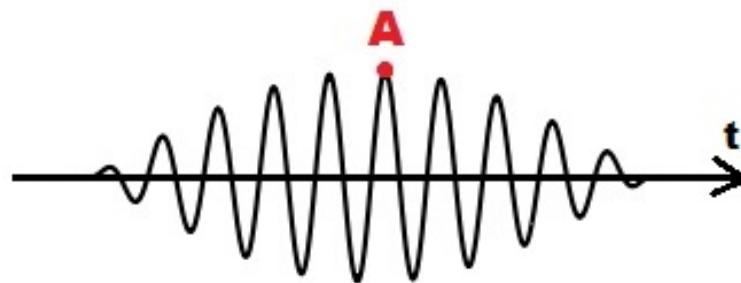
2.3.1. Prędkość fazowa i grupowa

Prędkość fazowa określa jak szybko przemieszcza się punkt fali o stałej fazie. Dla fali sinusoidalnej $u(x,t) = \sin(kx - \omega t)$ można wyznaczyć zależność na prędkość fazową w następujący sposób:

$$kx - \omega t = \text{const} \Rightarrow \frac{d(kx - \omega t)}{dt} = 0 \Rightarrow k \frac{dx}{dt} - \omega = 0 \Rightarrow \frac{dx}{dt} = \frac{\omega}{k} \quad (2.14)$$

Jak widać dla skończonej wartości ω i k dążącego do zera, prędkość fazowa rośnie do nieskończoności. Dla przypadku pierwszych modów, gdzie zarówno k jak i ω zmierzają do zera, sytuacja wygląda różnie. Na rysunku 2.9 znajduje się przykładowy wykres zależności prędkości fazowej od iloczynu częstotliwości i grubości materiału dla cienkościennej rury. W przypadku fal prowadzonych dla elementów cienkościennych, często wykresy charakterystyki z zaznaczeniem wpływu grubości ściany. Na wykresie widać, że wszystkie mody poza pierwszymi dwoma dążą prawostronnie do nieskończoności. Pozostałe dwa zachowują się odmiennie - jeden wykres dąży do 0 (postać antysymetryczna), a drugi do wartości skończonej (postać symetryczna).

Jeśli fala składa się z więcej niż jednej składowej sinusoidalnej, które mają różne prędkości, należy określić jeszcze prędkość grupową. Prędkość każdej z pojedynczych składowych, będzie jej prędkością fazową. Prędkość grupowa dotyczy sygnału, złożonego ze wszystkich składowych. Można ją prezentować jako prędkość jakiegoś punktu na wykresie sygnału, np. maximum, jak na rysunku 2.10.

Rys. 2.9. Krzywe $V_p(\omega)$ [1]

Rys. 2.10. Przykładowy sygnał. Prędkość punktu A jest prędkością grupową

Weźmy sygnał złożony z kilku sinusoid o zbliżonej częstotliwości. Fazę każdego sygnału można zapisać jako $\omega_i t - k_i x + \phi_i$, gdzie i oznacza numer składowej. Założymy, że fazy tych sygnałów są równe w punkcie $x=0, t=0$. Wynika stąd, że ϕ_i są niezależne od ω . Punkt $x=0, t=0$ będzie maksimum, ponieważ wszystkie składowe zsumują się w tym punkcie. Aby znaleźć kolejny punkt z takim maximum, należy określić gdzie fazy po raz kolejny będą sobie równe. Punkt, w którym to zjawisko zajdzie musi być niezależny od ω , w otoczeniu pewnego ω . Symbolicznie można to zapisać jako:

$$\frac{d(\omega t - kx - \phi)}{d\omega} = 0 \Rightarrow t - \frac{dk}{d\omega}x = 0 \Rightarrow \frac{x}{t} = \frac{d\omega}{dk} \quad (2.15)$$

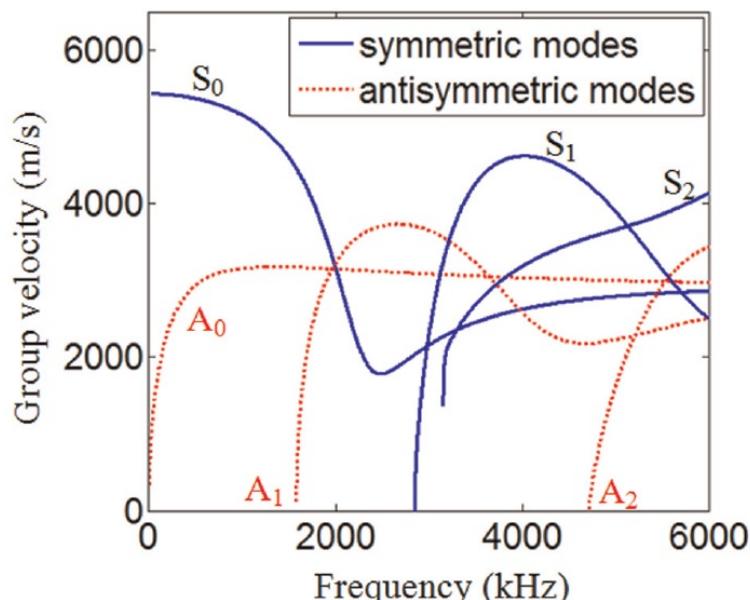
Maximum sygnału będzie więc w każdym punkcie x, t spełniającym powyższą zależność. Prędkość grupowa fali jest więc równa:

$$V_g = \frac{d\omega}{dk} \quad (2.16)$$

Z samego faktu istnienia takiej zależności nie wynika, że w sygnale znajdzie się jakiś charakterystyczny punkt, propagujący z taką prędkością. Wynika z niej natomiast,

że jeśli takowy punkt będzie, to będzie się poruszał z prędkością grupową. Należy jeszcze wspomnieć o wyznaczaniu prędkości grupowej sygnału o składowch, o różnych wartościach ω . W takim przypadku należy prędkość grupową wyznaczyć dla wartości ω sygnału dominującego. Dominujący sygnał można znaleźć przy pomocy przekształcenia Fouriera.

Na rysunku 2.11 znajdują się przykładowe krzywe, przedstawiające prędkość grupową dla fal Lamba, dla cienkościennej płyty aluminiowej.



Rys. 2.11. Wykres prędkości grupowej $V_g(\omega)$ [7]

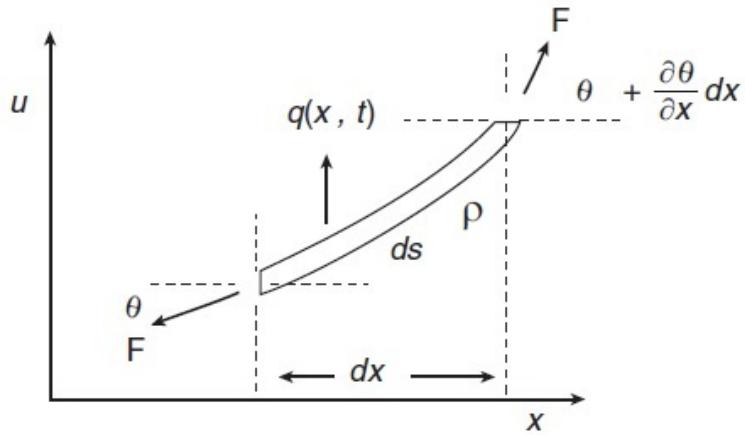
2.3.2. Analityczne wyznaczanie krzywych dyspersji

Zależności dyspersyjne są trudne, a często niemożliwe do wyznaczenia w sposób analityczny. Dla prostych przypadków rozwiązań analitycznych są znane, a kilka z nich jest poniżej opisanych w celu lepszego wyjaśnienia zjawiska dyspersji. Pierwszym przypadkiem będzie model nieskończenie długiej, napiętej sprężyny, przedstawionej na rysunku 2.12.

Symbole z rysunku oznaczają:

- ds – długość sprężyny
- θ – kąt przyłożenia siły zewnętrznej
- F – siły wewnętrzne
- q – siła oddziaływania sprężyny na jednostkę długości ugięcia
- ρ – masa sprężyny na jednostkę długości.

Korzystając z drugiej zasady dynamiki Newtona, zapisujemy równanie ruchu układu:



Rys. 2.12. Nieskończanie krótki odcinek napiętej sprężyny [1]

$$-F \sin \theta + F \sin(\theta + \frac{\partial \theta}{\partial x} dx) + q ds = \rho ds \frac{\partial^2 u}{\partial t^2}. \quad (2.17)$$

Założymy dodatkowo, że:

$$ds \approx dx, \sin \theta = \theta \text{ i } \theta = \frac{\partial u}{\partial x}. \quad (2.18)$$

To pozwala uprościć równanie:

$$-F\theta + F(\theta + \frac{\partial \theta}{\partial x} dx) = \rho dx \frac{\partial^2 u}{\partial t^2} \quad (2.19)$$

$$F \frac{\partial^2 u}{\partial x^2} + q = \rho \frac{\partial^2 u}{\partial t^2}. \quad (2.20)$$

Jeśli założymy brak siły zewnętrznej, to otrzymujemy proste równanie falowe:

$$\frac{\partial^2 u}{\partial x^2} = \frac{1}{c_0^2} \frac{\partial^2 u}{\partial t^2}, \quad c_0 = \sqrt{\frac{F}{\rho}} \quad (2.21)$$

gdzie

c_0 – prędkość fali.

Przyjmijmy rozwiązańe w postaci:

$$u(x, t) = A e^{i(kx - \omega t)}. \quad (2.22)$$

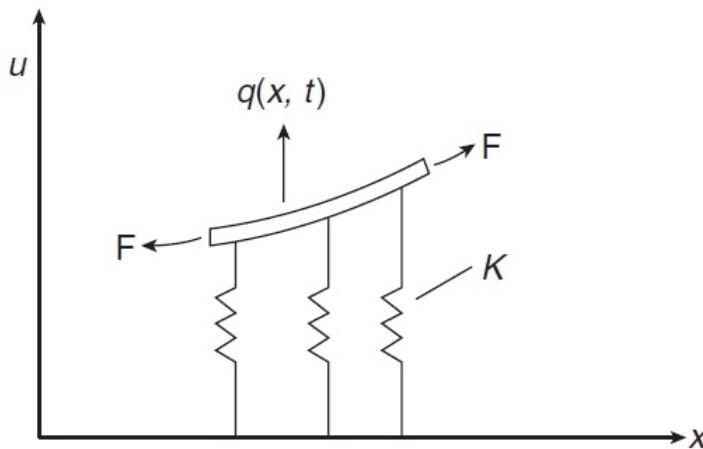
Jeśli wstawimy je do równania falowego to otrzymamy zależność dyspersyjną:

$$\omega^2 = c_0^2 k^2. \quad (2.23)$$

Jest to przypadek liniowej zależności $k(\omega)$, a więc dyspersja fali nie zachodzi. W takim przypadku prędkość fazowa, jest równa prędkości grupowej:

$$V_p = \frac{\omega}{k} = \frac{d\omega}{dk} = V_g \quad (2.24)$$

Prosta modyfikacja układu ze sprężyną jak na rysunku 2.13, powoduje skomplikowanie zależności dyspersyjnej.



Rys. 2.13. Nieskończoność krótki odcinek napiętej sprężyny na sprężystym podłożu [1]

Przyjmijmy siłę $q = -Ku(x, t)$, co prowadzi do równania równowagi:

$$\frac{\partial^2 u}{\partial x^2} - \frac{K}{F}u = \frac{1}{c_0^2} \frac{\partial^2 u}{\partial t^2}. \quad (2.25)$$

Ponownie założymy rozwiązanie w postaci $u(x, t) = Ae^{i(kx - \omega t)}$ i podstawmy je do równania równowagi. Prowadzi to do zależności:

$$\left(-k^2 - \frac{K}{F} + \frac{\omega^2}{c_0^2} \right) e^{i(kx - \omega t)} = 0. \quad (2.26)$$

Równanie to jest nazywane równaniem charaktersytycznym (lub dyspersyjnym). Rozwiązaniem jest:

$$-k^2 - \frac{K}{F} + \frac{\omega^2}{c_0^2} = 0. \quad (2.27)$$

Zależność $k(\omega)$ przyjmuje postać:

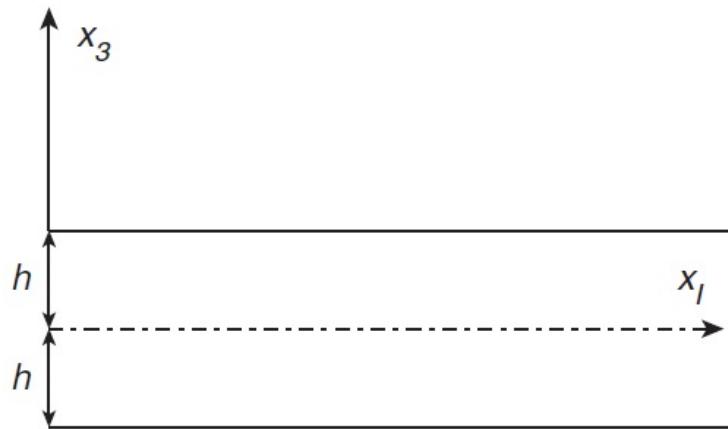
$$k^2 = \frac{\omega^2}{c_0^2} - \frac{K}{F}. \quad (2.28)$$

Podstawiając $k = \frac{\omega}{c_p}$, otrzymujemy :

$$c_p = \sqrt{c_0^2 \left(\frac{1}{1 - \frac{c_0^2 K}{\omega^2 F}} \right)}. \quad (2.29)$$

Jak widać w takim przypadku prędkość fazowa jest zależna od częstotliwości, a więc będzie zachodzić dyspersja fali.

Kolejnym problemem wartym wspomnienia, jest problem propagacji fal Lamba w cienkiej, nieskończonej płycie. Płyta wraz z przyjętym układem współrzędnych znajduje się na rysunku 2.14.



Rys. 2.14. Nieskończona płyta [1]

Jest kilka metod rozwiązywania takiego zagadnienia, ale nie są opisane one w tej pracy (metoda potencjałów, metoda fal cząstkowych). Jeśli przyjmiemy zerowe naprężenia na płaszczyznach zewnętrznych jako warunki początkowe, to związki dyspersyjne przyjmują zależności:

$$\frac{\tan(qh)}{\tan(ph)} = -\frac{4k^2 pq}{(q^2 - k^2)^2} \quad (2.30)$$

dla postaci symetrycznych oraz

$$\frac{\tan(qh)}{\tan(ph)} = -\frac{(q^2 - k^2)^2}{4k^2 pq} \quad (2.31)$$

dla postaci antysymetrycznych,

gdzie

$$p^2 = \frac{\omega^2}{c_L^2} - k^2, \quad a \quad q^2 = \frac{\omega^2}{c_T^2} - k^2 \quad (2.32)$$

c_T – prędkość fali poprzecznej

c_L – prędkość fali podłużnej

Równania 2.30, 2.31 nazywają się równaniami Rayleigha-Lamba i są nieroziwiązywalne w sposób analityczny. Jest to przykład bardziej skomplikowanego modelu, gdzie możliwe jest wyznaczenie związków k i ω , ale do wykreślenia krzywych dyspersji konieczne jest zastosowanie metod numerycznych.

2.3.3. Numeryczne wyznaczanie krzywych dyspersji

Pierwszym przypadkiem numerycznego wyznaczania krzywych dyspersji, jest rozwiązanie równań Rayleigha-Lamba przedstawionych w poprzedniej sekcji. Zakładamy, że interesują nas tylko rozwiązania rzeczywiste, które pojawią się jeśli k przyjmie wartości rzeczywiste bądź urojone (w równaniach występuje tylko czynnik k^2).

Poniżej znajdują się równania Rayleigha-Lamba w postaci ułatwiającej proponowany sposób rozwiązania:

$$\frac{\tan(qh)}{q} + \frac{4k^2 p \tan(ph)}{q^2 - k^2} = 0 \quad (2.33)$$

$$q \tan(qh) + \frac{(q^2 - k^2)^2 \tan(ph)}{2k^2 p} = 0. \quad (2.34)$$

Rozwiązać równanie możemy według przedstawionego algorytmu.

1. Wybierz iloczyn $(\omega h)_0$.
2. Wybierz początkową estymatę prędkości fazowe $(c_p)_0$ (a pośrednio też k).
3. Oblicz znak lewych stron równań R-L.
4. Wybierz kolejną wartość prędkości fazowej $(c_p)_1 > (c_p)_0$ i policz znaki lewych stron jeszcze raz.
5. Powtarzaj kroki 3 i 4 aż znaki się zmienią. Oznaczać to będzie, że pierwiastek równania znajduje się pomiędzy ostatnio wybranymi prędkościami. Założymy, że jest to pomiędzy $(c_p)_n$ i $(c_p)_{n+1}$.
6. Wykorzystaj jakiś algorytm iteracyjny do znalezienia pierwiastka c_p z odpowiednią dokładnością.
7. Po znalezieniu pierwiastka, kontynuuj wyszukiwanie kolejnych pierwiastków dla założonego $(\omega h)_0$, powtarzając kroki 2-6.
8. Wybierz kolejną wartość ωh i wykonaj dla niej kroki 2-7.

W ten sposób łatwo wyznaczyć krzywe dyspersji, ale należy zaznaczyć, że równania dyspersyjne zostały obliczone analitycznie dla konkretnego przypadku struktury, jaką jest płyta. Metody numeryczne ograniczają się w tym przypadku do rozwiązania wyznaczonych równań. Możliwość ogólniejszego podejścia do badań numerycznych, dla szerokiego wachlarza struktur daje metoda elementów skończonych.

Metoda ta daje możliwości dokładnego przewidzenia odpowiedzi dynamicznych, dla złożonych układów mechanicznych. Jeśli założymy przypadek małych odkształceń, to problem można zapisać w postaci równania różniczkowego:

$$M\ddot{u} + Ku = f^{ext} \quad (2.35)$$

gdzie

- M – macierz mas układu
- K – macierz sztywności układu
- u – przemieszczenia punktów układu
- f^{ext} – wektor sił zewnętrznych

Sposobem wyznaczenia krzywych dyspersji z tak określonego układu, może być rozwiązanie równania tj. znalezienie pola przemieszczeń dla wszystkich węzłów w dyskretnych chwilach czasowych, a następnie obliczenie wielowymiarowego przekształcenia Fouriera otrzymanego sygnału.

Równanie można rozwiązać na przykład z wykorzystaniem centralnej formuły różnicowej dla \ddot{u} :

$$\ddot{x} = \frac{u^{t+1} - 2u^t + u^{t-1}}{\Delta t^2}. \quad (2.36)$$

Po podstawieniu otrzymujemy zależność:

$$\frac{1}{\Delta t^2} M(u^{t-1} - 2u^t + u^{t-1}) + Ku^t = f^t \quad (2.37)$$

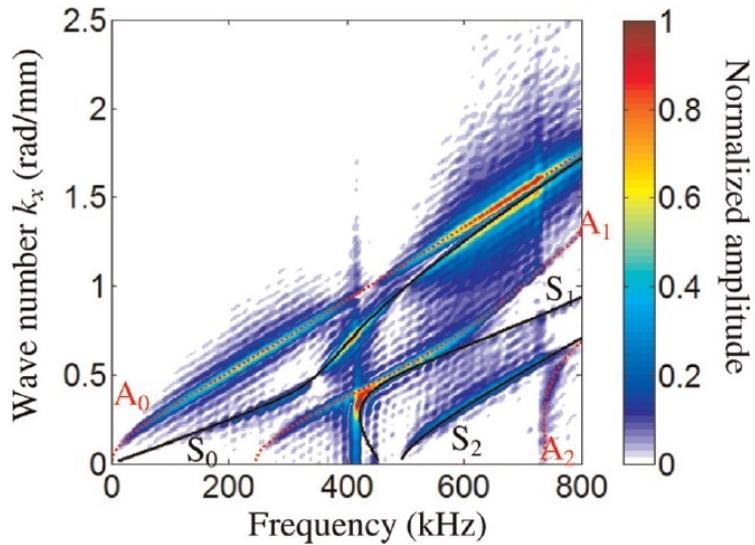
$$u^{t+1} = \Delta t^2 M^{-1} (f^t - Ku^t) + 2u^t - u^{t-1} \quad (2.38)$$

gdzie t oznacza obecną chwilę czasową, a $t + 1$ chwilę kolejną. Operacja odwracania macierzy mas jest kosztowna obliczeniowo. Można przyśpieszyć obliczenia przez zastosowanie skupionej macierzy mas (wyrazy niezerowe tylko na głównej diagonali).

Jeśli interesuje nas propagacja tylko w jednym kierunku, to dla obliczonego sygnału $u[m, n]$ (m - liczba kroków czasowych Δt , n - liczba kroków geometrycznych Δx w kierunku x) przekształcenie Fouriera można obliczyć według wzoru:

$$U[p, q] = \sum_{k=0}^{m-1} \sum_{l=0}^{n-1} u[k, l] e^{-i2\pi(-p\frac{k}{m} + q\frac{l}{n})} \quad (2.39)$$

Mając dyskretną postać przekształcenia Fouriera można wykreślić krzywe dyspersji, wyznaczając dla każdej próbki wartość częstotliwości i liczby falowej. Można to zrobić wiedząc, że częstotliwość zawiera się w przedziale $[0, \frac{1}{2\Delta t} 2\pi]$, a liczba falowa $[0, \frac{1}{2\Delta x} 2\pi]$. Przykładowe krzywe przedstawione są na rysunku 2.15.



Rys. 2.15. Krzywe dyspersji wyznaczone przy pomocy dwuwymiarowego DFT [7]

Innym sposobem wyznaczania krzywych dyspersji z pomocą modelu MES, jest symulacja układu dla wymuszeń harmonicznych o ustalonych częstotliwościach, w których chcemy wyznaczyć krzywe. Jako przykład rozpatrzmy jednorodny pręt. Dla przypadku pręta wymuszanego na jednej płaszczyźnie przekroju, dane należy zebrać przed odbiciem fali od swobodnego końca pręta, tak aby sygnał propagował tylko w jednym kierunku. W liniach równoległych do osi, obliczamy pola prędkości przemieszczenia punktów. Może to być też pole przemieszczeń lub naprężeń, ale pole prędkości jest stosunkowo łatwe do wyznaczenia. Znając już prędkości dla wszystkich punktów w jednej linii, w chwili t , możemy wyznaczyć widmo otrzymanego sygnału. Następnie z widma sygnału odczytujemy liczby falowe, dla których powstały widoczne piki. Pik dla największej liczby falowej dotyczy pierwszego modu, a każdy kolejny, z coraz mniejszymi liczbami falowymi, modów kolejnych.

Pole prędkości można wyznaczyć na podstawie różnicowych formuł centralnych 2.40 oraz 2.41.

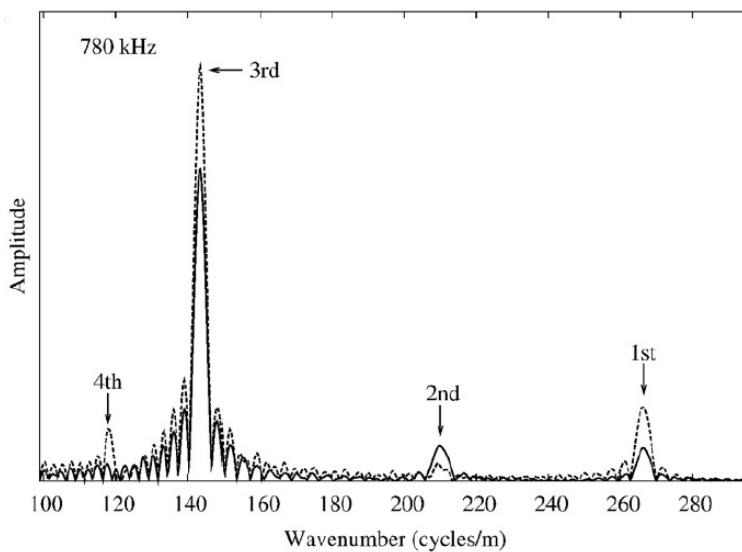
$$\ddot{u}^{t+1} = \ddot{u}^n + \frac{\Delta t}{2}(\ddot{u}^n + \ddot{u}^{n+1}) \quad (2.40)$$

$$u^{n+1} = u^n + \Delta t(\dot{u}^n + \frac{\Delta t}{2}\ddot{u}^n) \quad (2.41)$$

Założymy, że znamy pełne rozwiązanie w chwili t . Przemieszczenie w chwili $t+1$ możemy więc obliczyć z zależności 2.41. Następnie podstawiamy wynik do równania 2.35 i obliczamy przyspieszenie w chwili $t+1$. Ostatecznie korzystając z formuły 2.40 możemy wyznaczyć prędkość w chwili $t+1$.

Przykładowe widmo, z którego możemy odczytać liczby falowe czterech modów znajduje się na rysunku 2.16. Jak widać nie wszystkie liczby falowe da się wykryć przy

pomocy danych z jednej linii, więc niezbędne jest prowadzenie obliczeń dla kilku różnych linii.



Rys. 2.16. Widmo sygnału prędkości na liniach równoległych do osi pręta [8]

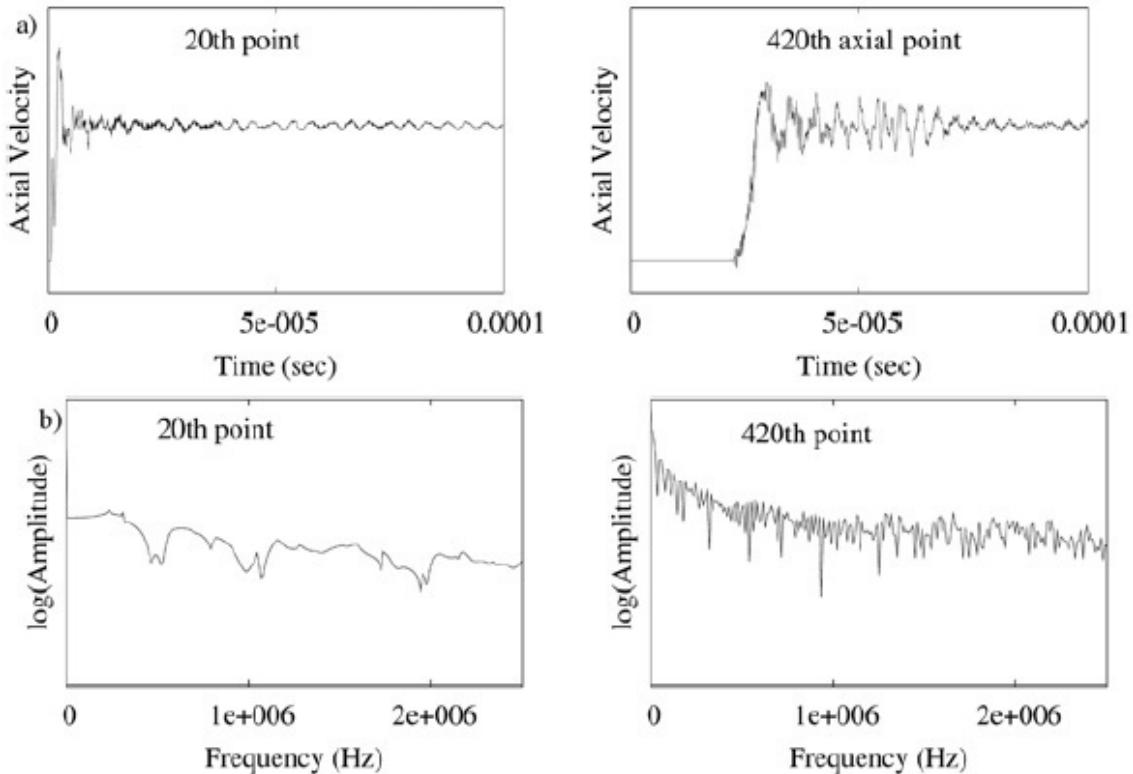
Wadą tego sposobu jest konieczność prowadzenia obliczeń dla każdej częstotliwości, dla jakiej chcemy mieć punkty krzywych dyspersji. Jedne przebiegi dają nam tylko jeden punkt na każdej krzywej. Fakt prowadzenia obliczeń tylko w liniach równoległych do osi pręta, ogranicza dodatkowo obszar propagujących fal do fal podłużnych.

Sposobem zmniejszenia liczby symulacji jakie trzeba przeprowadzić jest symulacja z wymuszeniem szerokopasmowym. Pozwala to na uzyskanie informacji o krzywych dyspersji, dla częstotliwości zawierających się w wymuszającym sygnale. Pojawia się jednak kilka dodatkowych kroków jakie trzeba wykonać, aby uzyskać liczby falowe modów dla poszczególnych częstotliwości.

Aby uzyskać dane do wykreszenia krzywych dyspersji, należy obliczyć przestrzenny rozkład prędkości na liniach równoległych do osi dla wszystkich częstotliwości. W tym celu trzeba wykonać następujące czynności.

1. Wyznaczyć odpowiedzi czasowe dla każdego punktu na interesującej nas lini równoległej do osi pręta.
2. Obliczyć transformaty Fouriera tych sygnałów
3. Z widma sygnału w każdym punkcie pobrać informacje o amplitudzie dla kolejnych częstotliwości. Na tej podstawie odtworzyć amplitudowy przebieg sygnału na całej linii dla każdej interesującej nas częstotliwości.
4. Dla obliczonych w pkt. 3 przebiegów, obliczyć transformaty Fouriera i z wykresu odczytać wartości liczby falowej, dla wybranych częstotliwości.

Rysunki 2.17 oraz 2.18 przedstawiają kolejne kroki postępowania, dla odpowiedzi na wymuszenie krokiem jednostkowym preta zbudowanego z 1601 węzłów na długości jednej lini.

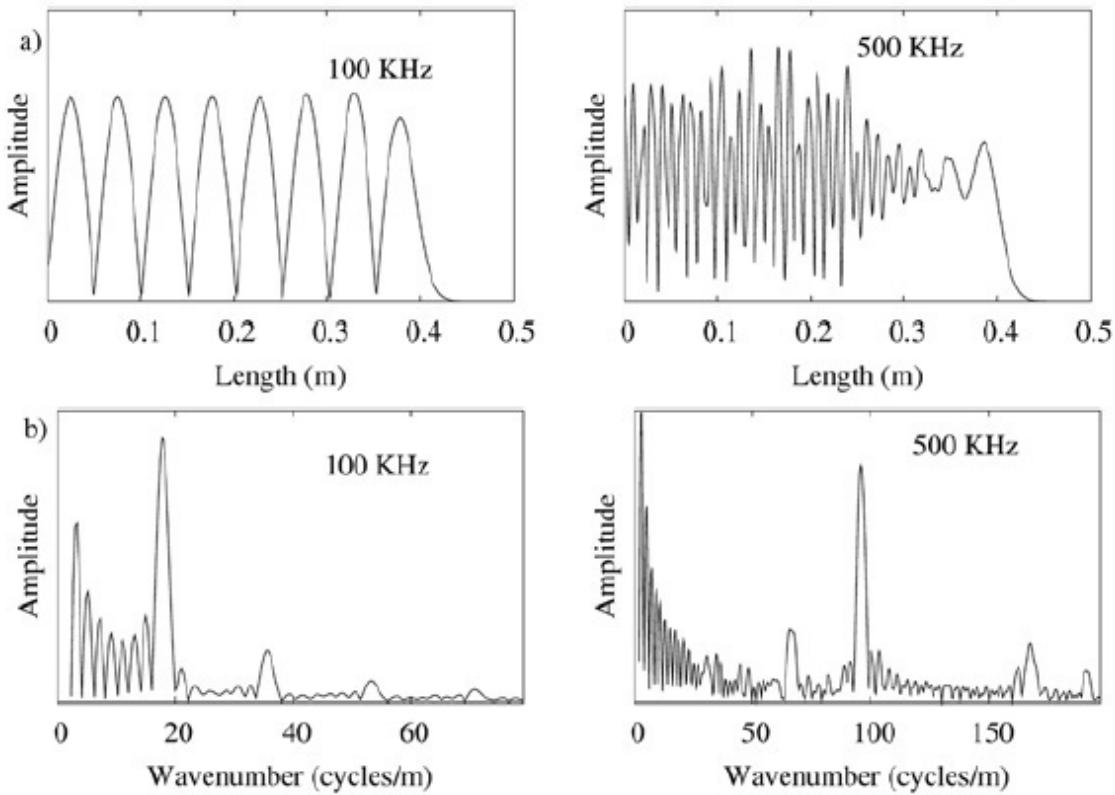


Rys. 2.17. a) Odpowiedzi na szerokopasmowe wymuszenie w wybranych punktach jednej lini b) Przekształcenia Fouriera odpowiedzi czasowych [8]

Ostatnia przedstawiona tutaj metoda skupia się, na rozwiązyaniu zagadnienia własnego modelu wyznaczonego z pomocą MES. Tym razem zakładając będziemy wybraną liczbę falową i dla niej obliczać częstotliwości poszczególnych modów. Po raz kolejny za przykład weźmy preć. Pręt dzielimy na komórki, które będą się składać z kolejnych płaszczyzn, na których znajdują się węzły. Znając odległość płaszczyzn i wybierając liczbę falową, możemy określić przesunięcie fazowe fali pomiędzy płaszczyznami. Sytuacja przedstawiona jest na rysunku 2.19.

W liniowych systemach do opisu popagacji fali wystarcza jedna komórka i siły wewnętrzne pochodzące od komórek sąsiednich. W przypadku braku sił zewnętrznych i przy założeniu skupionej macierzy mas, równanie układu można zapisać w postaci 2.42. Rysunek 2.20 przedstawia sposób wyznaczania macierzy sztywności dla komórki środkowej, z uwzględnieniem zależności z komórkami sąsiednimi.

$$M_0 \ddot{x}_0 + \sum_{p=-1}^1 K_p x_p = 0 \quad (2.42)$$



Rys. 2.18. a) Odtworzony sygnał amplitudowy na długości pręta w jednej linii b) Przekształcenie Fouriera odtworzonych sygnałów [8]

Uwzględniając wszystkie informacje równanie układu można zapisać jako 2.43. Podstawiając dodatkowo zależności $\ddot{x} = x\omega^2$, $M_{sys} = M_0$ oraz $K_{sys} = K_{-1}e^{-ikx_k} + K_0 + K_1e^{ikx_k}$ otrzymujemy równanie 2.44.

$$M_0 \ddot{x}_0 + K_{-1}x_0 e^{-ikx_k} + K_0 x_0 + K_1 x_0 e^{ikx_k} = 0 \quad (2.43)$$

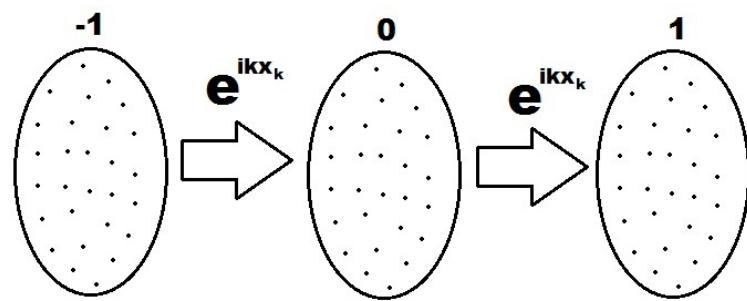
$$(M_{sys}\omega^2 + K_{sys})x_0 = 0 \quad (2.44)$$

Ostatecznie rozwiązując zagadnienie własne dla pary macierzy M_{sys} i K_{sys} otrzymujemy kwadraty częstości własnych układu dla założonego k . Każda z częstości należy do jednego modu krzywych dyspersji.

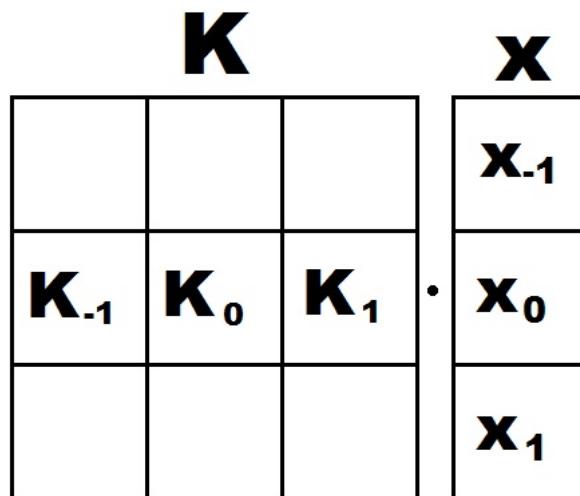
Wadami tej metody są konieczność prowadzenia obliczeń dla każdej liczby falowej, którą chcemy uwzględnić w wynikach, oraz brak dokładnej informacji, która częstość własna należy do której postaci fali.

Dużą zaletą z kolei jest fakt, że wyznaczamy jedną metodą krzywe dyspersji dla fal podłużnych i poprzecznych.

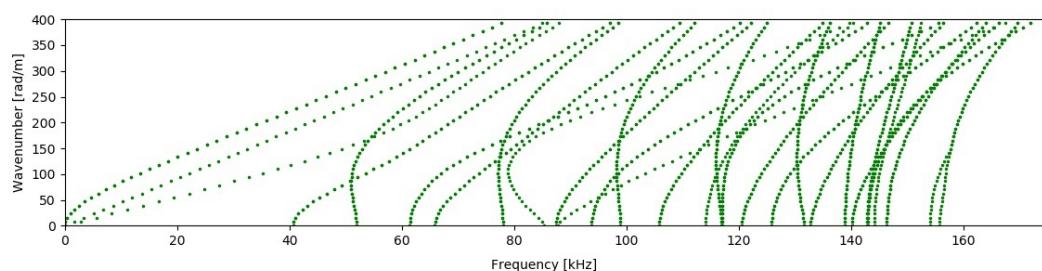
Przykładowe krzywe dyspersji uzyskane w ten sposób przedstawia rysunek 2.21.



Rys. 2.19. Kolejne komórki pręta



Rys. 2.20. Wyznaczanie macierzy sztywności środkowej komórki z uwzględnieniem zależności z komórkami sąsiednimi



Rys. 2.21. Krzywe uzyskane przez wielokrotne rozwiązywanie zagadnienia własnego pręta

2.3.4. Eksperimentalne wyznaczanie krzywych dyspersji

Eksperimentalne badania fal prowadzonych przeprowadza się, wzbudzając odpowiednio obiekt, a następnie zbiera się dane o powstałej fali w innych punktach konstrukcji lub w tym samym punkcie po odbiciu fali. Zebrane dane analizuje się za pomocą algorytmów, w których duże znaczenie ma przekształcenie Fouriera.

Jako wzbudniki fal prowadzonych często stosowane są przetworniki piezoelektryczne (na przykład z ceramik PZT). Proste zjawisko piezoelektryczne z jakiego korzystają takie przetworniki, pozwala przekształcić energię elektryczną na energię mechaniczną. Efekt odwrotny pozwala zbierać informację o energii odkształcenie w materiale i sygnał mechaniczny przekształcić w elektryczny.

Materiały piezoelektryczne znalazły szerokie zastosowanie ze względu na dobre parametry mechaniczne, odporność na wiele substancji chemicznych oraz bardzo duże możliwości kształtowania przetworników. Wzbudniki i sensory piezoelektryczne często występują jako cienkie fragmenty materiału piezoelektrycznego, które można wykonać w dowolnym kształcie. W przypadku kiedy chcemy uzyskać większą amplitudę wzbudzenia, stosuje się stopy stworzone z wielu warstw piezoelektryka.

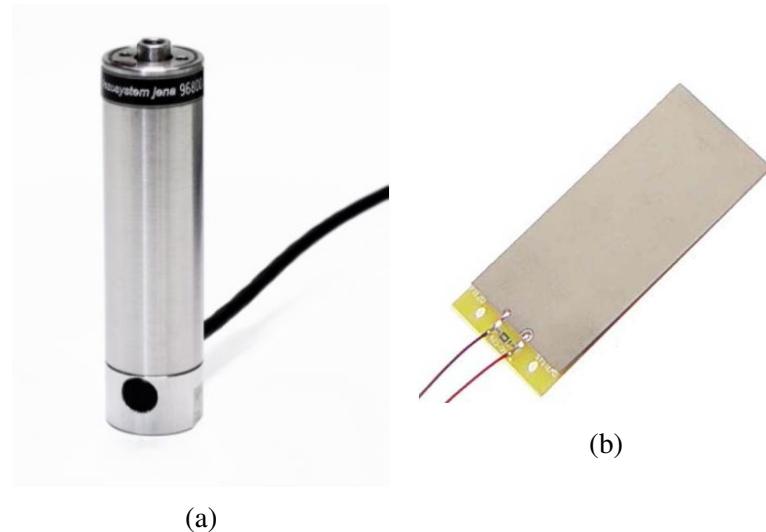
Obecna ilość dostępnych na rynku rozwiązań pozwala na dobranie odpowiedniego przetwornika do niemal każdego projektu bazującego na niskich częstotliwościach. Podczas analizy rozwiązań komercyjnych okazało się, że duża część takich przetworników posiada ograniczenia działania do kilku kHz. Jest to najczęściej zbyt niska częstotliwość, nawet w przypadku badania wyłącznie modów podstawowych. Dostępne są też rozwiązania dla częstotliwości do nawet kilkuset kHz, co w przypadku fal prowadzonych może stanowić wystarczającą wartość.

Przykłady przetworników piezoelektrycznych w formie płytka oraz stopy znajdują się na rysunku 2.22.

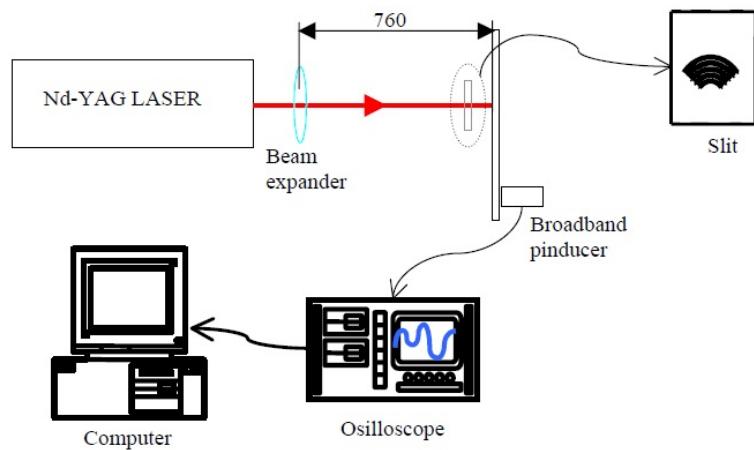
Innym sposobem badań jest wykorzystanie techniki laserowej. Za pomocą wiązki lasera możliwe jest wzbudzenie fali sprężystych w materiale. Na szczególną uwagę zasługuje możliwość wzbudzenia fali krótki impulsem, którego widmo będzie miało bardzo duży zakres częstotliwości. Sygnał taki możemy porównać do delty Diraca, dla której widmo ma stałą amplitudę.

Za pomocą lasera możemy wykonywać także pomiary drgań materiału. Posłużyć może do tego na przykład interferometr laserowy, wykorzystujący efekt Dopplera. Detekcja jest jednak ograniczona do fal powierzchniowych. Atutem takiego pomiaru jest możliwość wykonania go dla wielu punktów na badanym obiekcie. Dzięki temu możemy uzyskać przestrzenny obraz propagacji fali.

Przykładowe stanowisko do badania płyt za pomocą wymuszenia laserem przedstawia rysunek 2.23.



Rys. 2.22. Przetworniki piezoelektryczne: (a) w formie sotsu, (b) w formie płytki



Rys. 2.23. Przykładowy układ stanowiska do badania fal Lamba w cienkiej płycie [7]

3. Wykorzystywane zagadnienia metody elementów skończonych

BARTŁOMIEJ PIWOWARCZYK

Metoda elementów skończonych jest zaawansowanym sposobem rozwiązywania równań różniczkowych cząstkowych. Jest ona szczególnym przypadkiem metody Galerkina. Nazwa pochodzi od nazwiska Borisa Grigoriewicza Galerkina, który jako pierwszy przeprowadził usystematyzowane studia na temat tego typu metod. Za pomocą tej metody można rozwiązać skomplikowane zadania dziedzin takich jak mechanika ciała stałego, mechanika płynów czy teoria pola elektromagnetycznego. Podstawy tej metody opisane są w [16] oraz [17]. Informacje na temat pochodzenia i estymacji błędów zaczerpnięto z [18] oraz [19].

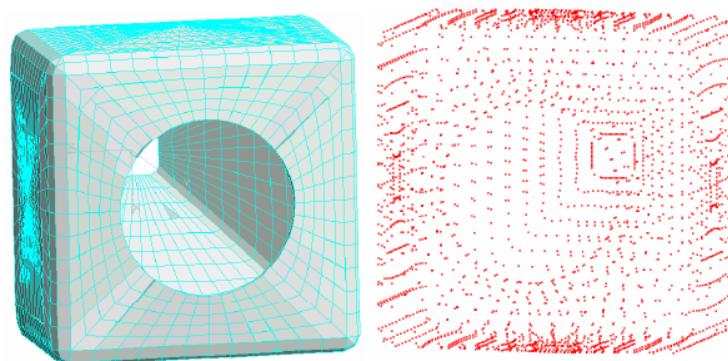
3.1. Rozwiązywanie zadań z pomocą MES

W wytrzymałości materiałów poszukujemy zwykle sił działających w podporach konstrukcji i naprężeń wewnętrz jej konkretnych elementów. Innym podejściem jest znalezienie pola przemieszczeń dla każdego punktu konstrukcji. W jednym i drugim przypadku znalezienie niewiadomych jest osiągalne wyłącznie dla mało skomplikowanych konstrukcji.

Metoda elementów skończonych bierze swoją nazwę od wydzielanych fragmentów konstrukcji, które są właśnie elementami skończonymi. Elementy tworzymy na siatce węzłów, które są punktami wewnętrz lub na brzegach konstrukcji. Węzły są także punktami wspólnymi sąsiadujących elementów. Rozwiązywanie opiera się na znalezieniu przemieszczenia węzłów i na tej podstawie obliczenia rozwiązania we wszystkich punktach wewnętrz elementów. Kolejne etapy rozwiązywania zadania za pomocą MES przedstawione są poniżej. Rysunek 3.1 przedstawia przykład obiektu, który został zdyskretyzowany za pomocą elementów skończonych.

1. Generacja siatki węzłów.
2. Budowa elementów skończonych na stworzonej siatce.
3. Wyznaczanie macierzy mas i sztywności dla każdego elementu (macierze lokalne).

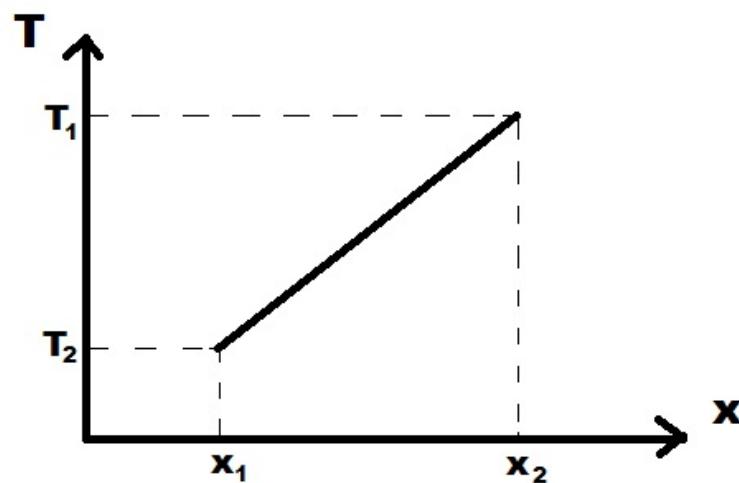
4. Agregacja macierzy mas i sztywności dla całej konstrukcji (macierze globalne).
5. Wprowadzenie wektora obciążień.
6. Wprowadzenie warunków brzegowych. Mogą to być siły czynne (modyfikacja wektora obciążień), naprężenia początkowe czy przemieszczenia wskazanych węzłów.
7. Rozwiążanie wyznaczonego liniowego równania różniczkowego $M\ddot{x} + Kx = F$ - znalezienie przemieszczeń węzłów.
8. Wyznaczanie odkształceń i naprężen.
9. Wyznaczenie reakcji w podporach (węzłach, dla których założono przemieszczenie)



Rys. 3.1. Przykład dyskretyzacji obiektu za pomocą elementów skończonych [16]

3.2. Funkcje kształtu

Weźmy metalowy pręt o długości L . W punkcie x_1 na pręcie przykładamy temperaturę T_1 , a w punkcie x_2 temperaturę T_2 . Przewidujemy, że po nieskończonym długim czasie rozkład temperatury w pręcie pomiędzy tymi punktami będzie liniowy (Rys. 3.2).



Rys. 3.2. Temperatur pomiedzy punktami pręta

Temperaturę pomiędzy tymi punktami możemy przedstawić funkcją 3.1.

$$T(x) = a_0 + a_1 x, \quad \text{dla } x \in [x_1, x_2] \quad (3.1)$$

Można wyznaczyć współczynniki funkcji liniowej mając dwa jej punkty.

$$a_1 = \frac{T_2 - T_1}{x_2 - x_1} \quad (3.2)$$

$$a_0 = T_2 - a_1 x_2 \quad (3.3)$$

Podstawiając współczynniki do funkcji, możemy wyznaczyć funkcje kształtu.

$$T(x) = T_1 - \frac{T_2 - T_1}{L} x_1 + \frac{T_2 - T_1}{L} x = \frac{x_2 - x}{L} T_1 + \frac{x - x_1}{L} T_2 = N_1(x) T_1 + N_2(x) T_2, \quad (3.4)$$

gdzie

N_1, N_2 – funkcje kształtu.

Funkcje kształtu określają w jakim stopniu wartość temperatury z jednego węzła wpływa na wartość temperatury w dowolnym punkcie wewnętrz elementu. Dla większej liczby węzłów w jednowymiarowym elemencie funkcje kształtu są wielomianami wyższych rzędów. Poniżej znajduje się sposób wyznaczania funkcji kształtu na przykładzie 3-węzłowego elementu jednowymiarowego.

$$T(x) = a_0 + a_1 x + a_2 x^2 = \begin{bmatrix} 1 & x & x^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \mathbf{p} \mathbf{a}^e = \mathbf{N} \mathbf{T}^e \quad (3.5)$$

gdzie

\mathbf{p} – wektor zmiennych kolejnych jednomianów $T(x)$

\mathbf{a}^e – wektor współczynników kolejnych jednomianów $T(x)$

\mathbf{N} – wektor funkcji kształtu

\mathbf{T}^e – wektor wartości w węzłach.

Z równania 3.5 wynika poniższa zależność.

$$\mathbf{T}^e = \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \mathbf{M}^e \mathbf{a}^e \quad (3.6)$$

Wyznaczając z tego równania \mathbf{a}^e i podstawiając do równania 3.5, otrzymamy wzór na funkcje kształtu.

$$\mathbf{p} \mathbf{a}^e = \mathbf{p}(\mathbf{M}^e)^{-1} \mathbf{T}^e = \mathbf{N} \mathbf{T}^e \Rightarrow \mathbf{N} = \mathbf{p}(\mathbf{M}^e)^{-1} \quad (3.7)$$

Funkcje kształtu obliczone w taki sposób mają postać jak poniżej.

$$\begin{aligned} N_1 &= \frac{1}{\det \mathbf{M}^e} (x_2 x_3^2 - x_2^2 x_3 + (x_2^2 - x_3^2)x + (-x_2 + x_3)x^2) \\ N_2 &= \frac{1}{\det \mathbf{M}^e} (x_1^2 x_3 - x_1 x_3^2 + (x_3^2 - x_1^2)x + (x_1 - x_3)x^2) \\ N_3 &= \frac{1}{\det \mathbf{M}^e} (x_1 x_2^2 - x_1^2 x_2 + (x_1^2 - x_2^2)x + (-x_1 + x_2)x^2) \end{aligned} \quad (3.8)$$

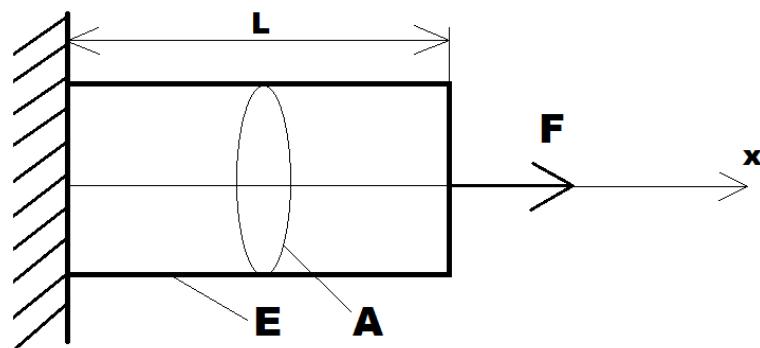
Prawidłowo wyznaczone funkcje kształtu mają dwie ważne własności. Pierwsza związana jest z węzłami i mówi że każda funkcja przyjmuje wartość 1 w jednym węzle, a we wszystkich pozostałych 0. Druga własność mówi, że suma funkcji kształtu wewnętrz elementu skończonego wynosi 1. Matematycznie zapisać te własności można następująco:

$$\begin{cases} N_n(x_m, y_m) = 1, & \text{dla } n = m \\ N_n(x_m, y_m) = 0, & \text{dla } n \neq m \end{cases} \quad (3.9)$$

$$\sum_1^n N_n = 1. \quad (3.10)$$

3.3. Wyznaczanie macierzy sztywności i mas elementów

Weźmy model jednorodnego pręta, obciążonego jak na rysunku 5.21 o module Younga E , polu przekroju A i długości L .



Rys. 3.3. Model pręta

Energię potencjalną i kinetyczną układu można wyrazić poniższymi wzorami.

$$V = \frac{1}{2} \int_0^L \varepsilon A \sigma dx - F u_{x=L} = \frac{1}{2} \int_0^L EA \left(\frac{du}{dx} \right)^2 dx - F u_{x=L} \quad (3.11)$$

$$K = \frac{1}{2} \int_0^L \rho A \left(\frac{\partial u}{\partial t} \right)^2 dx \quad (3.12)$$

gdzie

V – energia potencjalna

K – energia kinetyczna

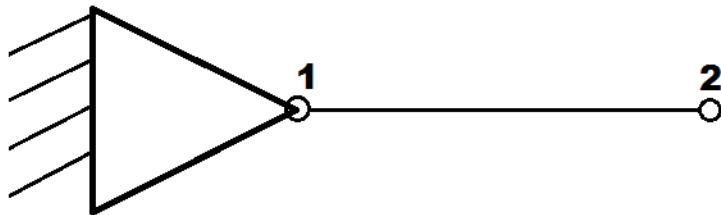
u – przemieszczenie

ε – odkształcenie

σ – naprężenie

F – siła zewnętrzna skupiona

Następnie stwórzmy uproszczony model, do którego wyznaczymy zależności MES.



Rys. 3.4. Uproszczony model pręta

Znając funkcje kształtu i wartości przemieszczeń węzłów można wyznaczyć przemieszczenie w każdym punkcie pręta, a następnie odpowiednie pochodne po przemieszczeniach i po czasie.

$$u = N_1 u_1 + N_2 u_2 = \begin{bmatrix} N_1 & N_2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \quad (3.13)$$

$$\frac{\partial u}{\partial x} = \begin{bmatrix} \frac{dN_1}{dx} & \frac{dN_2}{dx} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \mathbf{B} \mathbf{u}^e \quad (3.14)$$

$$\frac{\partial u}{\partial t} = \begin{bmatrix} N_1 & N_2 \end{bmatrix} \begin{bmatrix} \frac{du_1}{dt} \\ \frac{du_2}{dt} \end{bmatrix} = \mathbf{N} \dot{\mathbf{u}}^e \quad (3.15)$$

Mając wyznaczoną energię, możemy obliczyć lagranżjan i wstawić go do dynamicznych równań Lagrangea drugiego rodzaju. Dzięki tej operacji otrzymamy końcowe, dynamiczne równanie ruchu oraz postać macierzy mas i sztywności. Równania Lagrangea drugiego rodzaju mają postać:

$$\frac{d}{dt} \frac{\partial L}{\partial \dot{\mathbf{u}}^e} - \frac{\partial L}{\partial \mathbf{u}^e} = 0, \quad L = K - V \quad (3.16)$$

gdzie

L – lagranżjan.

Pierwszy człon wynosi

$$\frac{d}{dt} \frac{\partial L}{\partial \dot{\mathbf{u}}^e} = \int_0^L \rho A \mathbf{N}^T \mathbf{N} \ddot{\mathbf{u}}^e dx, \quad (3.17)$$

zaś drugi

$$\frac{\partial L}{\partial \mathbf{u}^e} = - \int_0^L E \mathbf{A} \mathbf{B}^T \mathbf{B} \mathbf{u}^e dx + F \mathbf{N} \mathbf{u}_{x=L}^e. \quad (3.18)$$

Równanie dynamiki układu zapisane jest poniżej.

$$\int_0^L \rho A \mathbf{N}^T \mathbf{N} dx \ddot{\mathbf{u}}^e + \int_0^L \mathbf{A} \mathbf{B}^T E \mathbf{B} dx u = F \mathbf{N} \mathbf{u}_{x=L}^e \quad (3.19)$$

Stąd postaci macierzy mas i sztywności są następujące:

$$\mathbf{M}^e = \int_0^L \rho A \mathbf{N}^T \mathbf{N} dx \quad (3.20)$$

$$\mathbf{K}^e = \int_0^L \mathbf{A} \mathbf{B}^T E \mathbf{B} dx. \quad (3.21)$$

W przypadku kiedy występuje więcej niż jedna stała materiałowa, zamiast E pojawia się macierz materiałowa D . W przypadku bardziej złożonych obiektów o większej liczbie elementów skończonych, całkowanie niezbędne do obliczenia macierzy staje się bardzo czasochłonne. Ponieważ funkcje kształtu są wielomianami, optymalnym rozwiążaniem jest zastosowanie kwadratur Gaussa. Kolejny problem to określenie granic całkowania. W przypadku przedstawionym powyżej nie widać specjalnej trudności, natomiast dla trójwymiarowych obiektów o nieregularnych kształtach wymaga to dodatkowej serii obliczeń.

W takich przypadkach można zastosować mapowanie na współrzędne naturalne. W takich współrzędnych element może mieć z góry ustalone współrzędne węzłów, co za tym idzie także granice całkowania są znane. Dla obiektu trójwymiarowego przyjmijmy współrzędne rzeczywiste x, y, z i współrzędne naturalne ξ, η, ζ . Mapowania z jednych

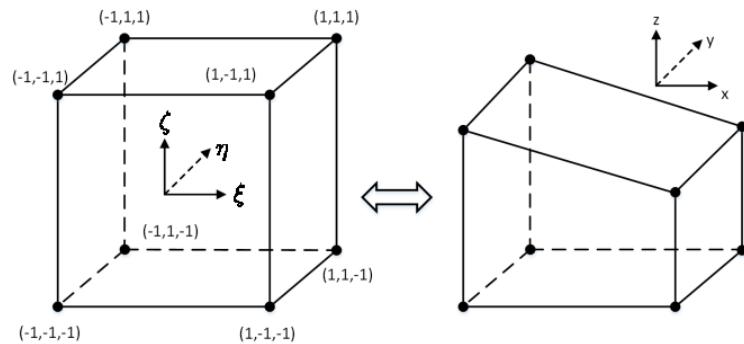
współrzędnych na drugie oblicza się według poniższego wzoru. Wizualizacja procedury przedstawiona jest na rysunku 3.5.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \sum_{i=1}^n \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} N_i(\xi, \eta, \zeta) \quad (3.22)$$

x_i, y_i, z_i – współrzędne rzeczywiste punktów

N_i – funkcje kształtu we współrzędnych naturalnych

n – liczba węzłów elementu.



Rys. 3.5. Przekształcenie izoparametryczne

Dla elementów w takich współrzędnych funkcje kształtu są znane i stabelaryzowane.

Dla elementu sześciennego wypisane są poniżej.

$$\begin{aligned} N_1 &= \frac{1}{8}(1 - \xi)(1 - \eta)(1 - \zeta) \\ N_2 &= \frac{1}{8}(1 + \xi)(1 - \eta)(1 - \zeta) \\ N_3 &= \frac{1}{8}(1 + \xi)(1 + \eta)(1 - \zeta) \\ N_4 &= \frac{1}{8}(1 - \xi)(1 + \eta)(1 - \zeta) \\ N_5 &= \frac{1}{8}(1 - \xi)(1 - \eta)(1 + \zeta) \\ N_6 &= \frac{1}{8}(1 + \xi)(1 - \eta)(1 + \zeta) \\ N_7 &= \frac{1}{8}(1 + \xi)(1 + \eta)(1 + \zeta) \\ N_8 &= \frac{1}{8}(1 - \xi)(1 + \eta)(1 + \zeta) \end{aligned} \quad (3.23)$$

Ostatnim elementem niezbędnym do prawidłowego całkowania we współrzędnych naturalnych jest wyznaczenie jakobianu przekształcenia. Macierz Jacobiego przedstawiona jest w równaniu 3.24.

$$\mathbf{J} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} & \frac{\partial z}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} & \frac{\partial z}{\partial \eta} \\ \frac{\partial x}{\partial \zeta} & \frac{\partial y}{\partial \zeta} & \frac{\partial z}{\partial \zeta} \end{bmatrix} \quad (3.24)$$

Ostatecznie wyznaczanie macierzy mas i sztywności dla elementu sześciennego zostanie zmodyfikowane jak poniżej:

$$\mathbf{K}^e = \int_V \mathbf{B}^T E \mathbf{B} dV \rightarrow \mathbf{K}^e = \int_{-1}^1 \int_{-1}^1 \int_{-1}^1 \mathbf{B}^T \mathbf{D} \mathbf{B} \det \mathbf{J} d\xi d\eta d\zeta. \quad (3.25)$$

$$\mathbf{M}^e = \int_V \rho \mathbf{N}^T \mathbf{N} dV \rightarrow \mathbf{M}^e = \int_{-1}^1 \int_{-1}^1 \int_{-1}^1 \rho \mathbf{N}^T \mathbf{N} \det \mathbf{J} d\xi d\eta d\zeta \quad (3.26)$$

Cała procedura pozwala na całkowanie we współrzędnych naturalnych, gdzie granice całkowania są stałe. Uwzględnienie jacobianu powoduje, że całka w nowych współrzędnych jest równa całce na współrzędnych rzeczywistych.

3.4. Agregacja globalnych macierzy mas i sztywności

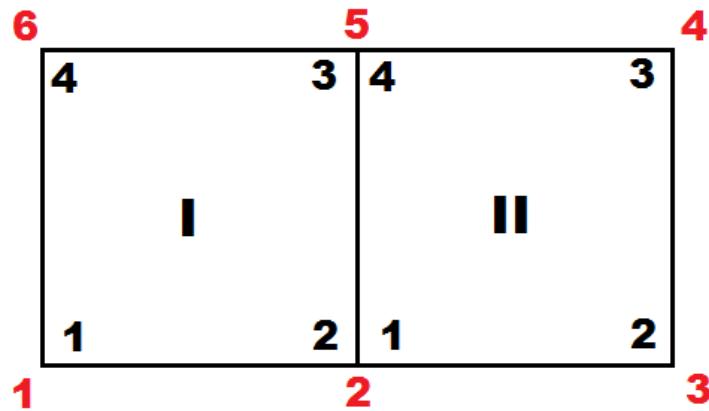
Metoda agregacji macierzy globalnych zostanie przedstawiona na przykładzie macierzy sztywności dwóch elementów kwadratowych i obiektu z nich zbudowanego. Wspomniany obiekt przedstawia rysunek 3.6. Czarne cyfry oznaczają numerację lokalną wewnętrz elmentu, a czerwone numerację globalną.

Algorytm agregacji polega na umieszczaniu odpowiednich elementów macierzy lokalnych do macierzy globalnej. Pokrywające się elmenty są sumowane. Ponieważ podmacierz sztywności dla jednego punktu lub zależności pomiędzy punktami jest umieszczana w macierzy globalnej bez wewnętrznych zmian, przyjmijmy zapis uproszczony:

$$\mathbf{K}_n^{ij} = \begin{bmatrix} K_{xx}^{ij} & K_{xy}^{ij} \\ K_{xy}^{ij} & K_{yy}^{ij} \end{bmatrix} \quad (3.27)$$

gdzie

- i, j – numery punktów
- n – numer elementu
- x, y – współrzędne punktów.



Rys. 3.6. Dwa elementy skończone kwadratowe tworzące obiekt

W takim wypadku lokalne macierze sztywności dla obydwu elementów możemy zapisać jako:

$$\mathbf{K}_I = \begin{bmatrix} \mathbf{K}_I^{11} & \mathbf{K}_I^{12} & \mathbf{K}_I^{13} & \mathbf{K}_I^{14} \\ \mathbf{K}_I^{21} & \mathbf{K}_I^{22} & \mathbf{K}_I^{23} & \mathbf{K}_I^{24} \\ \mathbf{K}_I^{31} & \mathbf{K}_I^{32} & \mathbf{K}_I^{33} & \mathbf{K}_I^{34} \\ \mathbf{K}_I^{41} & \mathbf{K}_I^{42} & \mathbf{K}_I^{43} & \mathbf{K}_I^{44} \end{bmatrix} \quad (3.28)$$

$$\mathbf{K}_{II} = \begin{bmatrix} \mathbf{K}_{II}^{11} & \mathbf{K}_{II}^{12} & \mathbf{K}_{II}^{13} & \mathbf{K}_{II}^{14} \\ \mathbf{K}_{II}^{21} & \mathbf{K}_{II}^{22} & \mathbf{K}_{II}^{23} & \mathbf{K}_{II}^{24} \\ \mathbf{K}_{II}^{31} & \mathbf{K}_{II}^{32} & \mathbf{K}_{II}^{33} & \mathbf{K}_{II}^{34} \\ \mathbf{K}_{II}^{41} & \mathbf{K}_{II}^{42} & \mathbf{K}_{II}^{43} & \mathbf{K}_{II}^{44} \end{bmatrix}. \quad (3.29)$$

Macierze te mają wymiar 8x8, co odpowiada czterem punktom i dwóm współrzędnym dla każdego punktu. Macierz globalna wobec tego będzie miała wymiar 12x12 (6x6 podmacierzy). Poniżej przedstawione jest wyznaczanie kilku elementów macierzy globalnej.

Dla globalnego punktu 1 mamy:

$$\mathbf{K}^{11} = \mathbf{K}_I^{11}, \quad (3.30)$$

ponieważ punkt globalny 1 pokrywa się z punktem 1 macierzy I.

Dla globalnego punktu 2 mamy:

$$\mathbf{K}^{22} = \mathbf{K}_I^{22} + \mathbf{K}_{II}^{11}, \quad (3.31)$$

ponieważ punkt globalny 2 pokrywa się z punktem 2 macierzy I oraz punktem 1 macierzy II.

Dla zależności globalnych punktów 2 i 3 mamy:

$$\mathbf{K}^{23} = \mathbf{K}_{II}^{12}, \quad (3.32)$$

ponieważ punkt globalny 2 pokrywa się z punktem 1, a punkt globalny 3 z punktem 2 macierzy II.

Dla zależności globalnych punktów 2 i 5 mamy:

$$\mathbf{K}^{25} = \mathbf{K}_I^{23} + \mathbf{K}_{II}^{14}, \quad (3.33)$$

ponieważ punkt globalny 2 pokrywa się z punktem 2 macierzy I oraz punktem 1 macierzy II, a punkt globalny 5 z punktem 3 macierzy I oraz punktem 4 macierzy II.

Uwzględniając że sztywność względna punktów nie będących ze sobą w sąsiedztwie (np. 1 w pierwszym elemencie oraz 3 w drugim elemencie) wynosi 0, macierz globalna przyjmuje postać:

$$\mathbf{K} = \begin{bmatrix} \mathbf{K}_I^{11} & \mathbf{K}_I^{12} & 0 & 0 & \mathbf{K}_I^{13} & \mathbf{K}_I^{14} \\ \mathbf{K}_I^{21} & \mathbf{K}_I^{22} + \mathbf{K}_{II}^{11} & \mathbf{K}_{II}^{12} & \mathbf{K}_I^{13}I & \mathbf{K}_I^{23} + \mathbf{K}_{II}^{14} & \mathbf{K}_I^{24} \\ 0 & \mathbf{K}_{II}^{21} & \mathbf{K}_{II}^{22} & \mathbf{K}_{II}^{23} & \mathbf{K}_I^{24}I & 0 \\ 0 & \mathbf{K}_I^{31}I & \mathbf{K}_{II}^{32} & \mathbf{K}_{II}^{33} & \mathbf{K}_{II}^{34} & 0 \\ \mathbf{K}_I^{31} & \mathbf{K}_I^{32} + \mathbf{K}_{II}^{41} & \mathbf{K}_I^{42}I & \mathbf{K}_{II}^{43} & \mathbf{K}_I^{33} + \mathbf{K}_{II}^{44} & \mathbf{K}_I^{34} \\ \mathbf{K}_I^{41} & \mathbf{K}_I^{42} & 0 & 0 & \mathbf{K}_I^{43} & \mathbf{K}_I^{44} \end{bmatrix} \quad (3.34)$$

Oczywistym jest, że w przypadku innej numeracji węzłów globalnych zmieni się układ macierzy globalnej. Nie ma to jednak wpływu na ostateczny wynik rozwiązań MES. W modelach o dużej liczbie elementów skończonych macierze mas i sztywności są macierzami rzadkimi. Wykorzystuje się to w obliczeniach do oszczędzania pamięci komputera, poprzez zapis w pamięci tylko wartości niezerowych oraz ich położenia w macierzy.

Macierz mas wyznaczona w przedstawiony sposób jest nazywana macierzą konsystentną. Często stosuje się macierze skupione, które zawierają elementy tylko na diagonali. Pozwala to bardzo przyspieszyć obliczenia. Takie rozwiązanie jest ściśle rzecz biorąc niepoprawne, ponieważ nie ma algorytmu, który pozwala obliczyć macierz skupioną i zachować w pełni właściwości modelu. Macierze skupione oblicza się np. poprzez sumowanie wszystkich elementów w wierszu i umieszczenie tej sumy na elemencie diagonalnym. Dobrą stroną macierzy skupionych jest fakt, że rozwiązanie pozostaje zbieżne.

3.5. Rozwiązywanie wyznaczonego równania macierzowego

Przyjmijmy, że wyznaczone równanie różniczkowe po agregacji macierzy ma postać:

$$\mathbf{M}\ddot{\mathbf{x}} + \mathbf{K}\mathbf{x} = \mathbf{F}. \quad (3.35)$$

Rozwiązać to równanie można na wiele sposobów. Przedstawione zostaną tutaj trzy. Pierwszy sposób to metoda całkowania jawnego, druga - metoda całkowania niejawnego, a trzecie to metoda modalna.

W metodzie całkowania jawnego zakładamy rozpoczęcie obliczeń w kroku czasowym t i przybliżamy pochodną drugiego rzędu poprzez formułę centralną:

$$\ddot{\mathbf{x}}^t = \frac{\mathbf{x}^{t+1} - 2\mathbf{x}^t + \mathbf{x}^{t-1}}{\delta t^2}. \quad (3.36)$$

Po podstawieniu do równania 3.39 wyznaczamy \mathbf{x}^{t+1} :

$$\mathbf{x}^{t+1} = \Delta t^2 \mathbf{M}^{-1} (\mathbf{F}^t - \mathbf{K}\mathbf{x}^t) + 2\mathbf{x}^t - \mathbf{x}^{t-1}. \quad (3.37)$$

Metoda ta charakteryzuje się tym, że do stabilności rozwiązywania często wymaga małego kroku czasowego. Nadaje się ona za to do zrównoleglania obliczeń, co pozwala zastosować do tego celu procesory graficzne. Operacja odwracania macierzy \mathbf{M} we wzorze 3.37 jest kosztowna obliczeniowo. Zastosowanie macierzy skupionej pozwala znacznie przyspieszyć tą procedurę.

Metoda całkowania niejawnego różni się wyjściowym krokiem czasowym. Zakładamy początkową chwilę czasową $t+1$, co przy zastosowaniu tej samej formuły różnicowej wprowadza krok wstecz.

$$\ddot{\mathbf{x}}^{t+1} = \frac{\mathbf{x}^{t+1} - 2\mathbf{x}^t + \mathbf{x}^{t-1}}{\delta t^2}. \quad (3.38)$$

Podstawiamy równanie 3.38 do równania

$$\mathbf{M}\ddot{\mathbf{x}}^{t+1} + \mathbf{K}\mathbf{x}^{t+1} = \mathbf{F}^{t+1} \quad (3.39)$$

i otrzymujemy

$$\mathbf{x}^{t+1} = (\mathbf{M} + \Delta t^2 \mathbf{K})^{-1} (\Delta t^2 \mathbf{F}^{t+1} + 2\mathbf{M}\mathbf{x}^t - \mathbf{M}\mathbf{x}^{t-1}). \quad (3.40)$$

W tym wypadku nie unikniemy już odwracania macierzy $\mathbf{M} + \Delta t^2 \mathbf{K}$, co powoduje wydłużenie obliczeń. Zaletą tej metody jest fakt, że jest ona bezwarunkowo stabilna.

W metodzie modalnej jako pierwsze należy wyznaczyć wektory własne dla równania jednorodnego $\mathbf{M}\ddot{\mathbf{x}} + \mathbf{K}\mathbf{x} = 0$. Wektory własne wyznaczyć można z dokładnością do stałej

multiplikatywnej. Możliwe jest wyskalowanie tych wektorów w taki sposób, aby maciesz tych wektorów ϕ miała własność:

$$\phi^T \mathbf{M} \phi = \mathbf{I}. \quad (3.41)$$

Zakładając przekształcenie współrzędnych $\mathbf{x} = \phi \bar{\mathbf{x}}$ otrzymamy nową postać równania 3.39.

$$\ddot{\bar{\mathbf{x}}} + \Lambda \bar{\mathbf{x}} = \bar{\mathbf{F}}, \quad \Lambda = \phi^T \mathbf{K} \phi, \quad \bar{\mathbf{F}} = \phi^T \mathbf{F}. \quad (3.42)$$

Macierz Λ jest macierzą diagonalną, więc rozwiązywanie układu sprowadza się teraz do znalezienia rozwiązania dla pojedynczych oscylatorów harmonicznych. Elementy macierzy Λ są kwadratami częstości własnych drgań węzłów.

$$\begin{bmatrix} \ddot{\bar{x}}_1 \\ \ddot{\bar{x}}_2 \\ \ddot{\bar{x}}_3 \\ \vdots \\ \ddot{\bar{x}}_n \end{bmatrix} + \begin{bmatrix} \omega_1^2 & & & & \\ & \omega_2^2 & & & 0 \\ & & \omega_3^2 & & \\ 0 & & & \ddots & \\ & & & & \omega_n^2 \end{bmatrix} \begin{bmatrix} \bar{x}_1 \\ \bar{x}_2 \\ \bar{x}_3 \\ \vdots \\ \bar{x}_n \end{bmatrix} = \begin{bmatrix} \bar{F}_1 \\ \bar{F}_2 \\ \bar{F}_3 \\ \vdots \\ \bar{F}_n \end{bmatrix} \quad (3.43)$$

$$\begin{cases} \ddot{\bar{x}}_1 + \omega_1^2 \bar{x}_1 = \bar{F}_1 & \rightarrow \begin{bmatrix} \bar{x}_1 \\ \bar{x}_2 \\ \bar{x}_3 \\ \vdots \\ \bar{x}_n \end{bmatrix} \\ \ddot{\bar{x}}_2 + \omega_2^2 \bar{x}_2 = \bar{F}_2 & \rightarrow \\ \ddot{\bar{x}}_3 + \omega_3^2 \bar{x}_3 = \bar{F}_3 & \rightarrow \\ \vdots & \vdots \\ \ddot{\bar{x}}_n + \omega_n^2 \bar{x}_n = \bar{F}_n & \rightarrow \end{cases} \quad (3.44)$$

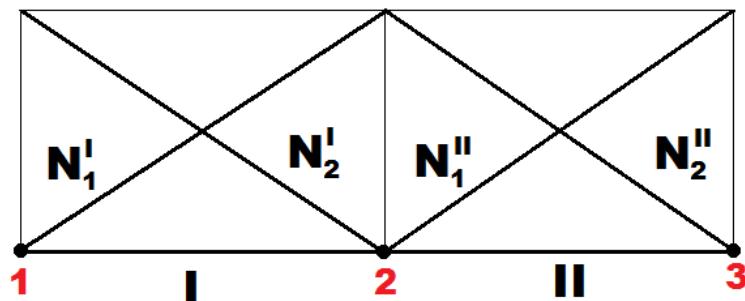
Po rozwiązaniu równań należy wyznaczyć ostateczne rozwiązanie $\mathbf{x} = \phi \bar{\mathbf{x}}$.

3.6. Zbieżność metody i błędy rozwiązania

O zbieżności rozwiązania możemy wnioskować na podstawie funkcji kształtu. Pierwszym warunkiem jest tzw. warunek zgodności. Mówią on o tym, że funkcje kształtu muszą być ciągłe w przestrzeni elementów. Dla prostego przypadku dwóch elementów liniowych jednowymiarowych ciągłość funkcji ilustruje rysunek 3.7.

Warunek ten jest zapewniony poprzez własność funkcji kształtu przedstawioną we wzorze 3.9.

Kolejny warunek nazywa się warunkiem bryły sztywnej (WBS). Zapewnia on brak straty bądź narastania energii podczas wyznaczania wyniku wewnątrz elementu. Warunek ten jest zapewniony poprzez własność funkcji kształtu przedstawioną w 3.10.



Rys. 3.7. Ciągłość funkcji w przestrzenii elementów

Ostatnim warunkiem jest tzw. warunek stałego odkształcenia (WSO). Mówiący on o tym, że jeśli w elemencie przyłożymy liniowe pole przemieszczeń, to odkształcenie będzie stałe w każdym punkcie elementu.

Jeśli spełnione są warunki zgodności, WBS oraz WBO to rozwiązanie dla elementu będzie zbieżne. Warunki te muszą być spełnione dla każdego elementu obliczanego modelu.

Zbieżność nie daje nam gwarancji, że rozwiązanie otrzymane przy pomocy symulacji MES jest poprawne. Przez poprawne rozumiane jest, że błąd rozwiązania jest dostatecznie mały. Aby mieć pewność, że rozwiązanie jest wystarczająco dokładne należy oszacować błąd maksymalny. W przypadku MES błąd powodują:

1. dyskretyzacja konstrukcji
2. zaokrąglenia arytmetyczne.

Pierwsza przyczyna występowania błędów związana jest z podziałem konstrukcji na elementy skończone. Błąd zawarty jest już w równaniu $\mathbf{M}\ddot{\mathbf{x}} + \mathbf{K}\mathbf{x} = \mathbf{F}$. Pojawia się dletem, że całe rozwiązanie wyznaczamy za pomocą wielomianowych funkcji kształtu. Błąd zbieżnego rozwiązania możemy zmniejszać poprzez wykorzystanie funkcji kształtu wyższego rzędu, bądź zagęszczenie siatki i budowę większej liczby elementów skończonych.

Estymacja błędów odbywa się na różne sposoby. Jednym z nich jest wykorzystanie równania 3.45.

$$F - \tilde{F}_i \approx c h_i^r \quad (3.45)$$

gdzie

F – rozwiązanie dokładne

\tilde{F}_i – i -te rozwiązanie przybliżone

c – współczynnik proporcjonalności

h_i – współczynnik zależny od zagęszczenia siatki

r – współczynnik zależny od stopnia wielomianów interpolujących.

Jedna symulacja nie daje możliwości wyznaczyć błędu. Aby tego dokonać należy przeprowadzić dwie symulacje, co pozwala obliczyć błąd drugiej (dokładniejszej). Przyjmijmy, że druga symulacja zawiera elementy o dwukrotnie mniejszych wymiarach, czyli $h_1 = 2h_2$. Współczynnik r przyjmuje wartości mniejsze od 1, ale dla uproszczenia przyjmijmy dokładnie 1.

Podstawiając odpowiednie indeksy do wzoru 3.45, możemy wyznaczyć błąd drugiej symulacji, znając wyniki zarówno pierwnej jak i drugiej.

$$F - \tilde{F}_2 \approx \frac{\tilde{F}_2 - \tilde{F}_1}{\left(\frac{h_1}{h_2}\right)^r - 1}. \quad (3.46)$$

Innym podejściem jest wyznaczanie błędów poprzez energię naprężenia elementów. Po wyznaczeniu przemieszczeń węzłów i obliczeniu naprężen, naprężenia nie są ciągle w przestrzeni elementów. Jeśli jeden węzeł należy do kilku elementów, to w każdym z nich może zostać wyznaczona inna wartość naprężenia dla takiego węzła. W ciągłym przypadku naprężenie byłoby funkcją ciągłą, dlatego błąd wynikający z naprężen wyznacza się w następujący sposób:

1. Obliczamy średnią naprężen w węźle, biorąc wartości dla węzła z każdego elementu, do którego należy.
2. Wyznaczamy błąd naprężenia w węzłach elementu, poprzez odejmowanie od naprężenia w węźle wartości średniego naprężenia w tym węźle.
3. Sumujemy błędy naprężen wszystkich elementów skończonych.
4. Wyznaczamy energię błędu naprężenia.
5. Wyznaczamy energię odkształcenia dla całej konstrukcji.
6. Obliczamy błąd procentowy według wzoru 3.47.

$$E = 100 \left(\frac{e}{U + e} \right)^{0.5} \quad (3.47)$$

gdzie

e – całkowita energia błędu dla konstrukcji

U – energia odkształcenia dla konstrukcji

E – błąd procentowy energii.

4. Metody kompensacji dyspersji

KATARZYNA RUGIEŁŁO

Istnieje wiele różnych metod kompensacji dyspersji w tej pracy szczegółowo opisane zostały wybrane trzy. Pierwsza polegająca na przygotowaniu sygnału, który sam skompensuje się do odpowiedniej postaci w trakcie propagacji o zadaną odległość. Druga bazująca na przybliżaniu krzywych dyspersji przy pomocy rozwinięcia w szereg Taylora, oraz trzecia polegająca na mapowaniu sygnału z dziedziny czasu na odległość

4.1. Cel kompensacji

4.1.1. Zastosowanie fal Lamba w nieniszczacych testach

Fale prowadzone akustyczne i ultradźwiękowe, w tym między innymi fale Lamba, są wykorzystywane na wiele sposobów między innymi do nieniszczących testów oraz oceny jakości obiektów. Stosowane są między innymi do diagnostyki uszkodzeń aktywnych lub pasywnych w monitorowaniu kondycji strukturalnej (SHM - Structural Health Monitoring). W niektórych przypadkach fale te są wykorzystywane do uzyskania lokalnych informacji o próbce, której to informacji nie można uzyskać konwencjonalnymi technikami ultradźwiękowymi. Przykładem takiego zastosowania może być inspekcja spoiwa klejowego [20]. Jest to przykład aplikacji o krótkim zasięgu, ponieważ odległość propagacji fal kierowanych jest stosunkowo mała. Drugi obszar zastosowania kontroli fal prowadzonych jest przeznaczony do testowania dalekiego zasięgu. W tym przypadku odległość propagacji jest stosunkowo duża. Ich główną zaletą, sprawiającą, iż są one tak często wykorzystywane do diagnozy urządzeń jest to, że mogą być wzbudzane przez elementy uruchamiające znajdujące się na lub wewnątrz struktury w jednym punkcie konstrukcji i mogą się rozprzestrzeniać na duże odległości. Konwencjonalna kontrola ultradźwiękowa dużych struktur jest bardzo czasochłonna, ponieważ testom musi zostać poddany każdy punkt badanej struktury, która ma być monitorowana. Wykorzystanie fal Lamba jest więc potencjalnie bardzo atrakcyjnym rozwiązaniem tego problemu. Jeżeli przetwornik odbierający znajduje się w odległym punkcie struktury, odebrany sygnał zawiera informacje o całej przebytej ścieżce propagacji między przetwornikami nadawczym i odbiorczym. W związku z tym test monitoruje całą ścieżkę, a nie pojedynczy punkt struktury. Pozwala to na znaczne

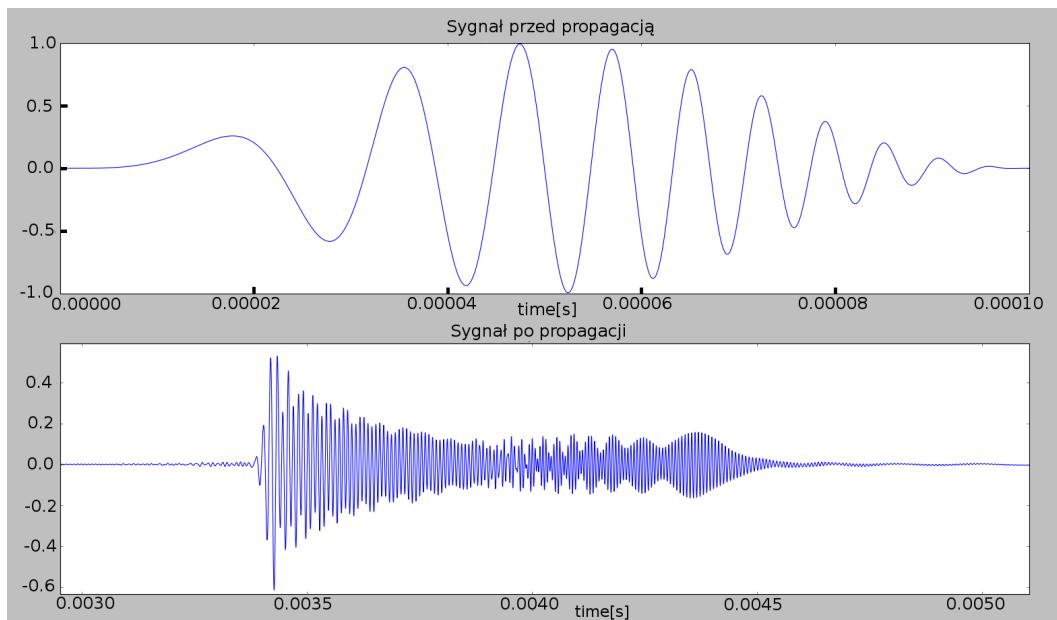
zaoszczędzenie czasu badań. W przypadku prętów niejednokrotnie stosuje się techniki, w których nadajnik po wyemitowaniu sygnału przełącza się w tryb odbiornika. Propagujący sygnał odbija się od końca pręta i wraca spowrotem, gdzie jest odbierany i może zostać poddany analizie. Takie podejście jest bardzo często stosowane, zwłaszcza w sytuacjach, w których dostęp do pręta możliwy jest tylko z jednej strony jak na przykład podczas badania kotw górniczych. Przykład ich położenia ilustruje rysunek 4.1



Rys. 4.1. Przykład zamontowania kotw górniczych, do których dostęp w przypadku testów jest tylko z jednej strony [2]

Nawet w najprostszych strukturach, takich jak swobodna płaska izotropowa płyta lub prosty stalowy pręt, może istnieć nieskończona liczba sterowanych trybów falowych. Co więcej, postaci te są generalnie dyspersywne. Obydwa te fakty utrudniają praktyczne zastosowanie fal kierowanych. W praktyce testy kontroli dalekiego zasięgu są wykonywane w sposób zadowalający, gdy stosuje się tylko jedną lub czasami dwie postaci fal prowadzonych, a pozostałe są tłumione. Tradycyjnie osiąga się to za pomocą specjalnych przetworników. Dzięki starannej kontroli częstotliwości i liczby falowej wzbudzenia, możliwe jest generowanie wybranych postaci fali Lamba oraz tłumienie pozostałych. Kontrola zakresu częstotliwości może być osiągnięta przez użycie sygnału o pewnej szerokości pasma zamkniętego w oknie Hanninga lub Gaussa. Zakres liczby falowej może być ograniczony przez użycie starannie zaprojektowanych sond EMAT lub za pomocą piezoelektrycznego przetwornika.

Powyższe metody mogą służyć do tłumienia sygnałów wywołanych przez niepożądane postaci fali, ale nie mogą zapobiec efektowi dyspersji, ponieważ zjawisko to występuje w falach kierowanych podczas ich propagacji w strukturze. Dyspersja sygnału powoduje, rozproszenie energii sygnału w czasie i przestrzeni w trakcie propagacji sygnału. W praktyce objawia się to wzrostem czasu trwania odbieranego sygnału w porównaniu do czasu trwania sygnału wejściowego. Rysunek 4.2 ilustruje przykład sygnału bez dyspersji oraz sygnału rozproszonego na skutek propagacji pewnej odległości. Łatwo zauważać, że sygnał przed propagacją trwa zaledwie 0,0001s natomiast po jego czas zwiększa się do około 0,0013s a zatem trwa ponad 10 razy dłużej.



Rys. 4.2. Przykład sygnału wejściowego, oraz sygnału rozproszonego

Zjawisko to pogarsza rozdzielczość i sprawia, że dane eksperymentalne są trudne do interpretacji z powodu nakładania się sygnałów. Używanie sygnałów wejściowych o określonej przepustowości może zmniejszyć problem dyspersji. Takie podejście koncentruje energię wejściową w ograniczonym zakresie częstotliwości, w którym jakiekolwiek zmiany prędkości pożądanego trybu fal kierowanych są małe. W praktyce oznacza to pracę w punktach na krzywych dyspersji dla konkretnego układu, w których prędkość grupowa jest stacjonarna lub prawie stacjonarna względem częstotliwości. Takie punkty zostały określone jako punkty „zerowej dyspersji” w [21], jest to określenie, które może wprowadzać w błąd, ponieważ niemożliwe jest skoncentrowanie całej energii sygnału wejściowego o skończonej długości na jednej częstotliwości. Zostało udowodnione [22], że istnieje optymalny sygnał wejściowy dla każdego punktu na krzywych dyspersji, który maksymalizuje rozdzielczość, jaką można uzyskać w tym punkcie. Jeśli jednak przyjmiemy, że przetwarzanie sygnału jest dozwolone przed wyświetlaniem informacji operatorowi, możliwe jest inne podejście do fal prowadzonych oraz ich roli w nieniszczących testach.

Należy również zaznaczyć, iż w praktyce rozproszone sygnały są zwykle uszkodzone z powodu szumów. Zachodzące na siebie i słabnące sygnały w porównaniu z szumem sprawiają, że dane z czujników są trudniejsze do interpretacji. W konsekwencji rozdzielczość może ulec pogorszeniu. Dlatego zrodziła się ogromna potrzeba opracowania technik przetwarzania sygnałów w celu zwiększenia rozdzielczości czasowej i amplitudy SNR.

Z punktu widzenia systemów SHM, kompresja sygnału lub usuwanie dyspersji w dziedzinie czasu może być wygodnym i intuicyjnym sposobem na łatwiejszą interpretację sygnałów czujnika.

4.1.2. Sygnał stosowany w symulacji

W ramach niniejszej pracy powstała aplikacja, pozwalająca użytkownikowi na podstawie podanych parametrów preta wygenerować krzywe dyspersji opisujące dany obiekt. Aplikacja pozwala również na wygenerowanie sygnału testowego, symulację jego propagacji w zadanym precie, oraz kompensację otrzymanego sygnału wybranymi metodami. Sygnałem stosowanym do testów był sygnał chirp zmodyfikowany przypomocy okna Hanninga. Sygnał o nazwie chirp jest sygnałem sinusoidalnym, w którym faza jest funkcją czasu. [23]. Sygnał o nazwie linear chirp to sygnał, w którym częstotliwość zmienia się w sposób liniowo zależny od czasu. Sygnał taki jest opisany wzorem:

$$s(t) = \sin(\phi(t)) \quad (4.1)$$

Gdzie $\phi(t)$ to funkcja fazy. Chwilową częstotliwość takiego sygnału jest związana funkcją fazy następującą zależnością:

$$f(t) = \frac{1}{2\pi} \frac{d(\phi(t))}{dt} \quad (4.2)$$

Aby zatem wygenerować sygnał linear chirp o rzadanych parametrach należy przyjąć, iż funkcja częstotliwości przybiera postać:

$$f(t) = f_0 + \frac{B}{T}t \quad (4.3)$$

Gdzie:

f_0 - częstotliwość początkowa

B - szerokość pasma częstotliwości

T - czas trwania sygnału

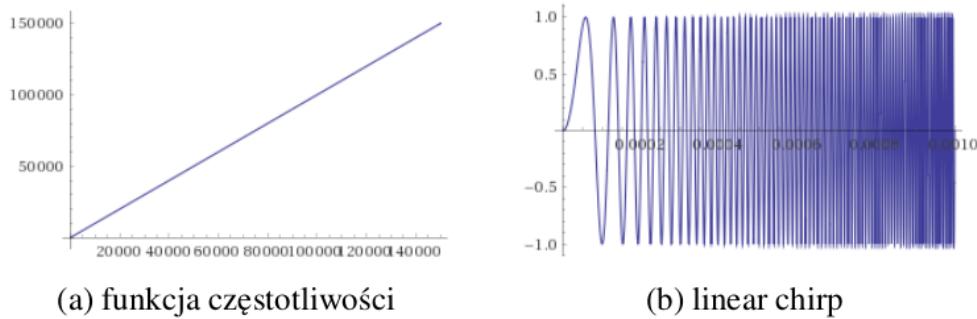
Łącząc równania (4.2) oraz (4.3) można funkcję fazy zapisać jako:

$$\phi(t) = 2\pi f_0 t + \frac{\pi B t^2}{T} \quad (4.4)$$

A zatem pełen wzór opisujący sygnał linear chirp można zapisać jako:

$$s(t) = \sin(2\pi f_0 t + \frac{\pi B t^2}{T}) \quad (4.5)$$

Przykład funkcji częstotliwości oraz uzyskanego w ten sposób sygnału przedstawia rysunek 4.3

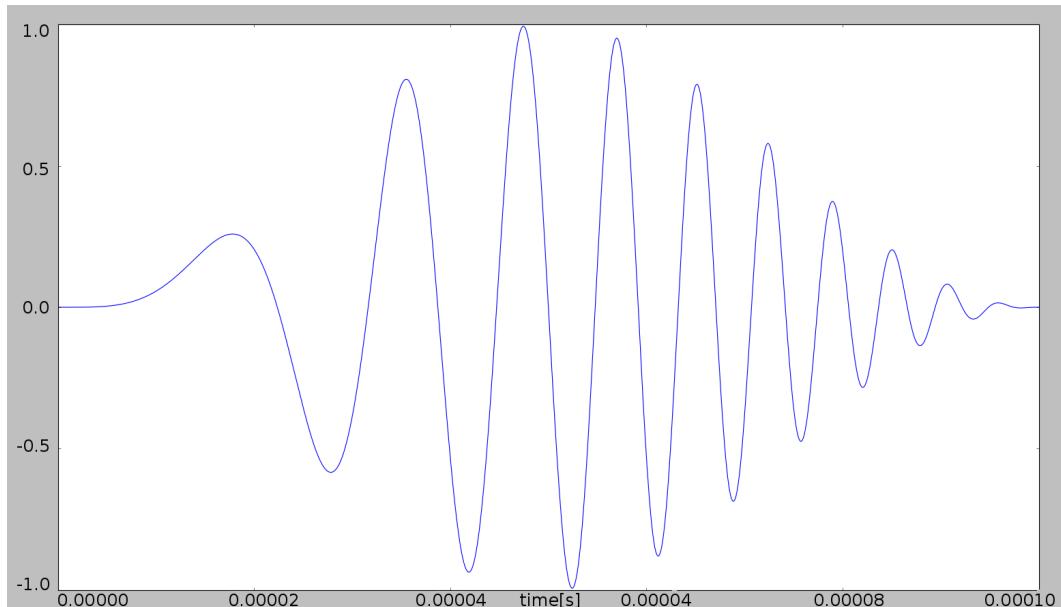


Rys. 4.3. Przykład sygnału linear chirp (b) oraz jego funkcji częstotliwości (a)

W prezentowanej pracy, sygnałem, który był poddawany symulacji był liniowy chirp dodatkowo pomnożony przez funkcję okna Hanninga daną wzorem:

$$w(n) = 0,5(1 - \cos(\frac{2\pi n}{N-1})) \quad (4.6)$$

Gdzie N to całkowita liczba próbek. uzyskany w ten sposób sygnał został zaprezentowany na rysunku 4.4



Rys. 4.4. Linear chirp pomnożony przez funkcję okna Hanninga

W aplikacji zaimplementowany został algorytm generujący opisany wyżej sygnał o następujących parametrach:

- czas trwania sygnału - 0.0001s
- częstotliwość minimalna - 0Hz
- częstotliwość maksymalna - 100kHz

Całość stworzona jest w oparciu o koncepcję otwartego kodu co daje użytkownikowi możliwość pisania własnych funkcji, generujących dowolne sygnały a następnie ich bezproblemową symulację w stworzonej aplikacji.

4.2. Agregacja krzywych dyspersji uzyskanych z zaimplementowanego solvera

Stworzona w ramach pracy aplikacja daje możliwość generowania zarówno krzywych dyspersji jak i krzywych wzbudzalności. Jednak ze względu na charakter zaimplementowanych algorytmów, tylko krzywe dyspersji wymagają agregacji, natomiast generowane krzywe wzbudzalności są od razu zagregowane.

4.2.1. Cel agregacji

Podstawą opracowania każdej z przedstawionych w tej pracy metod kompensacji jest znajomość krzywych dyspersji badanego obiektu, jakim jest stalowy pręt. Stanowią one pewnego rodzaju funkcję przejścia sygnału pomiędzy jednym punktem w czasie i przestrzeni a dowolnym innym punktem. Ich znajomość stanowi więc klucz do skompensowania efektu dyspersji. W stworzonej aplikacji, dzięki zastosowaniu odpowiednich algorytmów oraz na podstawie odpowiednich zadanych parametrów badanego obiektu, otrzymywane są poszukiwane krzywe dyspersji. Jednak dane są one w postaci chmury punktów zapisywanych w odpowiednich plikach. Aby funkcje te nadawały się do użycia należy każdy z wygenerowanych punktów przypisać do odpowiadającej krzywej. W ten sposób uzyskane zostaną funkcje, z których każda opisywać będzie zachowanie innej postaci fali prowadzonej. Rysunek 2.21 jest wykresem punktowym obrazującym wyniki otrzymane ze stworzonego solvera.

4.2.2. Algorytm agregacji

Krzywe dyspersji wyrażają zależność liczby falowej od częstości kątowej. Jako wynik w aplikacji otrzymujemy pojedynczy wektor zawierający kolejne wartości liczby falowej oraz zestaw odpowiadających danej liczbie częstości kątowych. Pierwszym krokiem w agregacji tak zapisanych danych jest zapisanie wszystkich danych w postaci chmury punktów o dwóch współrzędnych $A = (\omega, k)$ oraz posortowanie ich rosnąco wartościami ω . Liczba wygenerowanych krzywych odpowiada ilości punktów posiadających tę samą współrzędną k . Ponieważ są to wartości własne równania (2.44) to dla każdej wartości k jest ich tyle samo. Każda krzywa składa się z dokładnie takiej liczby punktów jaka jest długość wektora k . Kolejnym krokiem jest przyporządkowanie pierwszych dwóch punktów do każdej z krzywych. Mając punkty uporządkowane względem ω wystarczy wybrać

te o najmniejszej wartości k . Każdy z nich odpowiada kolejnemu trybowi fali. Punkty przydzielone do właściwego trybu zostają usunięte ze zbioru punktów do przydzielenia. Analogicznie postępujemy w przypadku drugiej grupy punktów. Wybieramy te o najniższej wartości k i przypisujemy do poszczególnych trybów. Agregacja kolejnych punktów musi przebiegać według innego schematu, ponieważ krzywe się wzajemnie przecinają i segregacja względem częstotliwości nie przyniesie zadowalających rezultatów. Przyjmując, że dla dowolnego modu, ostatni zagregowany punkt ma współrzędne $P_l = (\omega_l, k_l)$ z chmury punktów wybieramy zbiór potencjalnych punktów spełniających następujące warunki:

1. Wartość k potencjalnych punktów musi wynosić $k = k_p = v_k[l + 1]$

Gdzie k_p oznacza wartość k potencjalnych punktów, v_k oznacza posortowany rosnąco wektor wartości k , a $l + 1$ to indeks wartości z wektora v_k o jeden większy niż indeks ostatnio zagregowanego do danego trybu punktu

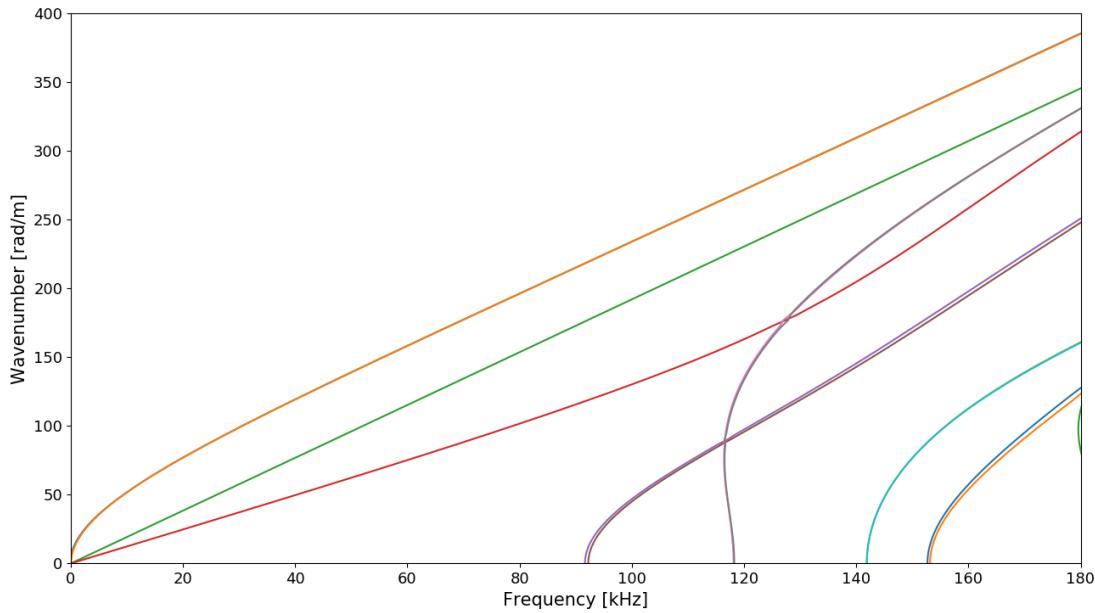
2. Wartość ω potencjalnych punktów musi znajdować się w pewnym, ograniczonym otoczeniu wartości ω_l

Ze zbioru potencjalnych punktów, które spełniają wyżej wymienione warunki należy następnie wybrać ten, który w najlepszy sposób będzie pasował do powstałego już fragmentu krzywej dyspersji. W celu wybrania najlepszego punktu obliczony zostaje kąt pomiędzy wektorami $\vec{v}_1 = \overrightarrow{P_{l-1}P_l}$ oraz $\vec{v}_2 = \overrightarrow{P_lP_{p_i}}$, gdzie P_{p_i} oznacza i -ty potencjalny punkt ze zbioru. Sformułowanie najlepiej pasujący punkt oznacza, iż kąt pomiędzy rozważanymi wektorami, obliczany ze wzoru:

$$\alpha = \arccos \frac{\vec{v}_1 \circ \vec{v}_2}{|\vec{v}_1| \cdot |\vec{v}_2|} \quad (4.7)$$

ma wartość najbliższą zeru. Usuwając zagregowane już punkty ze zbioru punktów do przydzielenia oraz postępując w sposób analogiczny do przedstawionego schematu, wszystkie punkty ze zbioru zostają przydzielone odpowiedniej krzywej. Wyniki agregacji zostały przedstawione na rysunku 4.5

Jak łatwo zauważyc opisany algorytm w sposób efektywny agreguje chmurę punktów do odpowiednich krzywych. Uporządkowane w ten sposób punkty zostały wykorzystane do opracowania trzech metod kompensacji, opisanych w kolejnych sekcjach tego rozdziału.



Rys. 4.5. Krzywe dyspersji po agregacji

4.3. Metoda odwracania sygnału w czasie

Metoda omawiana w tej sekcji polega na wygenerowaniu sygnału który po przepropagowaniu pewnej odległości sam się skompensuje. Technika kompensacji dyspersji poprzez wygenerowanie sygnału odwróconego w czasie została zaprezentowana w artykule [24]

4.3.1. Podstawy teoretyczne

Podstawową ideą prezentowanej w tej części pracy techniki kompensacji dyspersji jest wytworzenie sygnału, który poprzez nałożenie składowych częstotliwości w czasie propagacji skompensuje się, tworząc oczekiwany sygnał w pozycji pomiarowej. W przypadku w którym do obiektu wprowadzony zostanie zwykły sygnał sinusoidalny o liniowo zmieniającej się częstotliwości (linear-chirp) zostanie wzbudzone przynajmniej kilka trybów fali prowadzonej. Każdy tryb w każdej z częstotliwości może poruszać się z różną prędkością fazową i grupową. Ze względu na dyspersję sygnał wraz z propagacją będzie stawał się coraz dłuższy a jego amplituda będzie spadać, ponieważ ulegnie on rozproszeniu. Składowe częstotliwości sygnału odpowiadające najwyższej prędkością grupowym będą znajdować się z przodu, natomiast wolniejsze komponenty będą znajdować się z tyłu. Podstawa przedstawianej metody polega na założeniu, że tak otrzymany sygnał jesteśmy w stanie odwrócić w czasie, tak aby komponenty o mniejszej prędkości znalazły się z przodu sygnału a komponenty o większej prędkości znalazły się z tyłu. Wzbudzenie obiektu tak przygotowanym sygnałem da w odpowiedzi sygnał skompensowany tej samej postaci co pierwotne wzbudzenie. Skuteczność metody można wykazać analitycznie. Przyjijmy, że

sygnał zastosowany do wzbudzenia pręta zostanie zastosowany w punkcie $x = 0$ i będzie to sygnał $f(t)$. Sygnał wzbudzający w dziedzinie częstotliwości możemy zapisać jako:

$$[F(\omega)_{x=0}] = \int_{-\infty}^{\infty} f(t) e^{-i\omega t} dt \quad (4.8)$$

Jeśli przez nasz obiekt propaguje jedna postać fali to w punkcie $x = L$ można tę funkcję zapisać jako:

$$[F(\omega)_{x=L}] = \int_{-\infty}^{\infty} f(t) e^{-i(\omega t - kL)} dt \quad (4.9)$$

gdzie liczba falowa k jest funkcją częstotliwości ω , która jest opisana za pomocą krzywej dyspersji dla danego trybu fali. Równania (4.8) oraz (4.9) można powiązać funkcją przejścia $H(\omega)$ co można zapisać jako:

$$H(\omega) = \frac{[F(\omega)]_{x=L}}{[F(\omega)]_{x=0}} \quad (4.10)$$

Zakładając, że chcemy uzyskać sygnał $g(t)$, którego transformata Fouriera to $G(\omega)$ przy ($x = L$) wtedy wymagany sygnał wejściowy ($x = 0$) $Y(\omega)$:

$$Y(\omega) = \frac{G(\omega)}{H(\omega)} = G(\omega) [H(\omega)]^{-1} \quad (4.11)$$

Sygnał w dziedzinie czasu $y(t)$ jaki musi zostać wprowadzony do badanego pręta w punkcie $x = 0$ aby uzyskać odebrany sygnał $g(t)$ w punkcie $x = L$ można uzyskać z odwrotnej transformaty Fouriera, co pokazuje równanie (4.11):

$$y(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \frac{G(\omega)}{H(\omega)} e^{i\omega t} d\omega \quad (4.12)$$

W prostym przypadku gdy analizujemy pojedynczą postać fali prowadzonej:

$$[H(\omega)]^{-1} = e^{-ikL} = e^{-i\omega L/c} \quad (4.13)$$

$$y(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} G(\omega) e^{-i\omega(L/c-t)} d\omega \quad (4.14)$$

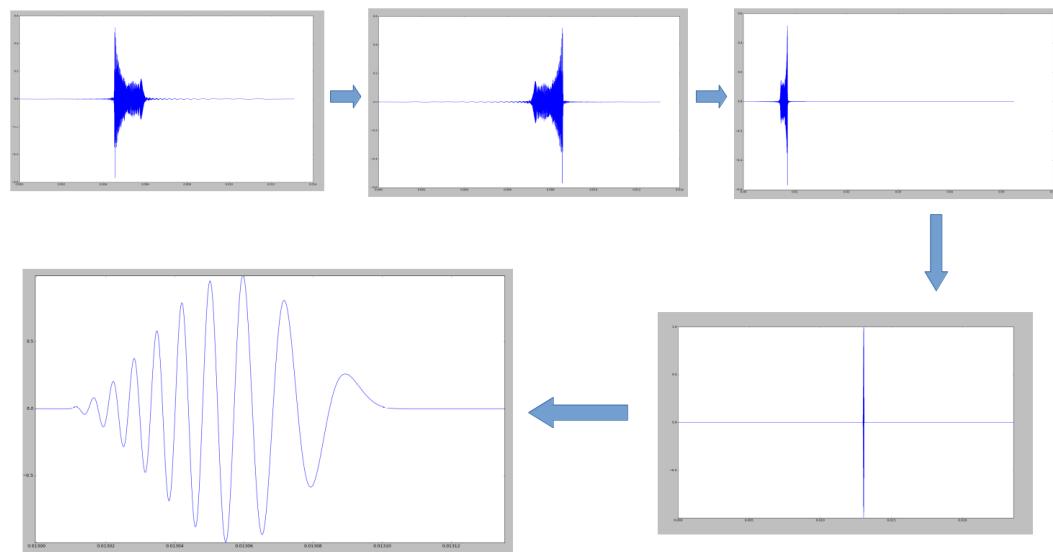
Przyjmując, że sygnał w domenie czas $g(t)$ jest aplikowany w punkcie $x = 0$ sygnał $y^*(t)$ w punkcie $x = L$ można zapisać jako:

$$y^*(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} G(\omega) e^{-i\omega(L/c+t)} d\omega \quad (4.15)$$

Technika kompensacji w tym przypadku polega na odebraniu sygnału $y^*(t)$ w przedziale czasowym $t \in [0, T]$ w którym cała paczka falowa zostanie odebrana przez odbiornik. Następnie sygnał zostaje odwrócony w czasie. Otrzymany w ten sposób sygnał może być zaaplikowany do badanego pręta. W trakcie propagacji zostanie on skompensowany do fali o kształcie takim jak kształt sygnału $g(t)$

4.3.2. Implementacja numeryczna

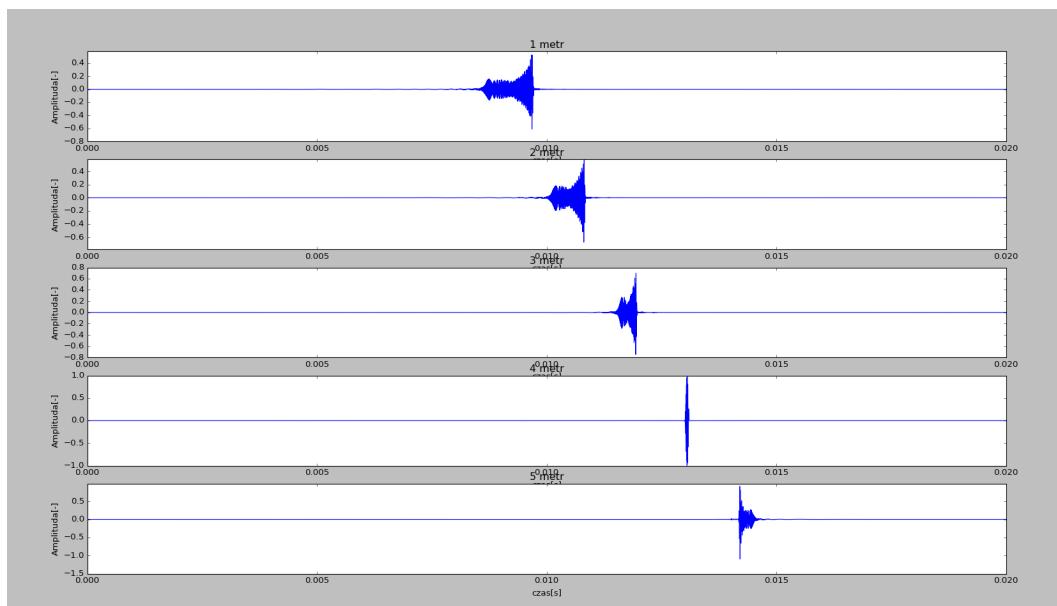
W ramach niniejszej pracy, omawiana metoda została zaimplementowana do aplikacji. Podstawa jej działania opiera się na znajomości krzywych dyspersji. Pierwszym etapem jest wygenerowanie sygnału, jaki chce się uzyskać w wyniku kompensacji. Następnie dzięki zaimplementowanym metodom należy uzyskać przewidywany kształt sygnału po przepropagowaniu zadanej odległości oraz odrócenie go w czasie. Sygnał otrzymany z tak przygotowanego sygnału wejściowego powinien skompensować się na zadanej odległości. Rysunek 4.6 przedstawia opisane kroki na przykładzie sygnału linear chirp.



Rys. 4.6. Algorytm kompensacji odwracania czasu

W zaprezentowanym przykładzie odwrócony w czasie sygnał został dodatkowo wypełniony zerami. Zabieg ten został zastosowany, ponieważ całość została przeprowadzona w ramach symulacji i wypełnienie sygnału było konieczne do przeprowadzenia prawidłowej symulacji propagacji. Przedstawiana metoda pozwala zatem wygenerować sygnał

kompensujący się na zadanej odległości do fali o oczekiwany kształcie. Pozwala na propagowanie zarówno pojedynczego trybu jaki nałożonych wielu postaci fali prowadzonej. Jednak aby móc stosować tę metodę konieczna jest znajomość długości ścieżki propagacji sygnału. Jeżeli ściażka propagacji ulegnie zmianie, konieczne jest wygenerowanie nowego sygnału adekwatnego do aktualnego badanego obiektu. W przypadku w którym podczas propagacji następowaliby dodatkowe odbicia, na przykład w sytuacji, w której fala propagującą przez dwa pręty złączone ze sobą, część energii zostanie odbita od punktu złączenia a część przeprowadza przez drugi pręt i odbije się na końcu. Stworzenie sygnału kompensującego się w takiej sytuacji byłoby bardziej skomplikowane i nie zostało opisane w tej pracy. Rysunek 4.9 prezentuje przykład fali przygotowanej do kompensacji po 4 metrach propagacji, który przeprowadzał kolejno 1,2,4 i 5 metrów.

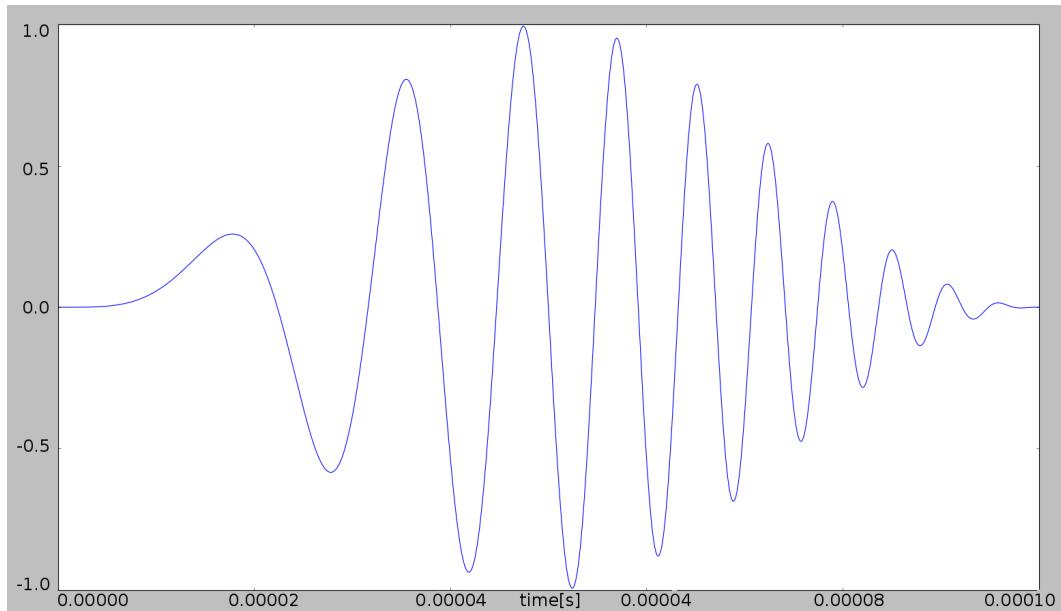


Rys. 4.7. Sygnał przygotowany do kompensacji po 4 metrach propagacji przedstawiony kolejno po 1, 2, 4 i 5 metrach propagacji

Jak łatwo zauważyc wprowadzony sygnał kompensuje się tylko i wyłącznie po przeprowadzaniu założonej odległości. Wraz z oddalaniem się od punktu, w którym przewidziany był odbiornik efekt dyspersji się nasila. Tak więc zarówno przed jak i za założonym punktem odbioru odebrany sygnał byłby rozproszony. Metoda ta może służyć do przeprowadzania nieniszczących testów obiektów o znanej charakterystyce przejścia. W sytuacji w której w strukturze nie będzie żadnych uszkodzeń otrzymywany sygnał będzie skompensowany. Natomiast gdy pojawią się uszkodzenia, nieciągłości w strukturze spowodują odbicie fali i skrócenie ścieżki propagacji co da informację o uszkodzeniach.

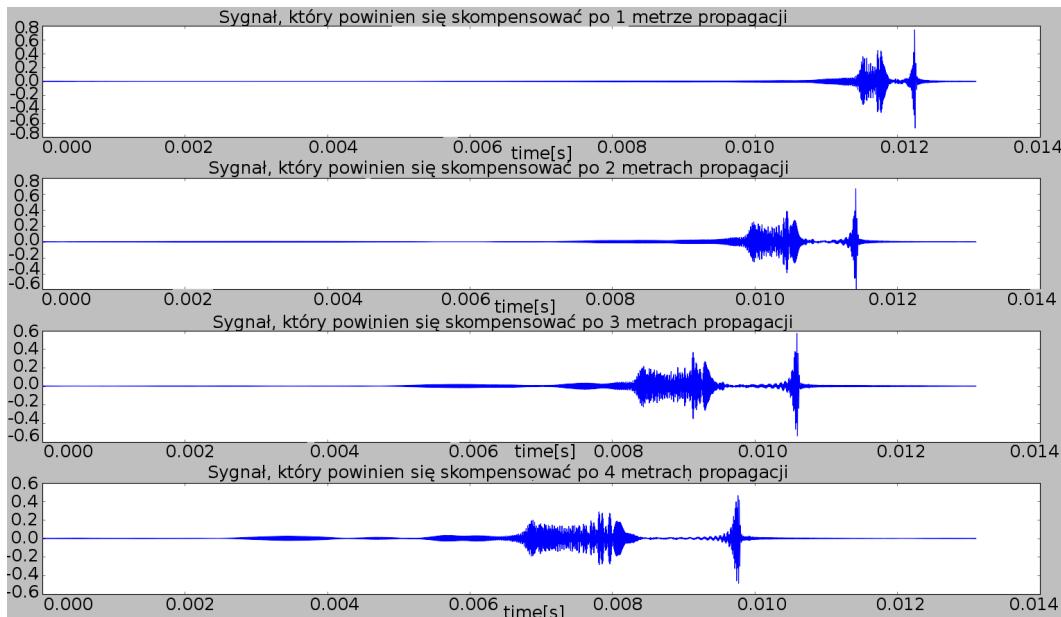
4.3.3. Wybrane wyniki z symulacji

Rysunek 4.8 przedstawia sygnał przekazany do funkcji, jako ten, do którego sygnał ma się skompensować po przeprowadzaniu zadanej odległości. Wygenerowany przez



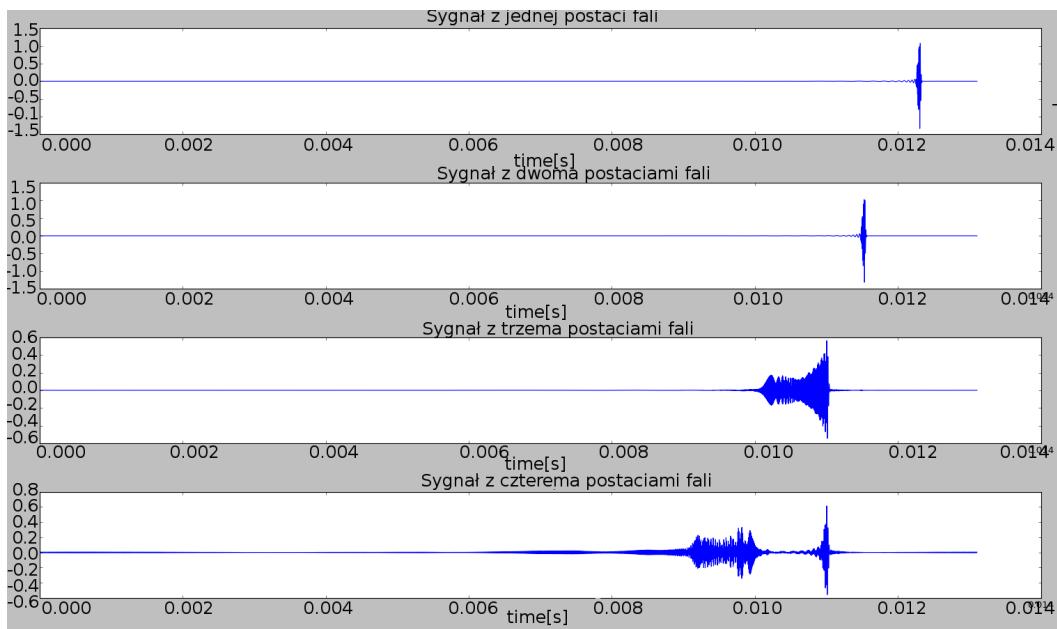
Rys. 4.8. Algorytm kompensacji odwracania czasu

aplikację sygnał jest różny dla różnych zadanych odległości. Obrazuje to rysunek 4.9 na którym przedstawiono sygnały mające się skompensować odpowiednio po 1, 2, 3, 4 metrach, gdy propagują pierwsi cztery postaci fali prowadzonej. Sygnał jest też różny



Rys. 4.9. Sygnał przygotowany do kompensacji po odpowiednio 1,2,3 oraz 4 metrach

dla różnej ilości propagujących trybów, rysunek 4.10. Jak łatwo zauważyc, im większa ilość propagujących postaci tym bardziej widoczne staje się zjawisko dyspersji i tym dłuższy staje się przepropagowany sygnał. Tak wygenerowane sygnały zostały przetestowane przez propagację w symulacji. Rysunek 4.11 ilustruje jak ważna jest znajomość długości ścieżki propagacji. Jeśli sygnał przebędzie zbyt krótką lub zbyt długą drogę to odebrany

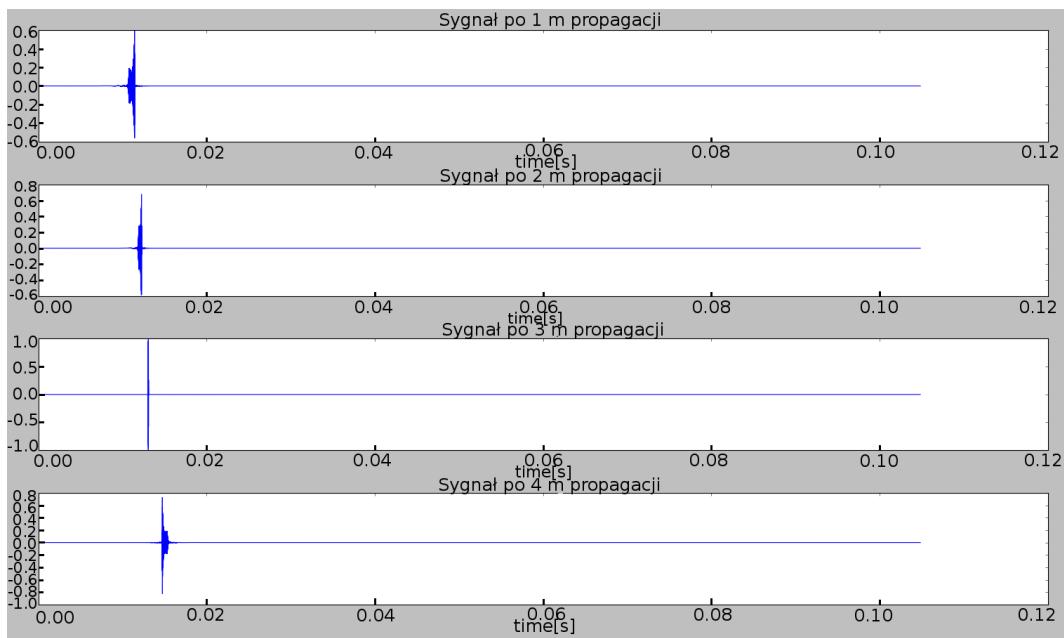


Rys. 4.10. Sygnał przygotowany do kompensacji po 2,5 metra propagacji zawierający kolejno 1,2,3 oraz 4 postaci fali prowadzonej

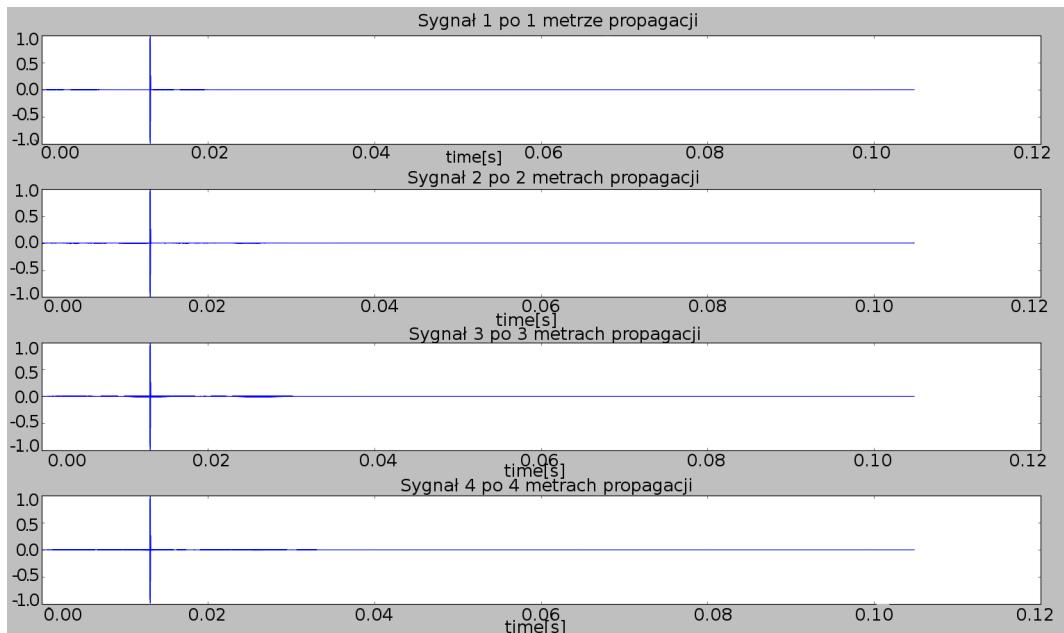
sygnał nie będzie skompensowany. Rysunek 4.12 ilustruje wyniki symulacji propagacji na odpowiednie odległości sygnałów przedstawionych na rysunku 4.9

Rysunek 4.13 ilustruje wyniki symulacji propagacji sygnałów przedstawionych na rysunku 4.10. Łatwo zauważać, że wyniki symulacji dają bardzo dobre rezultaty.

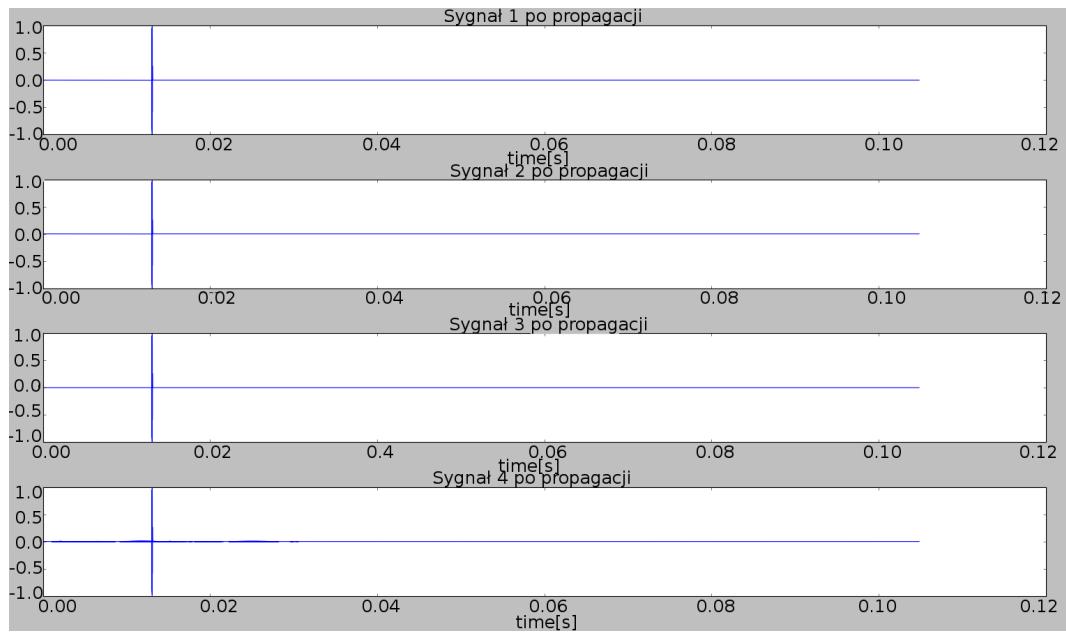
Rysunek 5.28 przedstawia porównanie wyniku symulacji sygnału linear chirp po propagacji dwóch metrów w przecie w którym propagowały trzy pierwsze tryby fali prowadzonej (kolor niebieski) oraz sygnału wygenerowanego w aplikacji, przygotowanego do kompensacji po dwóch metrach. Łatwo zauważać, że przedstawiana metoda pozwala znaczco skrócić czas trwania odbieranego sygnału. Sygnał bez kompensacji trwa 6e-4s natomiast z 1e-4 czyli dokładnie tyle ile żądany sygnał. A zatem można stwierdzić iż kompensacja została skutecznie przeprowadzona. Kolejny rysunek przedstawia porównanie sygnału założonego na wejściu, jako postać do której powinien on zostać skompensowany, oraz sygnału faktycznie otrzymanego po symulacji propagacji. Czas trwania oraz obwiednia sygnału zgadzają się z założeniami, jednak widać iż otrzymany sygnał jest sygnałem odwróconym w czasie w stosunku do założonego sygnału. Wynik nie jest zatem idealnym odwzorowaniem założeń, niemniej jednak otrzymany sygnał został maksymalnie skompensowany. W otrzymanym sygnale nie ma również widocznych szumów, pogarszających czytelność otrzymanego wyniku. Oczywistym jest fakt, iż w przypadku rzeczywistym szumy z pewnością by występowały. Idealne matematyczne odwzorowanie padanego obiektu nie jest możliwe, lecz mimo to metoda powinna dawać dobre rezultaty.



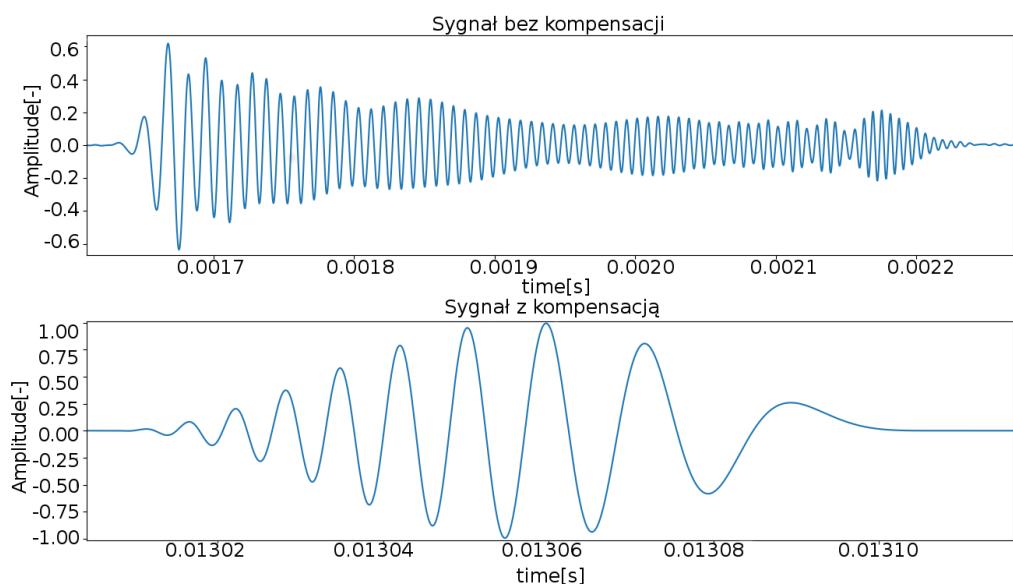
Rys. 4.11. Sygnał przygotowany do kompensacji po 3 metrach propagacji przepropagowany kolejno o 1,2,3 i 5 metrów



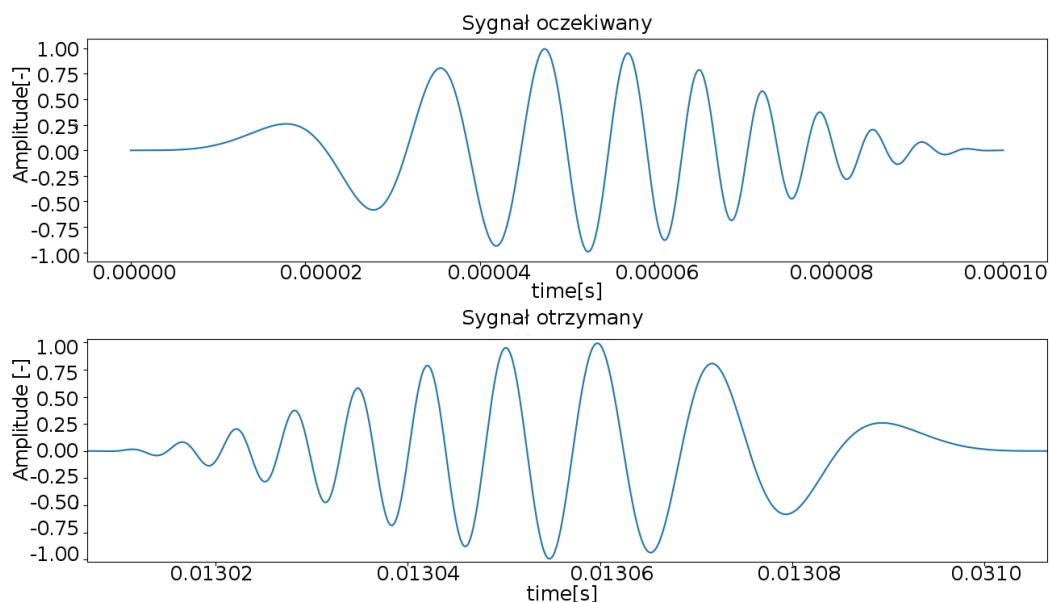
Rys. 4.12. Sygnały z rysunu 4.9 po przepropagowaniu odpowiednich odległości



Rys. 4.13. Sygnały z rysunu 4.10 po przepropagowaniu odpowiedniej odległości



Rys. 4.14. Porównanie sygnału bez kompensacji(górny) oraz z kompensacją (dolny)



Rys. 4.15. Porównanie sygnału oczekiwanej (górny) otrzymanego (dolny)

4.4. Metoda mapowania liniowego przy pomocy rozwinięcia w szereg Taylora

Prezentowana w poniższej sekcji metoda kompensacji dyspersji została przedstawiona w artykule [25]

4.4.1. Podstawy teoretyczne

Po wzbudzeniu badanego pręta odpowiednim sygnałem wejściowym, możliwe jest aby nadajnik przełączył się w tryb obioru sygnału i nasłuchiwał nadejścia odpowiedzi układu. W takim wypadku odpowiedź jaką uzyskamy będzie sumą odpowiedzi ze wszystkich odbić, od końca pręta oraz ewentualnych łączeń lub uszkodzeń. Proponowana w tym rozdziale metoda opiera się na założeniu, że w badanym obiekcie propaguje jedna wybrana postać drgań. Jej celem jest kompensacja powstałej dyspersji, tak aby sygnały z różnych punktów odbicia nie nachodziły na siebie i była możliwa ich interpretacja w celu ustalenia ilości punktów odbicia oraz oszacowania ich odległości od miejsca wzbudzenia na podstawie znajomości prędkości grupowej fali. Przy tak sformułowanych założeniach, sygnał otrzymany w odbiorniku można przedstawić wzorem:

$$g(t) = \sum_{n=1}^N N f_n(r_n, t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega) \sum_{n=1}^N (A_n(\omega) e^{-ikr_n}) e^{i\omega t} d\omega \quad (4.16)$$

Gdzie:

N - całkowita liczba ścieżek propagacji sygnału (liczba punktów odbicia)

r_n - długość n-tej ścieżki propagacji

A_n - współczynnik odbicia n-tego punktu odbicia.

Widmo częstotliwości takiego sygnału $G(\omega)$ możemy obliczyć przy pomocy transformaty Fouriera i zapisać wzorem:

$$G(\omega) = F(\omega) \sum_{n=1}^N (A_n(\omega) e^{-ikr_n}) \quad (4.17)$$

Jak wiadomo, dyspersja zależy od kształtu krzywej dyspersji, $k = K(\omega)$. Jeśli więc $K(\omega)$ jest funkcją kwadratową lub wyższego rzędu względem ω , to $G(\omega)$ reprezentuje widmo częstotliwości rozproszonych pakietów fal. Jeśli natomiast zależność $K(\omega)$ byłaby funkcją liniową, zjawisko dyspersji nie występowałoby, a $G(\omega)$ reprezentowałoby widmo częstotliwości sygnału, który nie uległ dyspersji. Wybraną krzywą dyspersji można przybliżyć przy pomocy rozwinięcia w szereg Taylora:

$$k = K(\omega) - k_0 + k_1(\omega - \omega_0) + k_2(\omega - \omega_0^2) + \dots \quad (4.18)$$

Gdzie:

$$k_0 = \frac{\omega_0}{c_p}$$

$$k_1 = \frac{dk}{d\omega} \Big|_{\omega=\omega_0}$$

$$k_2 = \frac{1}{2} \frac{d^2k}{d\omega^2} \Big|_{\omega=\omega_0}$$

c_p - prędkość fazowa

Wynika z tego, iż dyspersję sygnału można usunąć usuwając jej nieliniowy składnik, poprzez zastąpienie oryginalnej zależności $K(\omega)$ liniowym przybliżeniem tej funkcji. Zastosowanie takiego zabiegu sprawi, iż sygnał w dziedzinie czasu będzie postrzegany jako skompensowany do postaci niedyspersyjnej, a prędkość grupowa obliczona z liniowego przybliżenia krzywej dyspersji może zostać wykorzystana do określenia długości ścieżek propagacji wynikających z kolejnych odbić.

W szczególnym przypadku, w którym mamy do czynienia z pojedyńczą ścieżką propagacji ($N = 1$) i odległość między nadajnikiem a punktem odbicia jest znana, można usunąć dyspersję poprzez wyeliminowanie wyrażenia kwadratowego w $K(\omega)$. Matematycznie można to zrobić przy pomocy wzoru:

$$\tilde{G}(\omega) = (F(\omega)A_1(\omega)e^{-ikr_1})e^{ik_2(\omega-\omega_0)^2r_1} \quad (4.19)$$

Gdzie $\tilde{G}(\omega)$ jest zmodyfikowanym widmem częstotliwości, a r_1 jest odległością między nadajnikiem i odbiornikiem. Sygnał skompensowany w dziedzinie czasu można natychmiast uzyskać poprzez odwrotną transformatę Fouriera. Ponieważ jednak zazwyczaj długość ścieżki propagacji sygnału nie jest znana, a sygnał może składać się z wielu odbić, między innymi od uszkodzeń ($N > 1$) usuwanie dyspersji przy pomocy powyższego wzoru byłoby niepraktyczne.

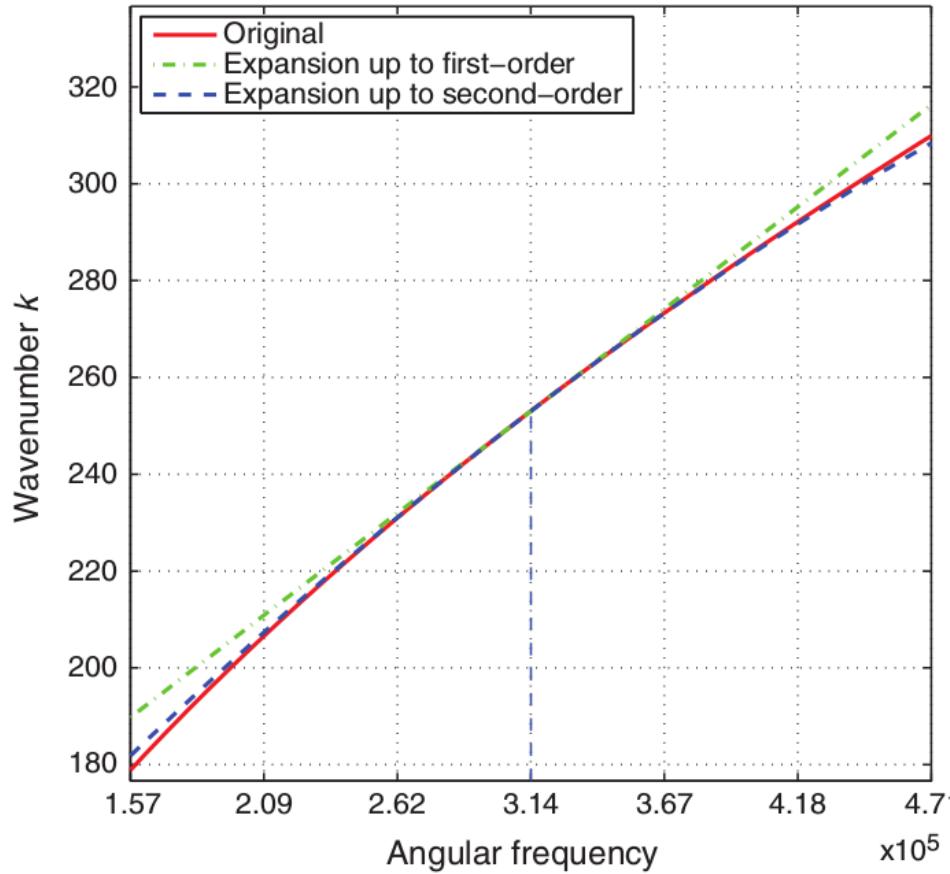
Rysunek 4.16 obrazuje przykład krzywej dyspersji, trybu A_0 , płyty aluminiowej o grubości 3,175 mm. Linia ciągła pokazuje zależności $K(\omega)$ uzyskaną metodami komputerowymi, natomiast linia przerywana kropkowana wskazuje z rozwinięcie szeregu Taylora pierwszego rzędu, a linia przerywana rozszerzenie szeregu Taylora drugiego rzędu w odniesieniu do centralnej częścią kątowej ($\omega_0 = 2\pi * 50kHz$)

Łatwo zauważać, że po pierwsze, rozszerzenie drugiego rzędu daje bardzo dobrze przybliżenie pierwotnego kształtu krzywej, po drugie, $K(\omega)$ jest monotoniczną funkcją ω w otoczeniu ω_0 .

Równanie 4.17 można zapisać w postaci złożenia funkcji:

$$G(\omega) = G(k) \circ K(\omega) \quad (4.20)$$

Gdzie \circ jest operatorem składania funkcji. $G(k)$ jest niejawną funkcją k z równania 4.17. Zamieniając $K(\omega)$ na $K_{lin}(\omega)$ będące aproksymacją $K(\omega)$ pierwszego rzędu w punkcie ω_0 , gdzie ω_0 oznacza częstotliwość o najwyższej energii, można wyprowadzić



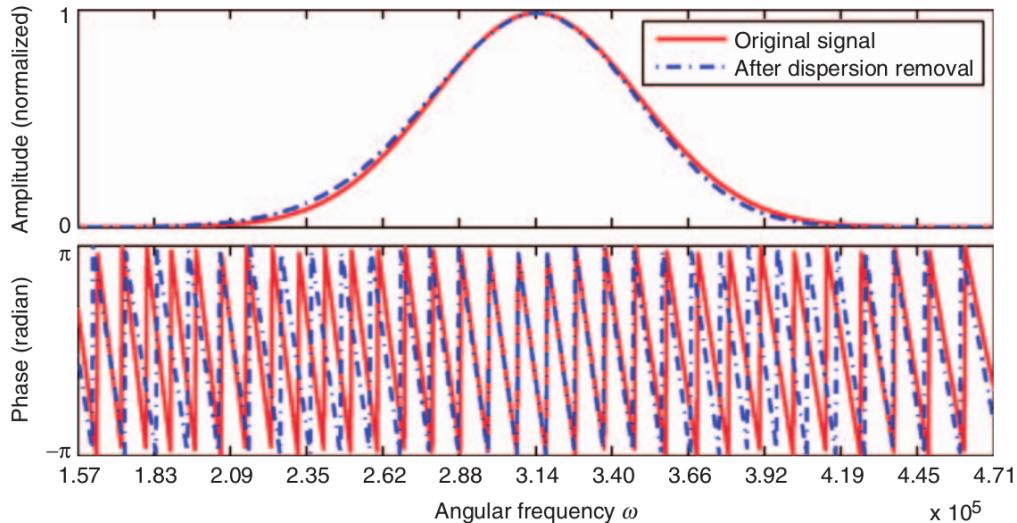
Rys. 4.16. Przykładowe porównanie oryginalnej krzywej oraz jej przybliżeń przy pomocy rozwinięcia w szereg Taylor'a [25]

zmodyfikowane widmo częstotliwości:

$$\tilde{G}(\omega) = G(k) \circ K_{lin}(\omega) \quad (4.21)$$

Ponieważ $G(\omega)$ jest znane oraz znana jest analizowana krzywa dyspersji, znane jest również $G(k)$, obliczając przybliżenie liniowe $K_{lin}(\omega)$ można w prosty sposób obliczyć $\tilde{G}(\omega)$, poprzez interpolację odpowiednich wartości. Opisana metoda może być określana mianem mapowania liniowego. W jej efekcie uzyskane zostaje nowe widmo częstotliwości $\tilde{G}(\omega)$. Po mapowaniu widmo amplitudy pozostaje bez zmian, natomiast widmo fazy stopniowo odbiega od pierwotnego w miarę oddalania się od wybranej, średniej częstotliwości, co dobrze ilustruje rysunek 4.17

Należy zaznaczyć, iż stosowanie omawianej metody jest możliwe tylko w sytuacji, gdy $K(\omega)$ jest funkcją monotoniczną



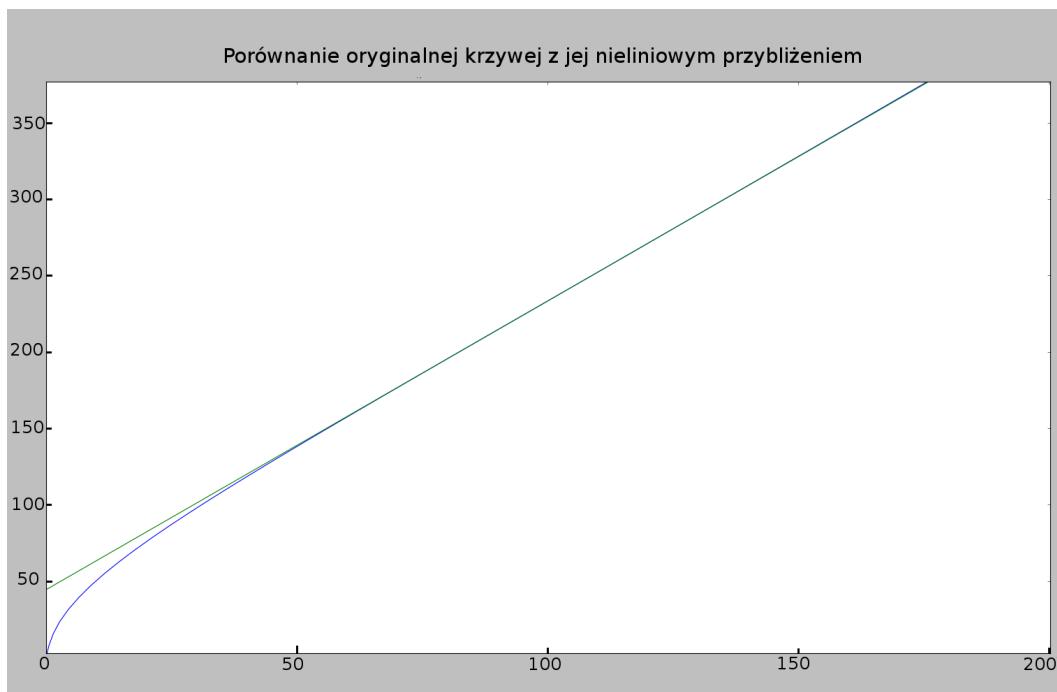
Rys. 4.17. Przykładowe porównanie oryginalnych charakterystyk oraz ich przybliżeń przy pomocy rozwinięcia w szereg Taylor'a [25]

4.4.2. Implementacja numeryczna

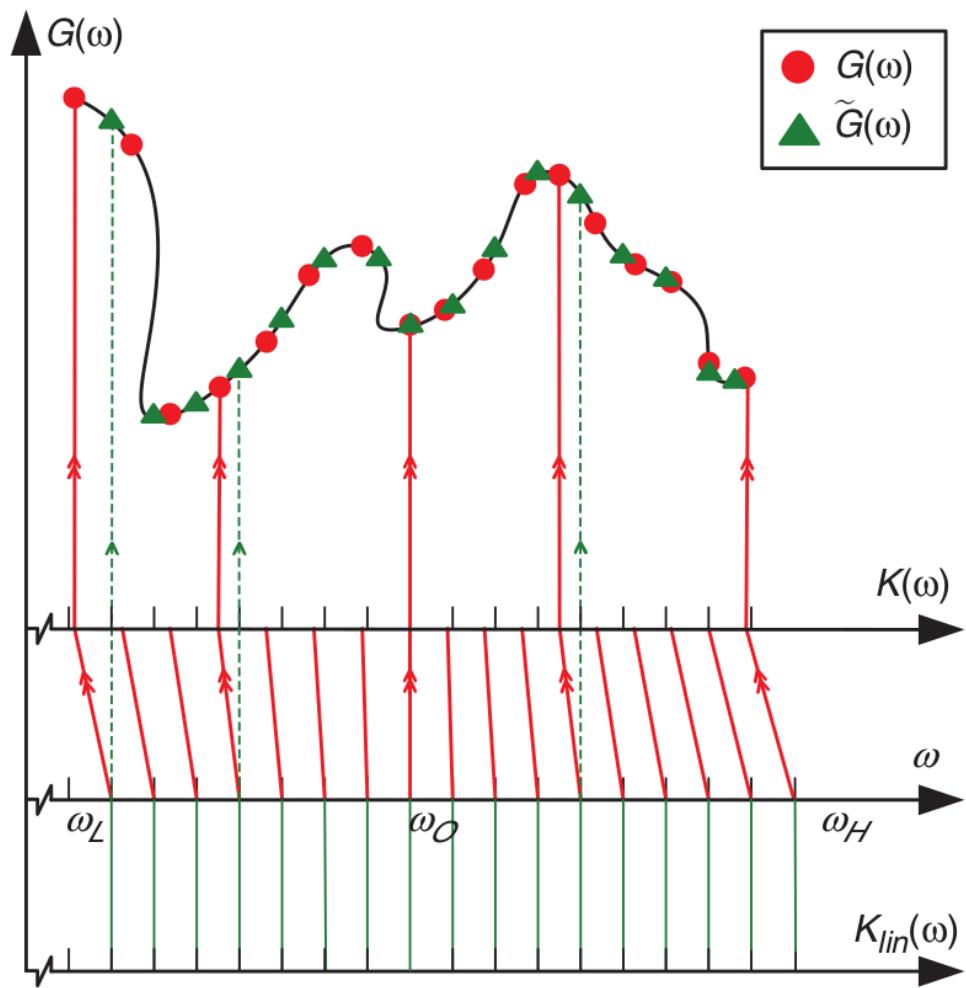
Implementacja prezentowanej metody opiera się głównie na znajomości krzywej dyspersji, której propagację bierzemy pod uwagę. Pierwszym krokiem, jest wygenerowanie odpowiedniego sygnału testowego. Mając wygenerowany sygnał można przystąpić do właściwego procesu kompensacji. W pierwszej kolejności analizowane jest widmo amplitudowe otrzymanego sygnału. Na jego podstawie uzyskiwana jest informacja o częstotliwości z największą energią. Zostaje ona wybrana na częstotliwość w której nastąpi przybliżenie liniowe. Po wybraniu ω_0 odnajdywane jest na krzywej dyspersji odpowiednia wartość liczby falowej. Korzystając z zależności opisujących wartości k_0 i k_1 wyliczone zostaje liniowe przybliżenie badanej krzywej. Uzyskane w aplikacji wyniki przedstawia rysunek 4.18. Linia niebieska prezentuje oryginalną krzywą dyspersji, natomiast linia zielona przedstawia jej przybliżenie uzyskane w aplikacji przy użyciu rozwinięcia w szereg Taylor'a.

Kolejnym krokiem implementowanego algorytmu, jest wyliczenie $G(k)$ na podstawie otrzymanego widma sygnału $G(\omega)$ oraz krzywej dyspersji. Następnie ponowne wyznaczenie zależności w dziedzinie częstotliwości, tym razem jednak używając przybliżenia liniowego zamiast oryginalnej zależności $k(\omega)$. Omawiany algorytm doskonale ilustruje rysunek 4.19

Przedstawia on krzywą $G(\omega)$ z aż trzema osiami poziomymi. Linia ciągła przedstawia zależność $G(K(\omega))$, czerwonymi kółkami oznaczony jest sygnał $G(\omega)$. Jak widać przejście z dziedziny $K(\omega)$ na ω jest nieliniowe co wynika z nieliniowego charakteru krzywej dyspersji. Zielonymi trójkątami oznaczono natomiast krzywą $G(K_{lin}(\omega))$. Przejście z dziedziny ω na $K_{lin}(\omega)$ jest przejściem o charakterze liniowym.



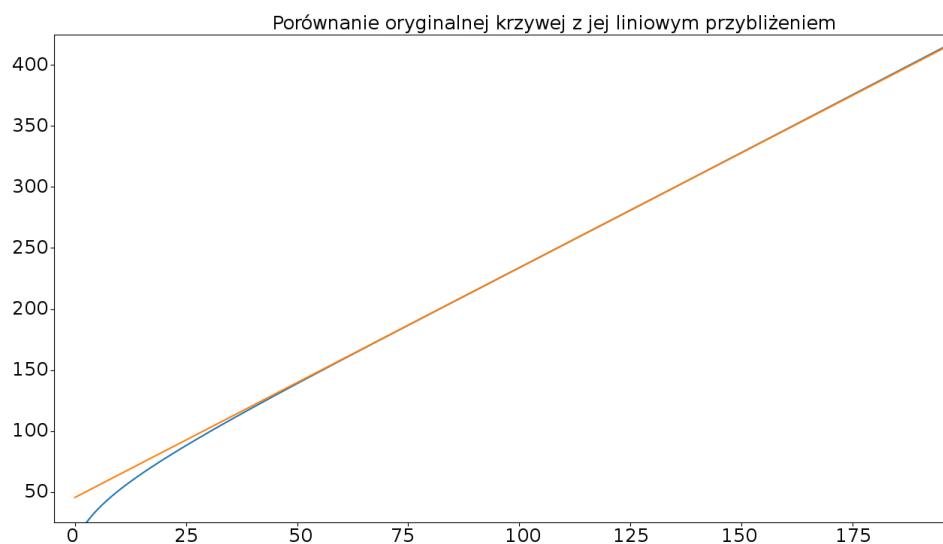
Rys. 4.18. Przykładowe porównanie oryginalnej krzywej oraz jej przybliżeń przy pomocy rozwinięcia w szereg Taylor'a



Rys. 4.19. Wizualizacja algorytmu [25]

4.4.3. Wybrane wyniki symulacji

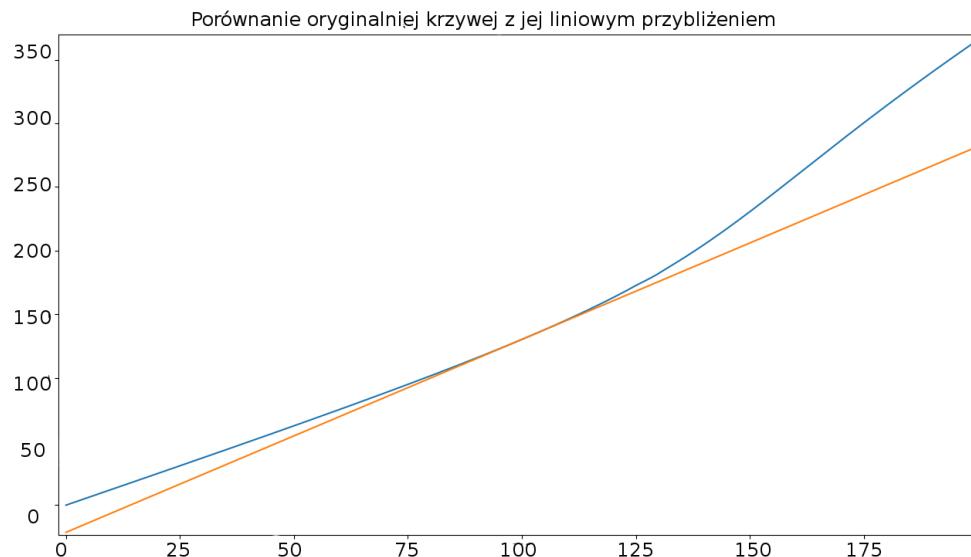
Dokładność odwzorowania przybliżenia liniowego oryginalnej krzywej zależy od jej kształtu. Im bardziej liniowa jest charakterystyka w pewnym otoczeniu wybranej częstotliwości, tym mniej widoczne będzie zjawisko dyspersji i tym dokładniejsze będzie odwzorowanie. Rysunek 4.20 pokazuje wygenerowane przybliżenie krzywej dyspersji odpowiadającej pierwszej postaci oraz jej liniowego przybliżenia. Rysunek 4.21 krzywą odpowiadającą trzeciej postaci oraz jej liniowego przybliżenia. Łatwo można zauważyć, że w przypadku pierwszej krzywej największe różnice są przy bardzo niskich częstotliwościach a dalej funkcja uzyskuje kształt niemalże liniowy. W przypadku drugiej krzywej dopasowanie jest zdecydowanie mniej dokładne



Rys. 4.20. Porównanie oryginalnej krzywej z jej liniowym przybliżeniem

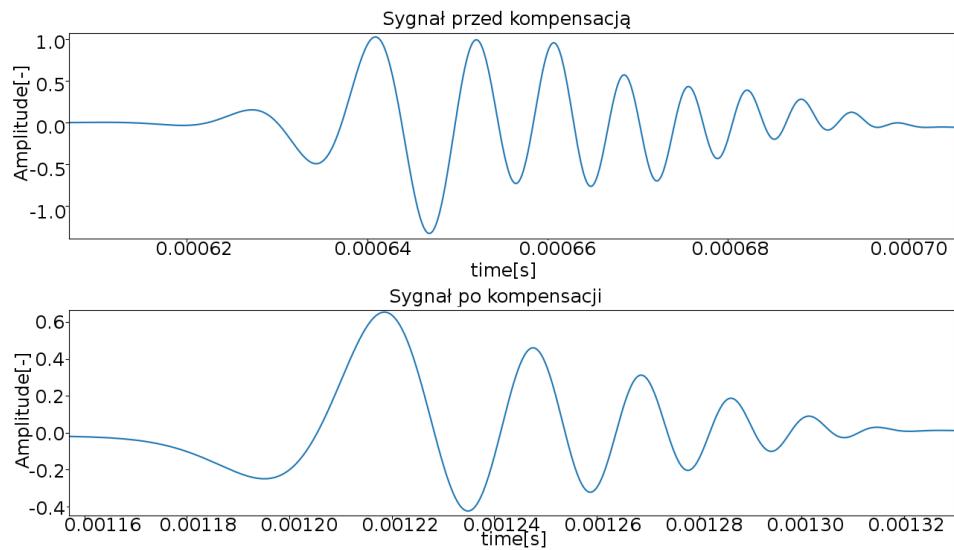
Kolejny rysunek przedstawia przykład sygnału przed i po kompensacji. W prezentowanym przykładzie propagowała pierwsza postać drgań, na odległość dwóch metrów. Sygnał skompensowany w znacznie większym stopniu przypomina sygnał wejściowy. Również czas jego trwania nieco się skrócił. Ze względu na małą dyspersyjność wybranego trybu, wydłużenie w czasie w prezentowanym przykładzie nie było bardzo znaczne, niemniej jednak stosując omawianą metodą kompensacji czas trwania sygnału udało się skrócić do wartości zbliżonej do długości sygnału oryginalnego.

Na ostatnim rysunku 4.23 przedstawione zostało porównanie otrzymanych w wyniku kompensacji sygnałów z sygnałem zadanym. Sygnał nie został odzyskany do postaci idealnie odpowiadającej pierwotnemu sygnałowi, co wynika bezpośrednio z zastosowanej procedury, jednak cel kompensacji, jakim jest skrócenie czasu trwania otrzymanego sygnału został osiągnięty. Dodatkowym atutem prezentowanej metody w stosunku do tej omówionej w poprzednim podrzodziale jest brak konieczności znajomości długości

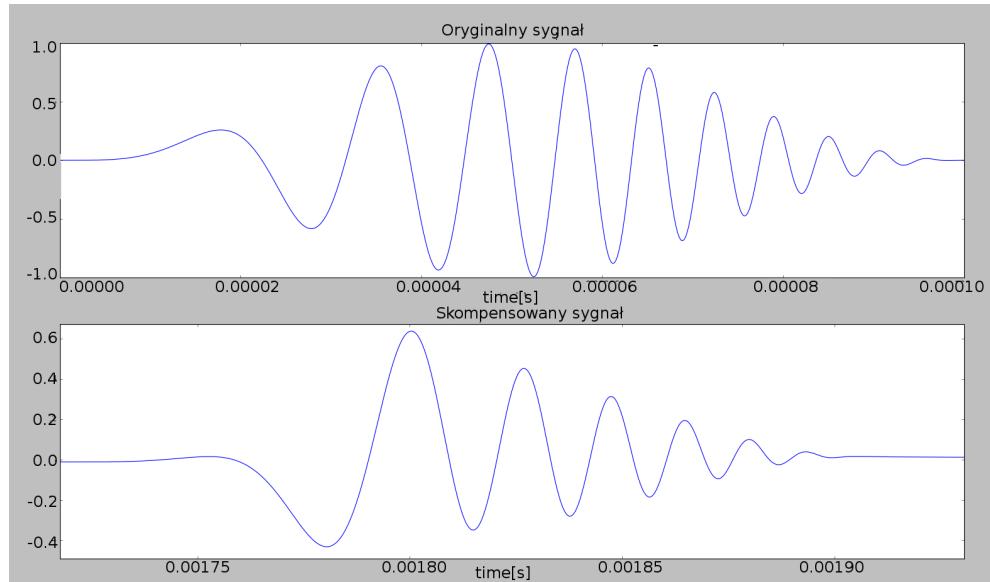


Rys. 4.21. Porównanie oryginalnej krzywej z jej liniowym przybliżeniem

ścieżki propagacji. Metoda pozwala zatem na badanie obiektu bez założeń o odległości na jaką sygnał będzie propagował oraz ilości jego odbić w obiekcie.



Rys. 4.22. Porównanie sygnału przed i po kompensacji



Rys. 4.23. Porównanie sygnału oryginalnego z sygnałem skompensowanym

4.5. Metoda mapowania sygnału z dziedziny czasu na dziedzinę odległości

Opisana w poniższej sekcji metoda została zaprezentowana w artykule [21]. Opiera się na odpowiednim zastosowaniu transformaty Fouriera. Pozwala na kompensację sygnału poprzez przejście z dziedziny czasu na dziedzinę odległości, dzięki czemu w wyniku otrzymywany jest skompensowany sygnał oraz informacja o długości ścieżki propagacji.

4.5.1. Podstawy teoretyczne

W niniejszej pracy rozważaną strukturą, w której powstają i propagują fale prowadzone jest długi stalowy pret. Przyjmując oznaczenie sygnału wejściowego jako $f(t)$, natomiast propagujący w przecie sygnał jako $u(x, t)$ zakłada się iż w miejscu wzbudzenia, zachodzi zależność:

$$u(x, t) = f(t) \quad (4.22)$$

Znając $u(x, t)$ w jednym punkcie w przestrzeni oraz znając charakterystykę propagującego trybu lub trybów fali prowadzonej, można odtworzyć sygnał po dowolnej odległości propagacji oraz w dowolnym punkcie czasowym. Można to osiągnąć rozważając przesunięcie fazowe każdej składowej częstotliwości oddzielnie:

$$u(x, t) = \int_{-\infty}^{\infty} F(\omega) e^{i(k(\omega)x - \omega t)} d\omega \quad (4.23)$$

Propagacja więcej niż jednego trybu fali oznacza, iż dana częstotliwość wzbudziła więcej niż jedną jej postać. Co za tym idzie w takim przypadku również możliwe jest odtworzenie przebiegu fali poprzez lekką modyfikację wzoru 4.23

$$u(x, t) = \int_{-\infty}^{\infty} F(\omega) e^{-i\omega t} \sum_{j=1}^n e^{ik_j(\omega)x} d\omega \quad (4.24)$$

gdzie:

$F(\omega)$ - transformata Fouriera sygnału $f(t)$

$k_j(\omega)$ - wartość j-tej liczby falowej odpowiadającej częstotliwości ω

Jak już zostało wspomniane, otrzymany sygnał $g(t)$ można odwzorować na funkcję odległości bez stosowania algorytmu kompensacji jedynie poprzez proste odwzorowanie:

$$x = v_{gr}t \quad (4.25)$$

Gdzie v_{gr} jest prędkością grupową trybu fali prowadzonej, mierzoną zazwyczaj dla częstotliwości środkowej, lub tej o największej energii. Odwzorowanie takie jest proste do osiągnięcia poprzez zwykłe skalowanie osi czasu $g(t)$ na odległość propagacji. W przypadku takiego odwzorowania, możliwości wykrycia defektów struktury, znajdujących się w bliskiej odległości od innych cech strukturalnych takich jak na przykład koniec badanego pręta czy punkt łączenia dwóch prętów, zależy od czasu trwania poszczególnych sygnałów. Długie sygnały będą się na siebie nakładać co może doprowadzić do niewykrycia uszkodzenia. Z tego względu sygnał $g(t)$ powinien być jak najkrótszy.

Przedstawiany algorytm kompensacji dyspersji, zastępuje proste mapowaczenie czasu na odległość przy pomocy równania 4.25 takim, które jest wykonywane w domenie liczby falowej. Takie podejście pozwala wziąć pod uwagę prędkości zależne od częstotliwości, czyli wzięcie pod uwagę dyspersji. Niezbędnym elementem omawianej metody jest znajomość krzywych dyspresji badanego obiektu, w zakresie propagowanych trybów. Dokładność z jaką wyliczone krzywe reprezentują rzeczywistą charakterystykę układu, znajduje odzwierciedlenie w jakości uzyskanych wyników. W sytuacji idelanej przedstawiany algorytm przywraca każdy sygnał do dokładnego kształtu pierwotnego, niezależnie od jego odległości propagacji.

Celem omawianej metody jest przekształcenie otrzymanego sygnału $g(t)$ w funkcję odległości propagacji a nie czasu, oraz skompensowanie rozproszenia otrzymanego sygnału. Dalsze propagowanie otrzymanego sygnału zarówno w przód jak i w tył można otrzymać stosując wzór 4.23 lub 4.24. Gdyby obliczyć kompletną mapę odległościowo czasową, sygnał propagujący dalej rozpraszałby się coraz mocniej zarówno w czasie jak i przestrzeni. Natomiast sygnał propagujący wstecz zbiegłby się do swojego minimum w pewnym punkcie, a następnie ponownie rozproszył. Minimum jego trwania przypadłoby na chwilę czasową $t = 0$. Dobrze ilustruje to rysunek 4.24. Obliczenie propagowanego wstecz sygnału dla ujemnych wartości x' w $t = 0$ daje dokładnie poszukiwane odwzorowanie z czasu na odległość propagacji, która jest poszukiwana i która kompensuje rozproszenie odebranych sygnałów. Nowa zmienna propagacji x' może być zdefiniowana jako $x = -x'$.

Wynika z tego, iż propagacja wsteczna może zostać opisana wzorem:

$$h(x) = u(-x', 0) = \int_{-\infty}^{\infty} G(\omega) e^{-ik(\omega)x} d\omega \quad (4.26)$$

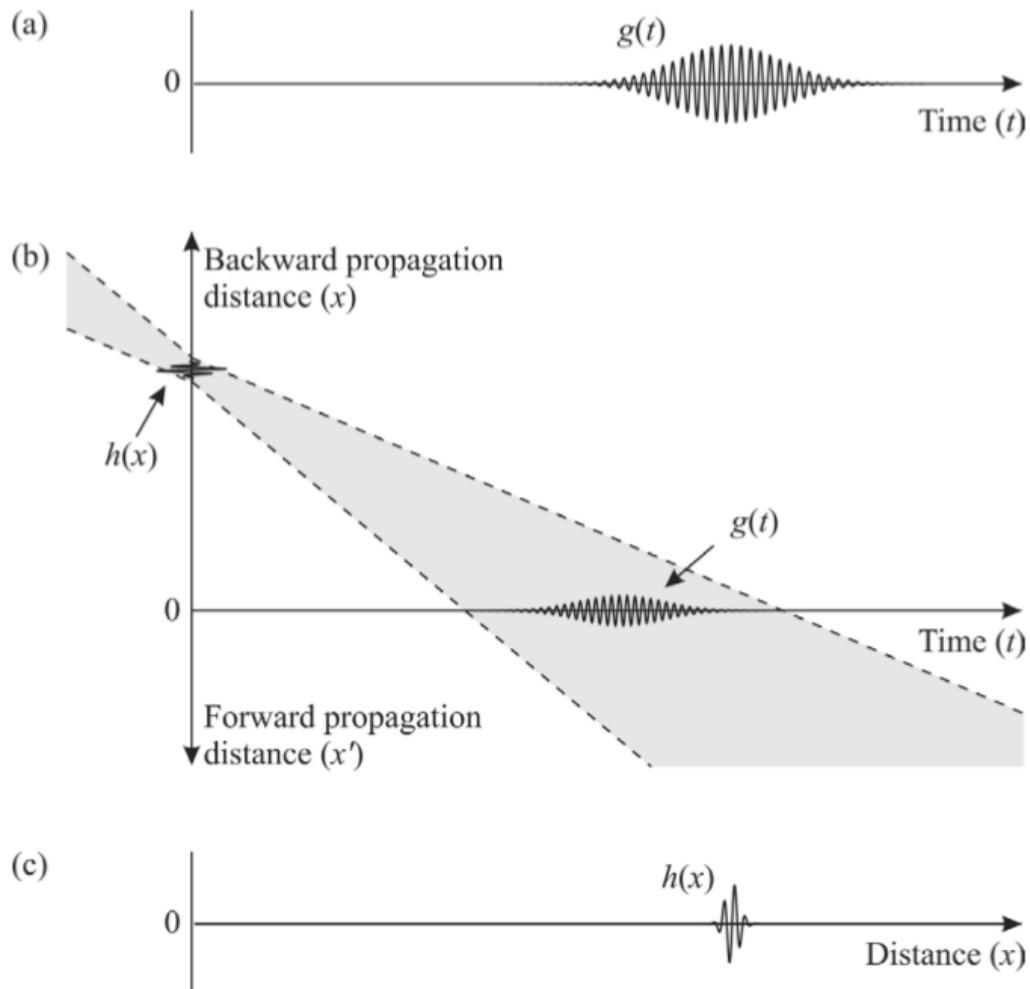
gdzie:

$G(\omega)$ - transformata Fouriera odebranego sygnału w dziedzinie czasu $g(t)$

$h(x)$ - skompensowany dyspersyjnie przebieg odległości.

Funkja $h(x)$ przedstawiona jest na rysunku 4.24 (c).

W niniejszej pracy nie został uwzględniony współczynnik odbicia A_j . Na potrzeby symulacji przyjęte zostało iż jest on stały, niezależny od częstotliwości i wynosi 1. Gdyby



Rys. 4.24. Zasada działania omawianej techniki kompensacji dyspersji [21]

jednak brać go pod uwagę to wzór 4.16 przyjałby postać:

$$g(t) = \sum_j \int_{-\infty}^{\infty} A_j(\omega) F(\omega) e^{ik(\omega)x_j - wt} d\omega \quad (4.27)$$

W przypadku stałego współczynnika odbić wszystkie sygnały w $h(x)$ powinny mieć ten sam kształt obwiedni co oryginalny sygnał wejściowy. Gdyby jednak zależał on od częstotliwości, wówczas obwiednia sygnału w $h(x)$ zostanie zniekształcona i na ogólny wydłużona. Takie zniekształcenie jest artefaktem charakterystycznym dla danego punktu odbicia fali. Co za tym idzie takie zakłócenie byłoby obserwowane nawet, gdyby zjawisko dyspersji nie występowało.

4.5.2. implementacja numeryczna

Równanie 4.26 jest podstawowym równaniem omawianej metody. Jest równaniem kompensacji dyspersji dla funkcji ciągły. W praktyce sygnał $g(t)$ nie jest sygnałem ciągłym a sygnałem dyskretnym, wygenerowanym w aplikacji lub otrzymanym z przetwornika. Na potrzeby tego rozdziału przyjmijmy następujące oznaczenia:

m - liczna próbek sygnału $g(t)$

Δt - okres próbkowania sygnału $g(t)$

$m\Delta t$ - całkowity czas trwania sygnału $g(t)$

n - liczba próbek otrzymanego sygnału w dziedzinie odległości $h(x)$

Δx - przestrzenny okres próbkowania $h(x)$

$n\Delta x$ - całkowita przestrzenna długość sygnału

Najprostszą implementacją omawianego algorytmu jest poddanie sygnału $g(t)$ szybkiej transformacie Fouriera, a następnie całkowanie numeryczne w zakresie częstotliwości sygnału wejściowego. Takie rozwiązanie byłoby niezwykle czasochłonne ponieważ całkowanie musiałoby zostać wykonane dla każdego z n punktów w skompensowanym sygnale $h(x)$. Metodą, która została zaimplementowana w opisanej aplikacji polega na modyfikacji równania 4.26. Ta zmiana pozwala na wykonywanie całkowania w ramach algorytmu odwrotnej transformaty Fouriera. Skompensowany sygnał jest funkcją odległości. Odwrotna transformata Fouriera musi zostać zastosowana na sygnale w dziedzinie liczby falowej. Wynika z tego, iż zmienna całkowania w równaniu 4.26 musi zostać zmieniona z ω na k . Można to łatwo zrobić używając poniższych zależności:

$$d\omega = v_{gr}(\omega)dk \quad (4.28)$$

$$\omega = v_{ph}(\omega)k \quad (4.29)$$

Gdzie:

$v_{gr}(\omega)$ - prędkość grupowa propagującej postaci

$v_{ph}(\omega)$ - prędkość fazowa propagującej postaci

Biorąc to pod uwagę równanie 4.26 można zapisać jako:

$$h(x) = \int_{-\infty}^{infty} H(k) e^{-ikx} dk \quad (4.30)$$

$$H(k) = G(k)v_{gr}(k) \quad (4.31)$$

$$\omega = \omega(k)$$

Równanie 4.30 ma postać odwrotnej transformaty Fouriera $H(k)$, w punktach spróbkowanych ze stałym przestrzennym okresem próbkowania Δk . Uzyskana z transformaty Fouriera funkcja $G(\omega)$ jest funkcją o punktach równomiernie spróbkowanych w dziedzinie częstotliwości. Ze względu na nielinowy charakter krzywych dyspersji przejście z dziedziny częstotliwości na dziedzinę liczby falowej dałoby w rezultacie funkcję o nierównomiernie spróbkowanej osi k . Aby uzyskać porządkany efekt konieczne jest użycie znanej z krzywej dyspersji zależności między wartościami ω oraz k , aby interpolować funkcję G , tak by znaleźć jej wartości w punktach równomiernie rozmieszczonych w dziedzinie liczby falowej.

$G(\omega)$ jest sygnałem zawierającym informacje zarówno o fazie jak i amplitudzie sygnału. W rozwiązyaniu numerycznym istotne jest aby dobrać odpowiednie wartości kroków. Zbyt duże ich wartości mogą spowodować utratę informacji, natomiast zbyt małe prowadzą do znacznego wydłużania czasu obliczeń, w zamian za co uzyskiwane wyniki są dokładniejsze i wpływ szumu staje się mniej znaczący. W celu zmniejszenia szumu powodowanego przez błędy interpolacji w końcowym, skompensowanym sygnale, otrzymany sygnał $g(t)$ wypełnia się zerami. W aplikacji użytkownik ma możliwość samodzielnie podać ile razy chce wydłużyć sygnał poprzez wypełnienie zerami. Ograniczeniem podlega zarówno Δk jak i ilość punktów liczb falowych n . Aby zapobiec zawijaniu sygnału w domenie odległości, minimalna długość skompensowanego sygnału $h(x)$ musi być większa od długości pierwotnego sygnału $g(t)$ pomnożonego przez maksymalną prędkość grupową v_{max} .

$$n\Delta x > m\Delta t v_{max} \quad (4.32)$$

Powyższa nierówność określa nam maksymalny rozmiar kroku liczby falowej jako:

$$\Delta k = \frac{1}{n\Delta x} < \frac{1}{m\Delta t v_{max}} \quad (4.33)$$

Aby cała energia pierwotnego sygnału została poprawnie odwzorowana na dziedzinę odległości, liczba falowa Nyquista k_{Nyq} musi być większa lub równa liczbie falowej trybu fali prowadzonej o częstotliwości Nyquista f_{Nyq} oryginalnego sygnału. Co można zapisać jako:

$$\begin{aligned} k_{Nyq} &\geq k(f_{Nyq}) \\ &\geq k\left(\frac{1}{2\Delta t}\right) \end{aligned} \quad (4.34)$$

Nierówność 4.34 oraz równanie 4.33 pozwalają określić minimalną liczbę punktów w sygnale $h(x)$

$$n > 2 \frac{k_{Nyq}}{\Delta k} \quad (4.35)$$

W zaimplementowanym algorytmie, pierwszym krokiem jest wydłużenie otrzymanego sygnału zadaną ilość razy oraz wykonanie szybkiej transformaty Fouriera na

wydłużonym sygnale. Następnie na podstawie przedstawionych ograniczeń dobierana jest wartość kroku Δk oraz obliczana wartość $k_{max} = k(\omega_{max})$. Znając te parametry tworzony jest wektor zawierający równomiernie rozłożone wartości liczby falowej. Przy użyciu krzywej dyspersji, interpolując liniowo potrzebne wartości obliczana jest funkcja $G(k)$ oraz $v_{gr}(k)$ w tych samych punktach. Następnie wyliczana zostaje funkcja $H(k)$ mająca równomiernie rozłożone wartości liczby falowej. Ostatnim etapem jest zastosowanie szybkiej odwrotnej transformaty Fouriera co pozwala uzyskać poszukiwany sygnał $h(x)$.

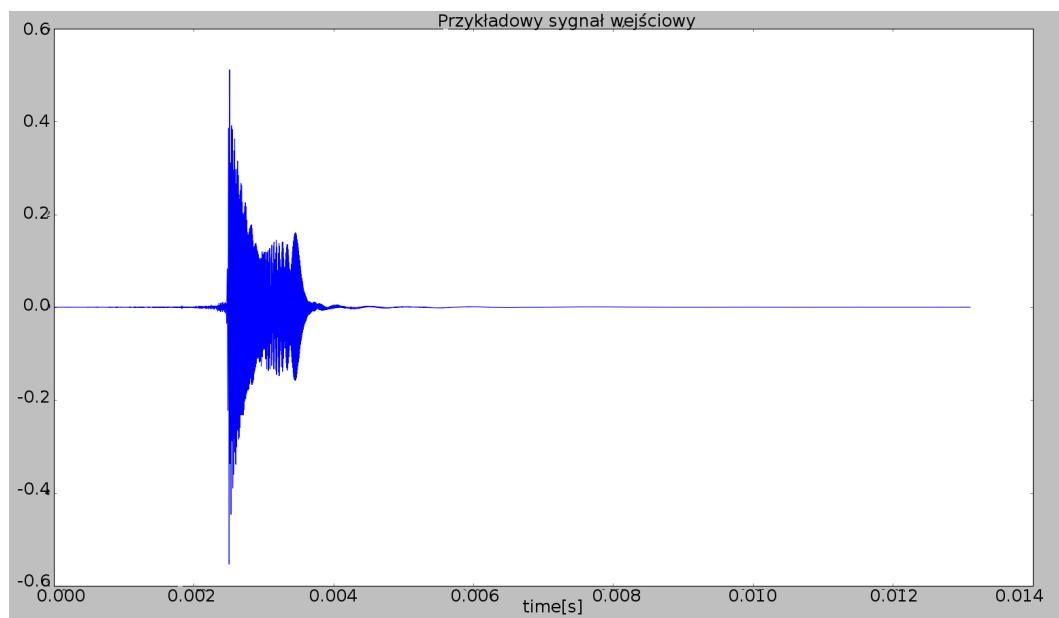
W przypadku gdy mamy do czynienia z propagacją większej ilości postaci fal lub gdy podczas odbicia następuje konwersja trybu fali prowadzonej dokonywane jest uśrednianie propagowanych postaci. Przy założeniu, że propagują jednocześnie dwie postacie równanie 4.26 można ponownie zapisać jako:

$$h(x) = \int_{-\infty}^{\infty} G(\omega) e^{i(k_1(\omega) + k_2(\omega))\frac{x}{2}} d\omega \quad (4.36)$$

W implementacji oznacza to, iż kilka krzywych dyspersji zostaje ośrednionych do jednej postaci w której $k_{12} = \frac{1}{2}(k_1(\omega) + k_2(\omega))$

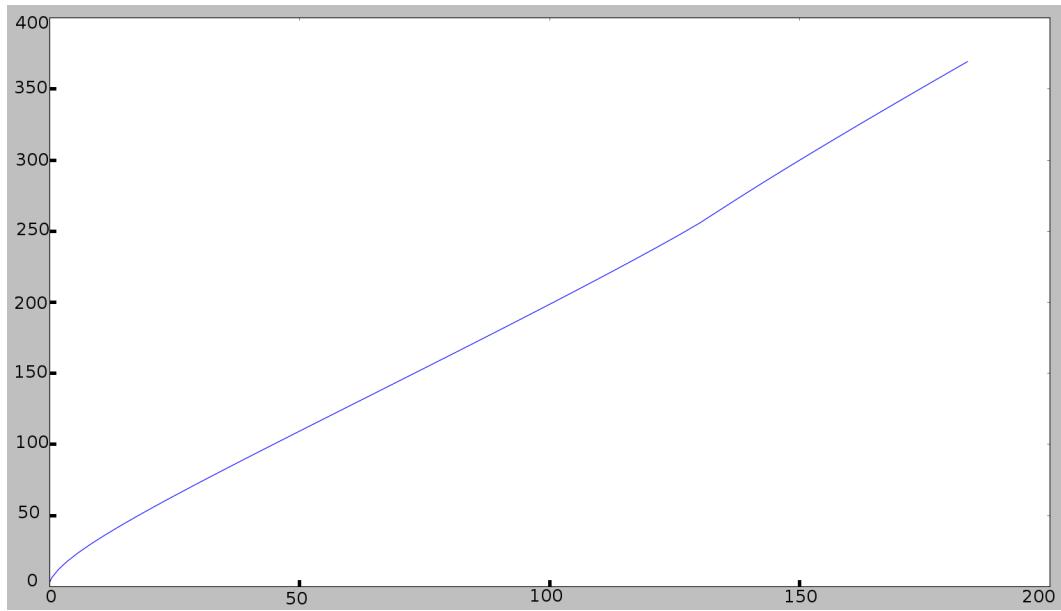
4.5.3. Wybrane wyniki symulacji

Przykładowe sygnał wejściowy wygenerowane w aplikacji zaprezentowane zostały na rysunku 4.25. Przykładowa krzywa dyspersji powstała w wyniku uśrednienia trzech



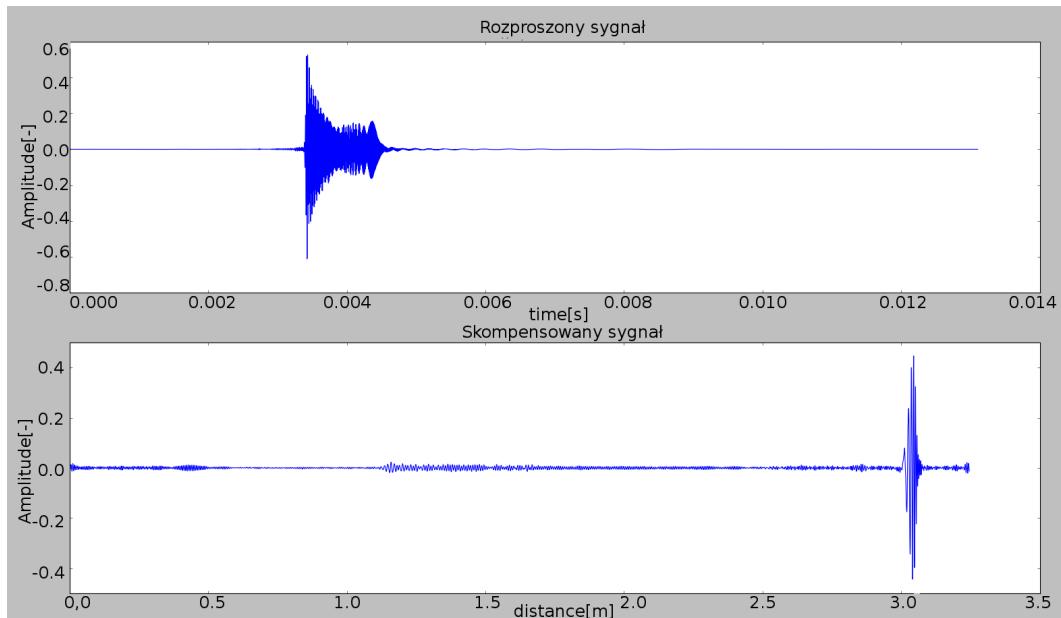
Rys. 4.25. przykładowy sygnał wejściowy

pierwszych postaci fal zaprezentowana została na rysunku 4.26. Na kolejnym rysunku



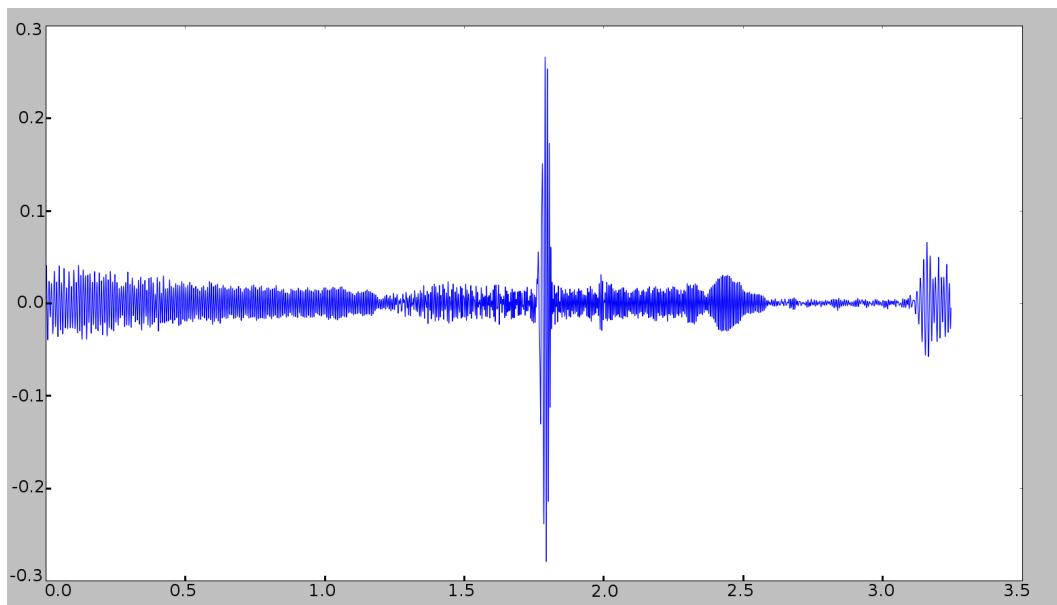
Rys. 4.26. Średnia krzywa dyspersji powstała z uśrednienia trzech pierwszych krzywych

4.27 przedstawiono porównanie sygnału przed i po kompensacji. Rozproszony sygnał został skompensowany do znacznie krótszej postaci. Dodatkowo w łatwy sposób możliwe jest odczytanie długości ścieżki propagacji, która w tym przypadku wynosiła dwa metry. Kolejny rysunek przedstawia wyniki symulacji, w której sygnał nie został dostatecznie



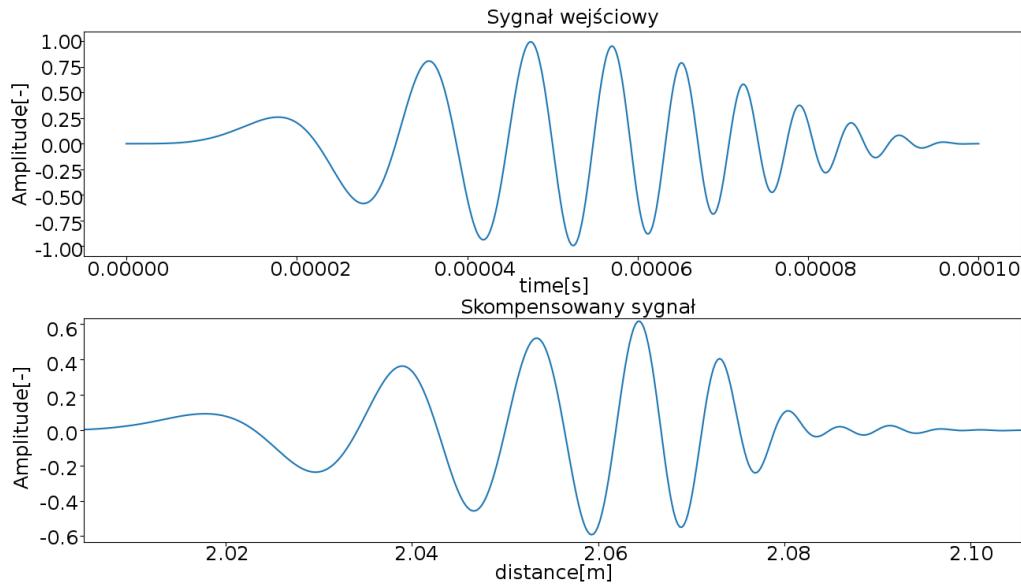
Rys. 4.27. Porównanie sygnału przed i po kompensacji

wydłużony w wyniku czego otrzymany wynik był nieczytelny 4.28 Ostatni rysunek prezentuje porównanie skompensowanego sygnału z sygnałem wejściowym, w przypadku, gdy symulowana była propagacja pierwszych trzech postaci fali prowadzonej. Omawiana metoda daje najlepsze wyniki kompensacji. Ze wszystkich prezentowanych w ramach



Rys. 4.28. Sygnał, który przeprowadzał 5 metrów, widoczne zawijanie sygnału

niniejszej pracy metod, ta najdokładniej odwzorowuje zadany na wejściu sygnał. Jej kolejnymi atutami są minimalne wymagania jeśli chodzi o dane wejściowe oraz największa ilość otrzymywanych informacji. Na podstawie znajomości jedynie przeprowadzonego sygnału oraz znajomości krzywych dyspersji propagujących postaci jest ona w stanie skompensować rozproszony sygnał, podając jednocześnie informację o długości ścieżki propagacji. Algorytm radzi sobie dobrze zarówno z kompensacją jednej propagującej postaci jak i dwiema a nawet trzema.



Rys. 4.29. Porównanie sygnału wejściowego oraz sygnału skompensowanego

4.5.4. Przykład zastosowania na podstawie symulacji wybranego pręta

Na potrzeby symulacji, jako testowany obiekt, przyjęty został pręt wykonany ze stali czarnej o grubości 25 mm długości dwóch metrów. Stałe materiałowe wyniosły odpowiednio:

współczynnik Poissona - $\nu = 0.3$

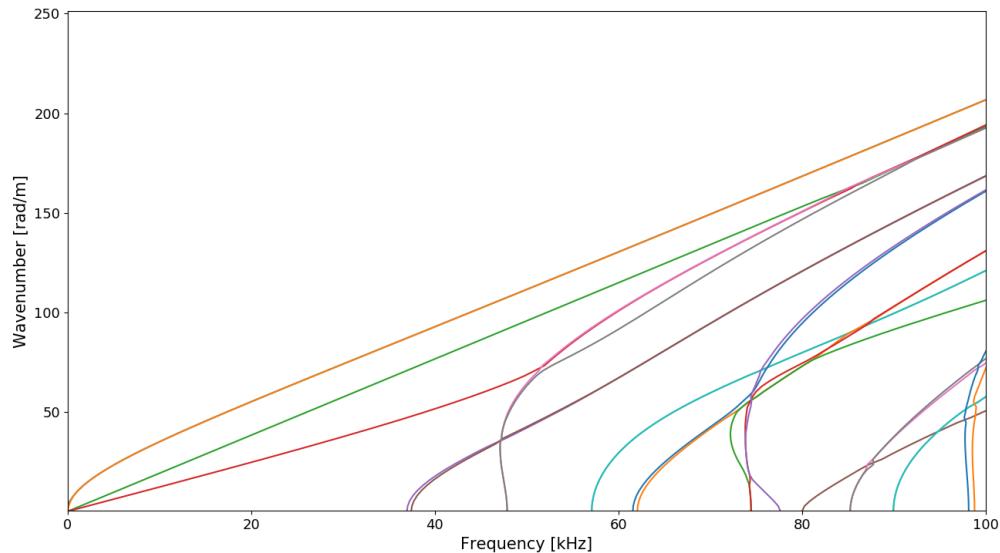
moduł Younga - $E = 210 \text{ GPa}$

Pierwszym etapem było wygenerowanie odpowiednich krzywych dyspersji. Zostały one przedstawione na rysunku 4.30

Do ich wygenerowania użyte zostały czworościenne elementy skończone. Płaszczyzna złożona z ośmiu okręgów, w pierwszym z nich wygenerowane osiem punktów. Rozkład punktów na pojedynczej płaszczyźnie przedstawia wynik 4.31

Do symulacji użyty został sygnał linear chirp, opisany na początku tego rozdziału. Pierwszym krokiem jest jego odpowiednie wydłużenie w czasie poprzez dołożenie na jego końcu odpowiedniej ilości zer. Następnie wykonana została symulacja propagacji tak przygotowanego sygnału na dystans dwóch metrów. Symulacja taka odpowiada sytuacji, w której odpowiedni wzbudnik został umieszczony na jednym z końców badanego obiektu, natomiast na jego drugim końcu znalazły się odpowiedni odbiornik. Do symulacji przyjęte zostało, iż w badanym obiekcie propagują trzy pierwsze postaci fali prowadzonej. Wyniki symulacji przedstawione zostały na rysunku 4.32

Kolejnym krokiem jest kompensacja tak otrzymanego sygnału. Otrzymany drogą symulacji, rozproszony sygnał został przekazany do funkcji kompensującej. W wyniku

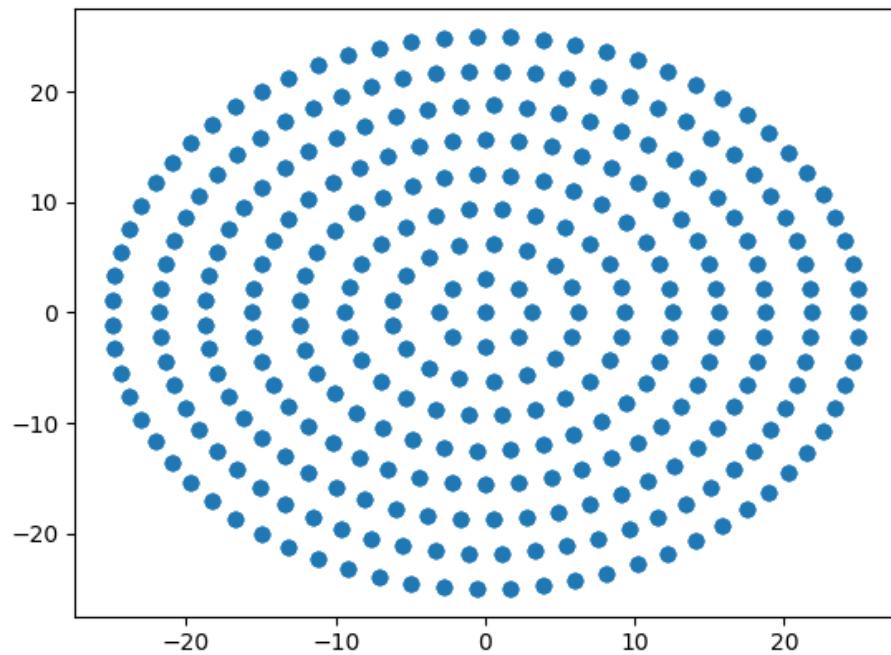


Rys. 4.30. Krzywe dyspersji analizowanego pręta

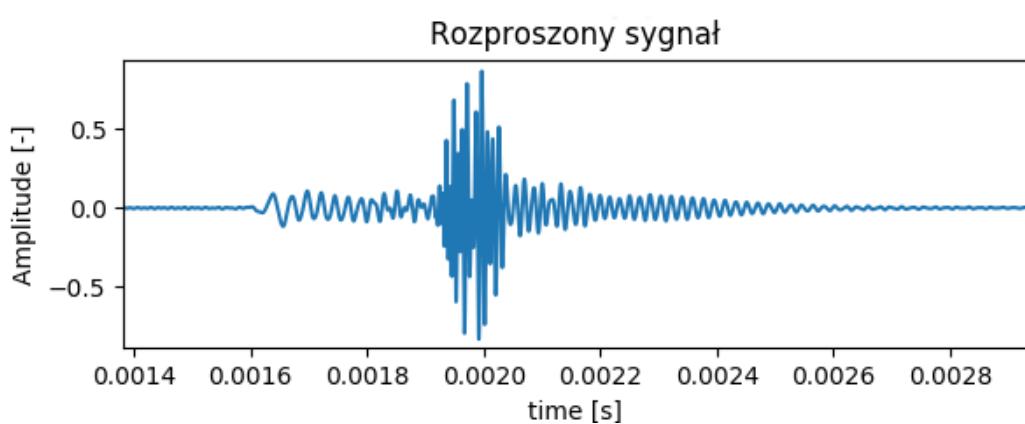
kompensacji otrzymany zostaje sygnał zmapowany na dziedzinę odległości. Z otrzymanych wyników łatwo odczytać, iż długość ścieżki propagacji wyniosła dwa metry. Jest to dokładnie taka odległość na jaką sygnał wejściowy został przepropagowany zaraz na początku. Uzyskany omawianą metodą sygnał przedstawiony został na rysunku 4.33.

Ostatnie rysunek przedstawia zestawienie pozwalające na porównanie: sygnału wejściowego, sygnału rozproszonego oraz sygnału skompensowanego. Łatwo zauważać, iż sygnał przed kompensacją uległ dyspersji i jest znacznie rozproszony, co uniemożliwiłoby jego identyfikację oraz określenie, czy badana struktura nie jest uszkodzona. Sygnał po kompensacji, jest niemal identyczny z sygnałem zadanym.

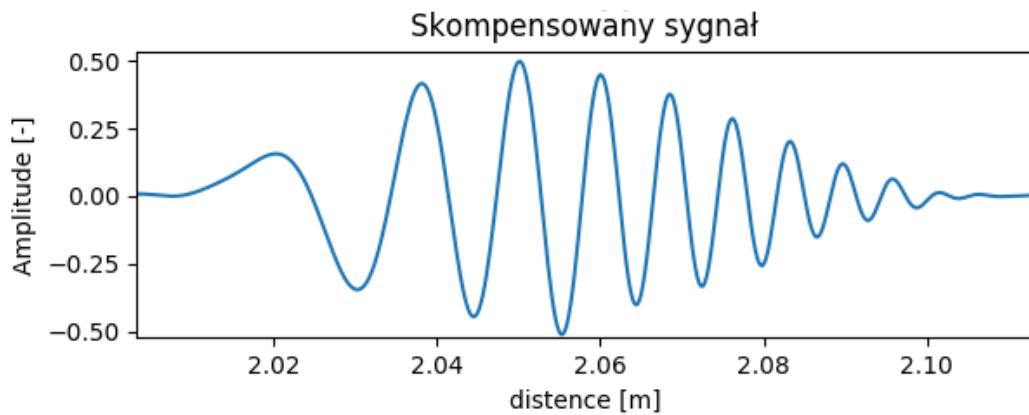
Stworzona aplikacja pozwala zatem na kompleksową symulację fali prowadzonej w zadanym pręcie. Znając parametry fizyczne zadanego materiału, możliwe jest wygenerowanie modelu pręta wykonanego z dowolnego materiału, oraz dowolnej długości. Możliwa jest również symulacja propagacji fali prowadzonej na dowolną odległość oraz, co zostało opisane w tym rozdziale, kompensacja dyspersji trzema wybranymi metodami. Jak widać na powyższym przykładzie omawiana metoda pozwala na uzyskanie bardzo dobrych wyników symulacyjnych.



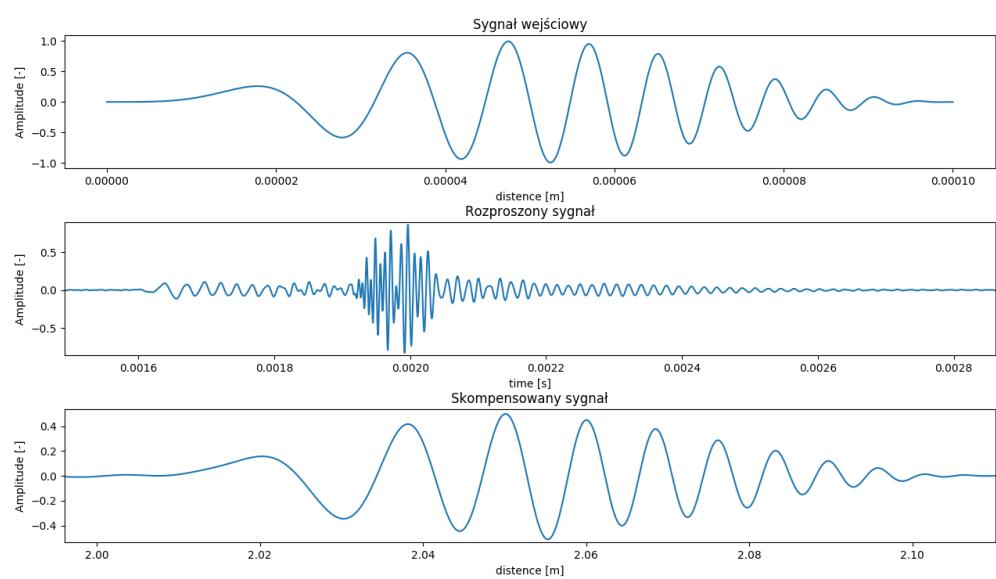
Rys. 4.31. rozkład punktów modelu MES na płaszczyźnie, użyty do generacji modelu



Rys. 4.32. Sygnał rozproszony, uzyskany drogą symulacji



Rys. 4.33. Sygnał skompensowany omawiana metodą



Rys. 4.34. Porównanie sygnałów: wejściowego, rozproszonego oraz skompensowanego

4.6. Porównanie opracowanych metod kompensacji

W tym rozdziale opisane zostały wybrane trzy metody kompensacji dyspersji, które zostały zaimplementowane do stworzonej aplikacji w ramach niniejszej pracy. Każda z przedstawionych metod ma zarówno wady jak i zalety i ograniczenia. W poniższym podrozdziale znajduje się porównanie opracowanych metod.

4.6.1. Metoda odwracania sygnału w czasie

Zalety:

1. Szybkość działania - ze wszystkich zaimplementowanych metod ta pozwala w najkrótszym czasie uzyskać rzadane rezultaty.
2. Skuteczność kompensacji - sygnał otrzymany w wyniku symulacji kompensuje się do postaci sygnału wejściowego z dokładnością do kolejności w czasie
3. Możliwość kompensacji sygnału wielopostaciowego - Metoda ta ze względu na swoją prostotę pozwala na skompensowanie zarówno sygnału będącego rozproszonym pojedynczym trybem fali prowadzonej jak i sygnału będącego złożeniem kilku rozproszonych postaci tej fali

Wady i ograniczenia:

1. Znana długość ścieżki propagacji - aby wygenerować żądaną sygnał musi być znana długość ściażki propagacji, po przebyciu której sygnał powinien się skompensować.
2. Otrzymany sygnał odwrócony w czasie - otrzymany po propagacji sygnał nie jest identyczny z żądanym sygnałem. Jak pokazano w symulacji otrzymany sygnał jest żądanym sygnałem odwróconym w czasie
3. Trudności w zastosowaniu w przypadku więcej niż jednego punktu odbicia - w przypadku odbić od kilku powierzchni, jak na przykład w przypadku badanie dwóch połączonych ze sobą prętów, stworzenie sygnału, który wprowadzony do obiektu powróci w skompensowanej formie jest trudniejsze i wymagałoby wielu prób heurystycznych.
4. Konieczność znajomości krzywych dyspersji - Aby móc wygenerować odpowiedni sygnał, który skompensuje się na zadanej odległości, niezbędna jest znajomość krzywych dyspersji do wstępnej symulacji sygnału.

Podsumowując, największą zaletą tej metody jest jej szybkość działania. Pomimo ograniczenia w postaci konieczności posiadania informacji o długości ścieżki propagacji, metoda ta wydaje się być skuteczna do kontroli zbadanych wcześniej obiektów. W przypadku badań heurystycznych wydaje się iż jej dodatkowym atutem może być brak konieczności znajomości krzywych dyspersji. Być może wystarczyłoby wcześniej zbadać obiekt, czyli

wprowadzić do obiektu sygnał, który ostatecznie chcemy otrzymywać po propagacji. Otrzymany eksperymentalnie sygnał odwrócić w czasie i znów wprowadzić do pręta. Tak przygotowany sygnał również powinien skompensować się podobnie jak w symulacji, jednak z pewnością już po pierwszej propagacji, w sygnale pojawią się szумy, które ostatecznie mogą pogarszać otrzymywane wyniki. Metoda wydaje się być dobra do kontroli dobrze, zbadanych obiektów, co do których należy się tylko upewnić, że nie powstały żadne uszkodzenia od czasu ostatniej kontroli. W takim wypadku po zadaniu sygnału oczekiwany jest konkretny rezultat, w przypadku pojawiienia się uszkodzeń, które spowodują dodatkowe i przedwczesne odbicie sygnału, otrzymany sygnał świadczyć będzie o uszkodzeniu. Nie dostarczy nam jednak informacji o miejscu jego wystąpienia.

4.6.2. Metoda mapowania liniowego przy pomocy rozwinięcia w szereg Taylora

Zalety:

1. Kompensacja dowolnej ilości odbić zadanego sygnału - Przedstawiona metoda pozwala na kompensację dowolnej ilości odbić sygnału wejściowego. Jeśli tylko punkty odbić nie będą zbyt blisko siebie, sygnały po kompensacji da się rozróżnić i określić liczbę punktów odbicia
2. Możliwość oszacowania długości ścieżki propagacji - znając prędkość grupową częstotliwości o najwyższej energii oraz czas przybycia sygnału możliwe jest oszacowanie długość ścieżki jego propagacji.

Wady i ograniczenia:

1. Czas obliczeń - ze wszystkich zaimplementowanych metod czas obliczeń tej jest najdłuższy.
2. Niedokładne odwzorowanie sygnału wejściowego - Wyniki z symulacji pokazują, iż ta metoda pomimo, że kompensuje sygnał do postaci zbliżonej do tej wejściowej to jednak jego obwiednia wydaje się być nieco zniekształcona w stosunku do oryginału
3. Możliwość kompensacji tylko pojedyńczej postaci fali - przedstawiona metoda pozwala na skompensowanie dyspersji tylko w przypadku, gdy mamy do czynienia z pojedyńczym trybem fali.
4. Znana krzywa dyspersji propagującego trybu fali - jest to informacja niezbędna do skompensowania sygnału.

Podsumowując, prezentowana metoda pozwala na skompensowanie sygnału odbitego od dowolnej ilości punktów. Jednak jej zasadniczym ograniczeniem jest propagacja fali tylko w jednej postaci, co jednak jest możliwe do osiągnięcia dzięki odpowiednio dobranemu wzbudnikowi oraz wyliczeniu postaci, która ma zostać wzbudzona oraz tych które powinny być stłumione. Przy odpowiednio spreparowanym sygnale wejściowym, zapewniającym propagację jedynie wybranej postaci fali, metoda ta powinna skutecznie skompensować

dyspersję oraz pozwolić na oszacowanie odległości na jakich znajdują się punkty odbicia, co za tym idzie zlokalizować potencjalne uszkodzenia.

4.6.3. Metoda mapowania sygnału z dziedziny czasu na dziedzinę odległości

Zalety:

1. Otrzymywana informacja o długości ścieżki propagacji - metoda ta przynosi podwójne korzyści, ponieważ oprócz skompensowanego sygnału dostarczana jest automatycznie również informacja o długości ścieżki propagacji sygnału. Otrzymany sygnał wyjściowy jest sygnałem w dziedzinie odległości, więc długość ścieżki propagacji można odczytać bezpośrednio z wygenerowanego wyniku
2. Kompensacja dowolnej ilości propagujących postaci oraz dowolnej liczby odbić sygnału - Jedynym elementem wymaganym aby skompensować otrzymany sygnał jest znajomość krzywych dyspersji opisujących propagujące postaci. Metoda pozwala na skompensowanie sygnału zarówno jednomowego jak i takiego, który zawiera kilka postaci fali.
3. szybkość działania - mimo iż zaimplementowany algorytm działa nieco dłużej niż pierwsza z omawianych metod, ta również dokonuje obliczeń w dość zadawalającym czasie. Dodatkowym atutem jest to, że poprzez drobne zmiany w aplikacji jesteśmy w stanie sterować szybkością jej działania. Poprzez zmianę odpowiednich parametrów użytkownik jest w stanie zdecydować czy zależy mu na dokładności wyników czy na szybkości wykonywanego algorytmu.
4. Jakość wyników - wyniki uzyskane z symulacji pokazują, iż kompensacja tą metodą pozwala na dokładne odwzorowanie sygnału wejściowego. Im dokładniejsze są opisujące go krzywe dyspersji tym otrzymany w wyniku kompensacji sygnał lepiej odwzorowuje stan faktyczny.

Wady i ograniczenia:

1. znajomość krzywych dyspersji - tak jak i poprzednie metody tak i ta również wymaga znajomości funkcji opisujących badany obiekt. Potrzebna jest również informacja o tym, które postaci zostały wzbudzone i propagowały w badanym obiekcie.

Ta metoda jest najskuteczniejszą z opracowanych w ramach tej pracy metod. Posiada najmniej ograniczeń jednocześnie pozwalając uzyskać najdokładniejsze rezultaty. Długość ścieżki propagacji nie tylko nie musi być znana jak również jest bezpośrednio przekazywana użytkownikowi wraz ze skompensowanym sygnałem.

5. Aplikacja do obliczeń numerycznych

KATARZYNA RUGIEŁŁO
BARTŁOMIEJ PIWOWARCZYK

W tym rozdziale przedstawione są wszystkie niezbędne informacje potrzebne do korzystania ze zbudowanej na potrzeby projektu aplikacji. Aplikacja napisana jest w skryptowym języku Python. IDE wykorzystanym do tworzenia projektu jest PyCharm. Python jak i wersja Community programu PyCharm są całkowicie darmowe. W dalszej części rozdziału przedstawiony jest sposób instalacji i konfiguracji środowiska dla systemu Windows 8 64-bit.

Aplikacja składa się z dwóch głównych części. Pierwsza obejmuje obliczanie modelu MES oraz wyznaczanie krzywych dyspersji i wzbudzalności na podstawie obliczonego modelu, a druga pozwala symulować propagację fali oraz metody kompensacji dyspersji. Każdy plik z rozszerzeniem .py zawierający kod w Pythonie nazywa się modułem. Każda ze wspomnianych części aplikacji składa się z kilku modułów. Oprócz tego jest kilka modułów z funkcjami pomocniczymi wykorzystywanymi w głównych częściach programu. Wszystkie moduły zostaną omówione w dalszej części rozdziału.

5.1. Instalacja i konfiguracja środowiska

BARTŁOMIEJ PIWOWARCZYK

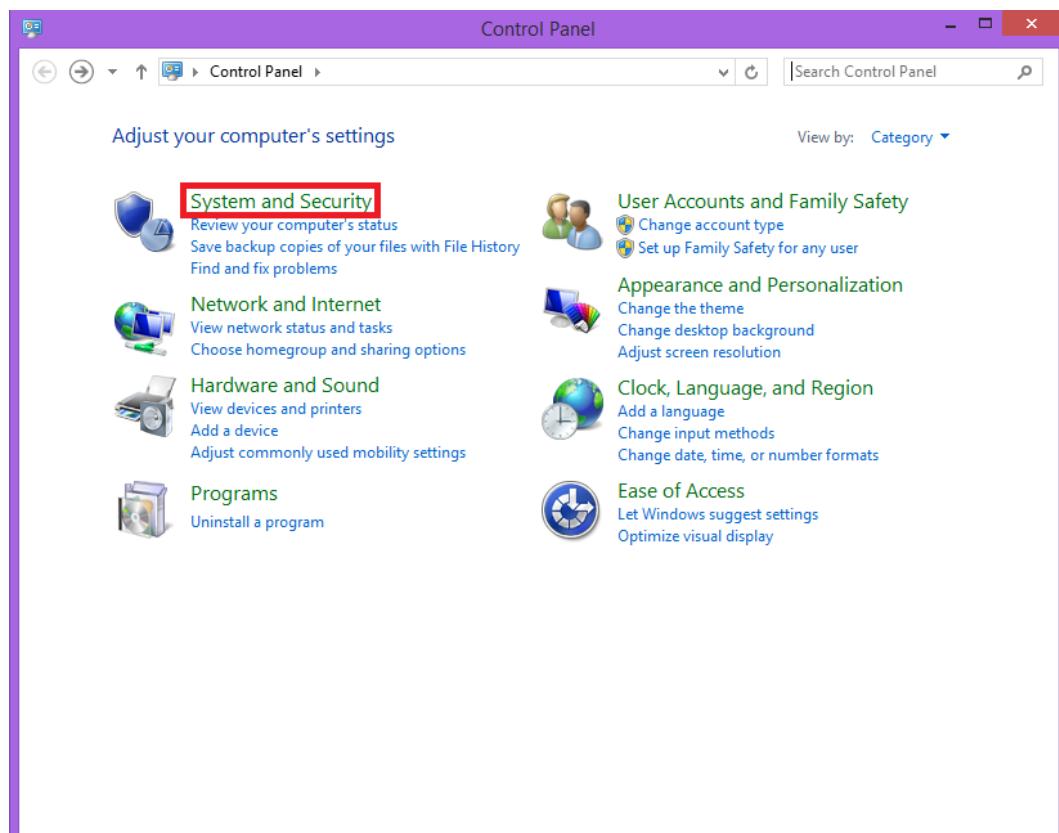
5.1.1. Python

Instalację rozpoczynamy od interpretera Python. Pliki instalacyjne pobrać można z oficjalnej strony Python "<https://www.python.org/>". Na stronie powinniśmy znaleźć pliki instalacyjne kolejnych wersji dla różnych systemów operacyjnych. Pobieramy plik odpowiedni dla naszego systemu. W omawianym przypadku będzie to win-64. Pliki instalacyjne zawarte są na płycie CD dołączonej do pracy. W projekcie korzystanu z wersji Python 3.6.3.

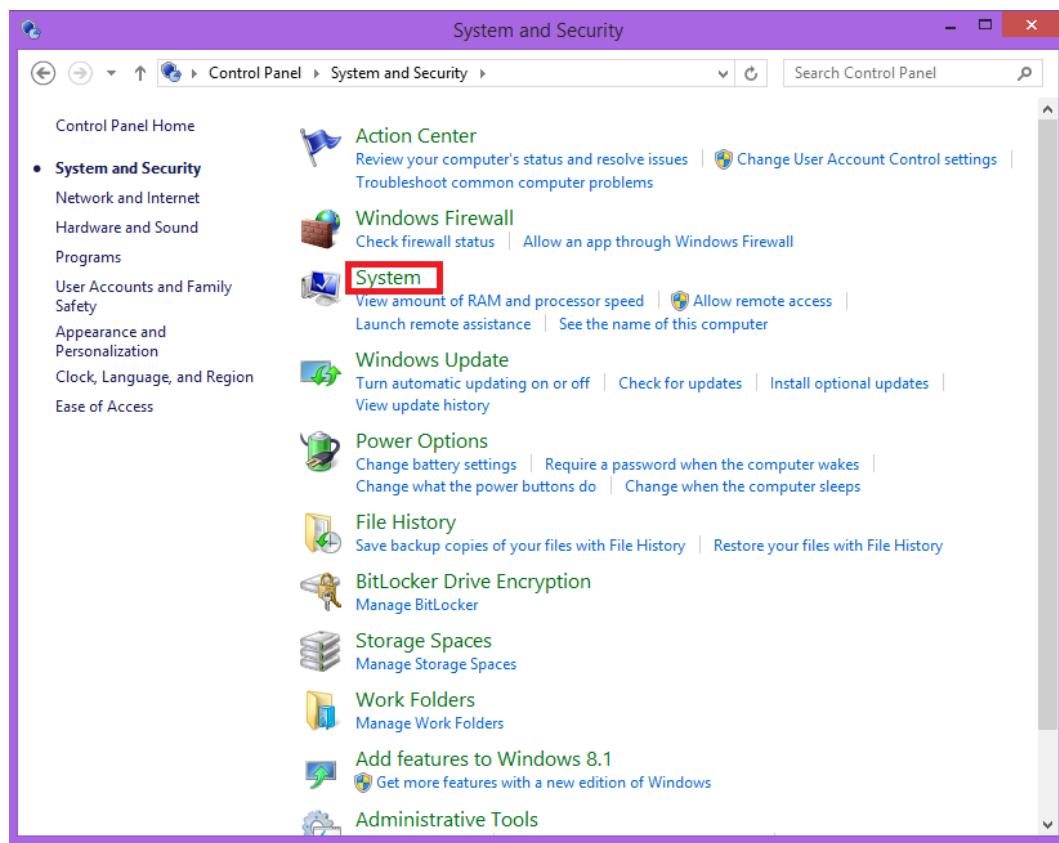
Po zainstalowaniu Pythona należy dodać ścieżkę do zmiennych systemowych. W tym celu wymagane są uprawnienia administratora. Poniżej znajdują się instrukcja dodawania ścieżki oraz sposób sprawdzenia czy Python jest widoczny przez system.

Otwieramy „Panel Sterowania” („Control Panel”). Wchodzimy w zakładkę „System i Zabezpieczenia” („System and Security”), rysunek 5.1, a następnie w „System”, rysunek 5.2. Z lewej strony klikamy „Zaawansowane ustawienia systemu” („Advanced system settings”), rysunek 5.3, a następnie „Zmienne środowiskowe” („Environment variables”), rysunek 5.4. Znajdujemy zmienną „Ścieżka” („Path”), rysunek 5.5 i otwieramy ją przez dwukrotne kliknięcie myszką. Przesuwamy do końca tekstu w nowo otwartym oknie i dopisujemy na końcu średnik, ścieżkę do folderu gdzie zainstalowaliśmy Pythona, kolejny średnik i ścieżkę do folderu Scripts, który znajduje się w katalogu instalacji, rysunek 5.6. Przykładowe ścieżki to ;F:\Programy\Python;F:\Programy\Python\Scripts. We wszystkich oknach które otworzyliśmy klikamy OK.

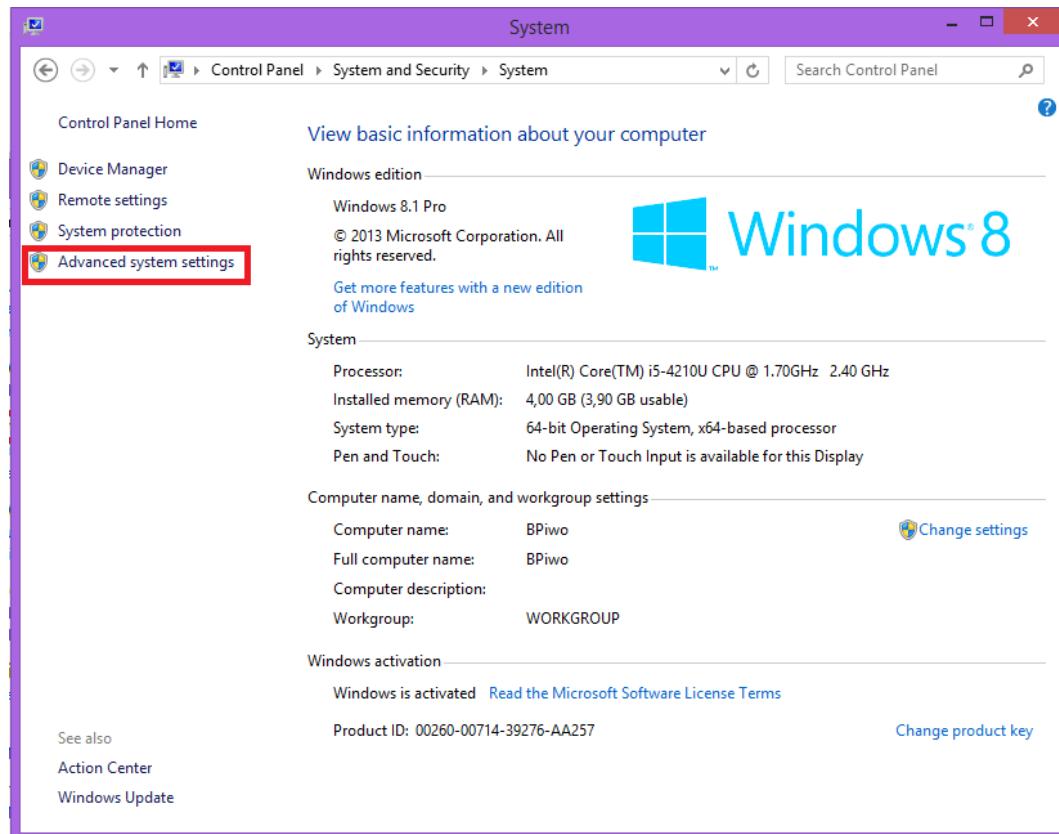
Aby sprawdzić czy całość operacji przebiegła pomyślnie należy otworzyć „Wiersz Poleceń” („Command Prompt”), np. poprzez kliknięcie symbolu Windows na klawiaturze, wpisanie „cmd” i otwarcie znalezionej aplikacji. W otwartym oknie wpisujemy komendę „python”, rysunek 5.7. Jeśli jest ona rozpoznana to dostaniemy odpowiedź jak na rysunku 5.8. Jeśli komenda nie zostanie rozpoznana, należy sprawdzić poprzednie kroki oraz prawidłowość ścieżek dodanych do zmiennej „Path”.



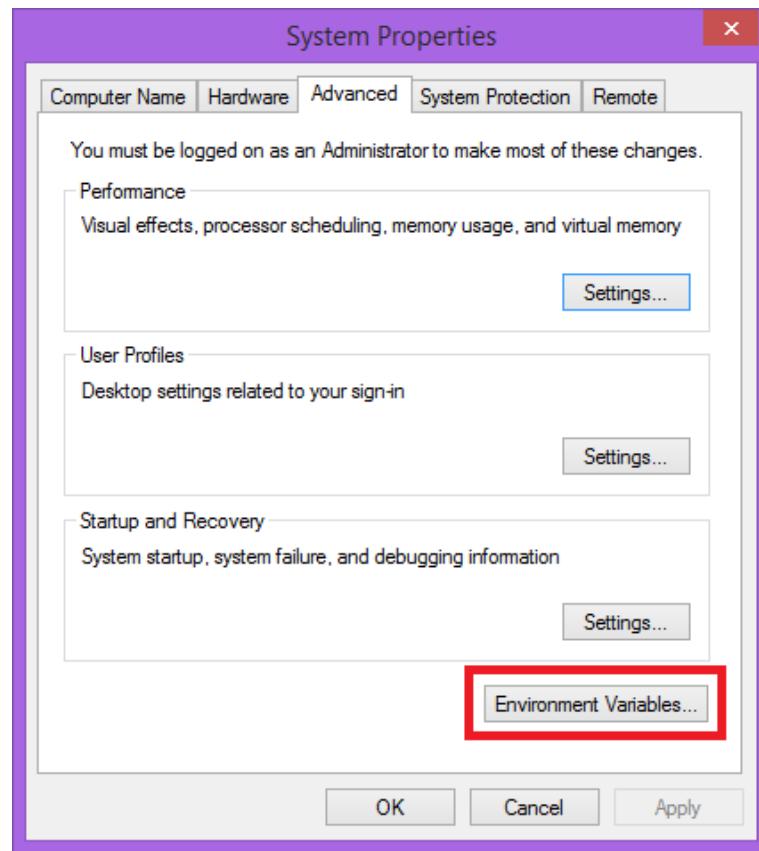
Rys. 5.1. Widok panelu sterowania



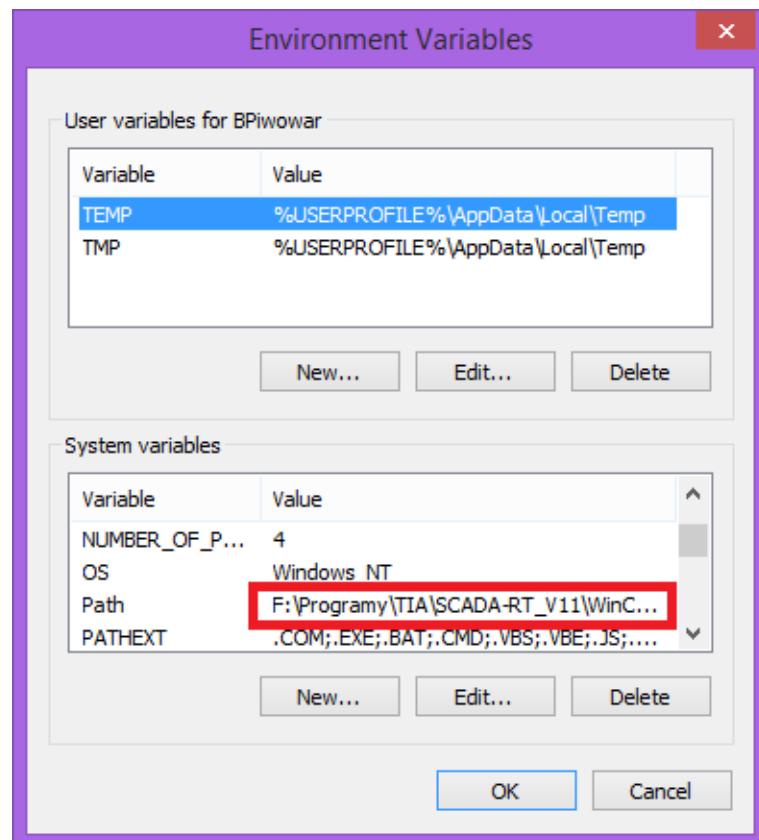
Rys. 5.2. Zakłada System i Zabezpieczenia



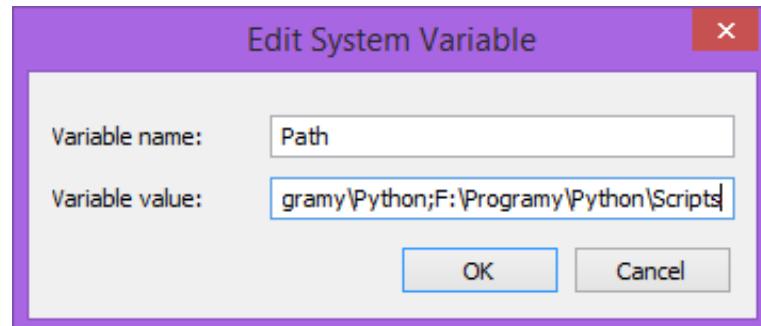
Rys. 5.3. Zakładka System



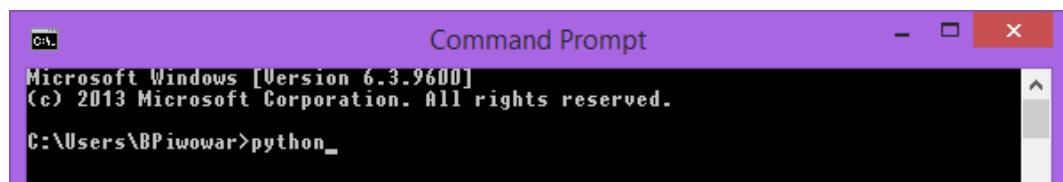
Rys. 5.4. Okno Zaawansowane ustawienia systemu



Rys. 5.5. Okno Zmienne środowiskowe



Rys. 5.6. Okno zmiennej Path



Rys. 5.7. Wiersz Poleceń

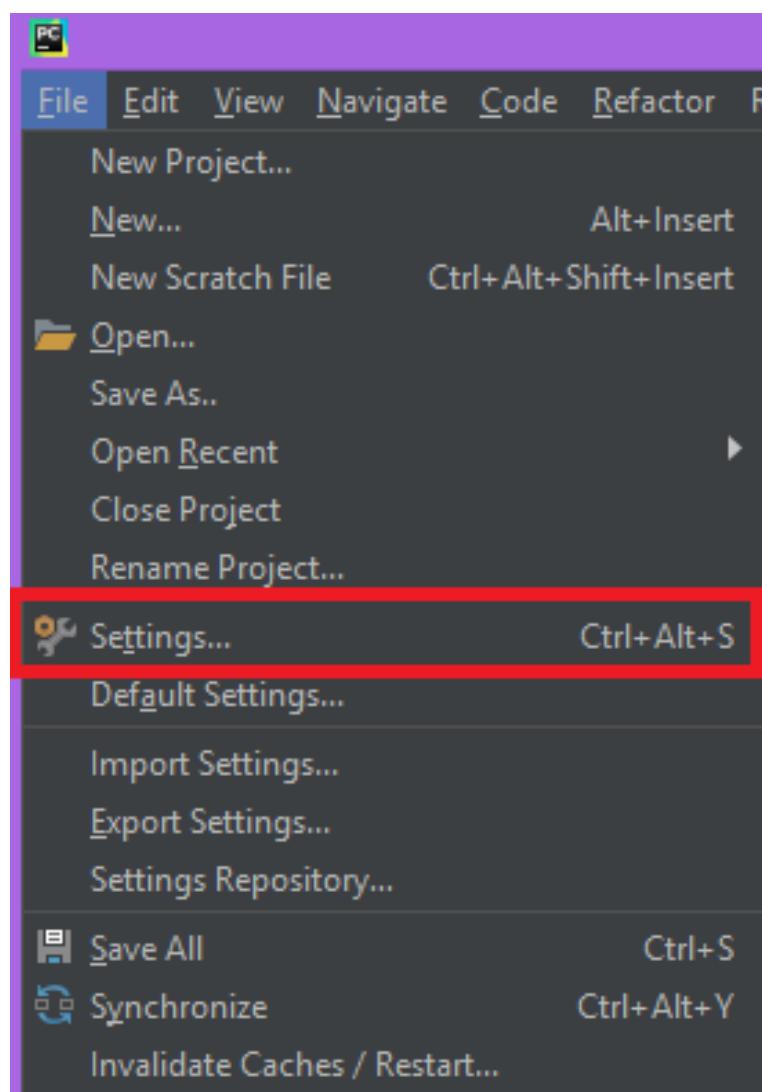


Rys. 5.8. Odpowiedź na komendę „Conda”

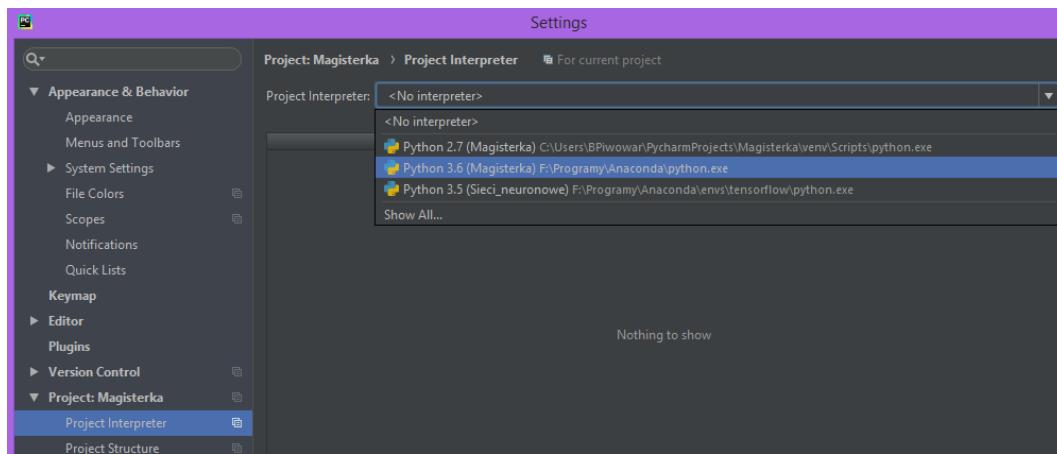
5.1.2. PyCharm

IDE wykorzystywany w projekcie jest PyCharm. Program jest dostępny do pobrania pod linkiem <https://www.jetbrains.com/pycharm/download/#section=windows>. Dodatkowo plik instalacyjny jest zamieszczony na płycie CD dołączonej do pracy. Po zainstalowaniu programu, można otworzyć projekt aplikacji w standardowy sposób tj. File -> Open. W oknie, które się otworzy należy wskazać katalog projektu.

Ostatnia rzecz dotycząca konfiguracji to ustawienie aktywnego interpretera dla projektu. W tym celu należy wejść w ustawienia („Settings”), rysunek 5.9. Następnie otwieramy zakładkę „Project”, klikamy na „Project interpreter” i w pasku wyboru na górze wybieramy interpreter ze ścieżką, gdzie zainstalowaliśmy Pythona, rysunek 5.10.



Rys. 5.9. Okno główne projektu w programie PyCharm



Rys. 5.10. Wybór interpretera

5.2. Potrzebne biblioteki

KATARZYNA RUGIEŁŁO

Poniżej znajduje się lista bibliotek, niezbędnych do korzystania ze zbudowanej aplikacji.

1. numpy
2. matplotlib
3. pygame
4. os
5. scipy
6. time
7. sumpy

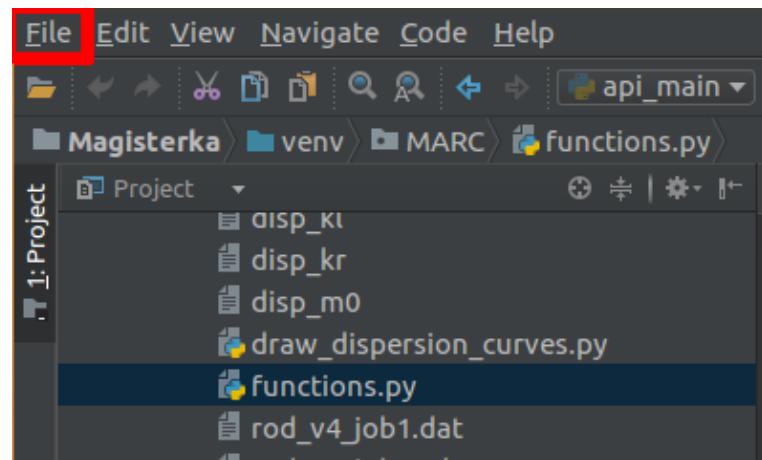
W przypadku, w którym którakolwiek z bibliotek nie była zainstalowana można to zrobić bezpośrednio z poziomu IDE. W tym celu należy z paska górnego wybrać zakładkę File

Następnie z paska wybrać opcję Settings

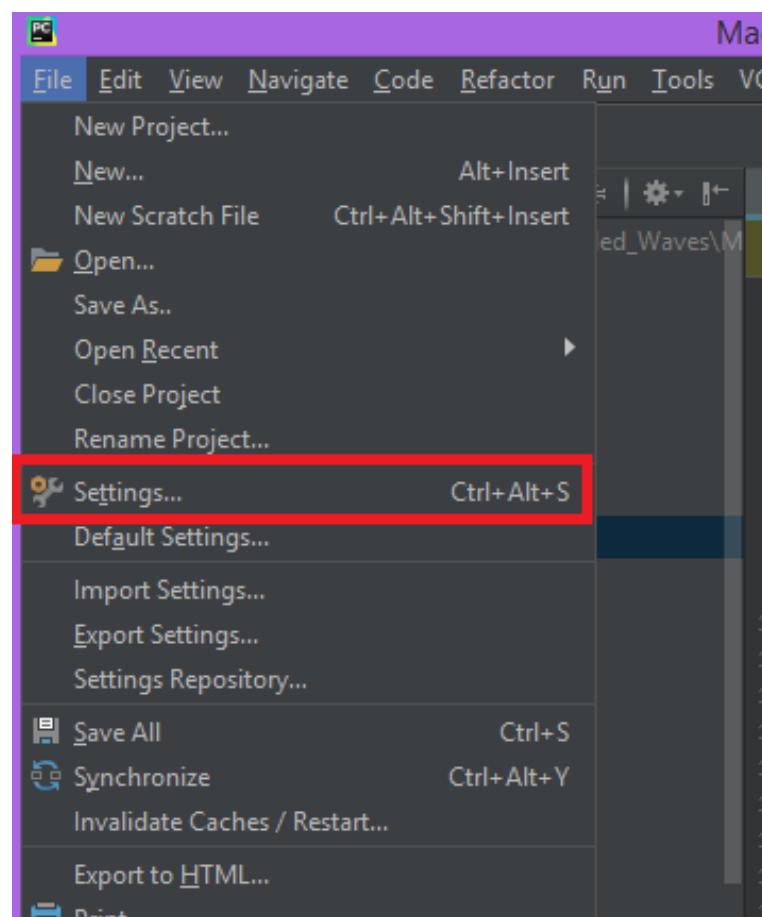
W nowo otwartym oknie należy znaleźć zakładkę z nazwą bieżącego projektu.

Z rozwijanego menu należy wybrać "Project interpreter". Następnie należy nacisnąć prawym przyciskiem myszy na symbolu zielonego plusa.

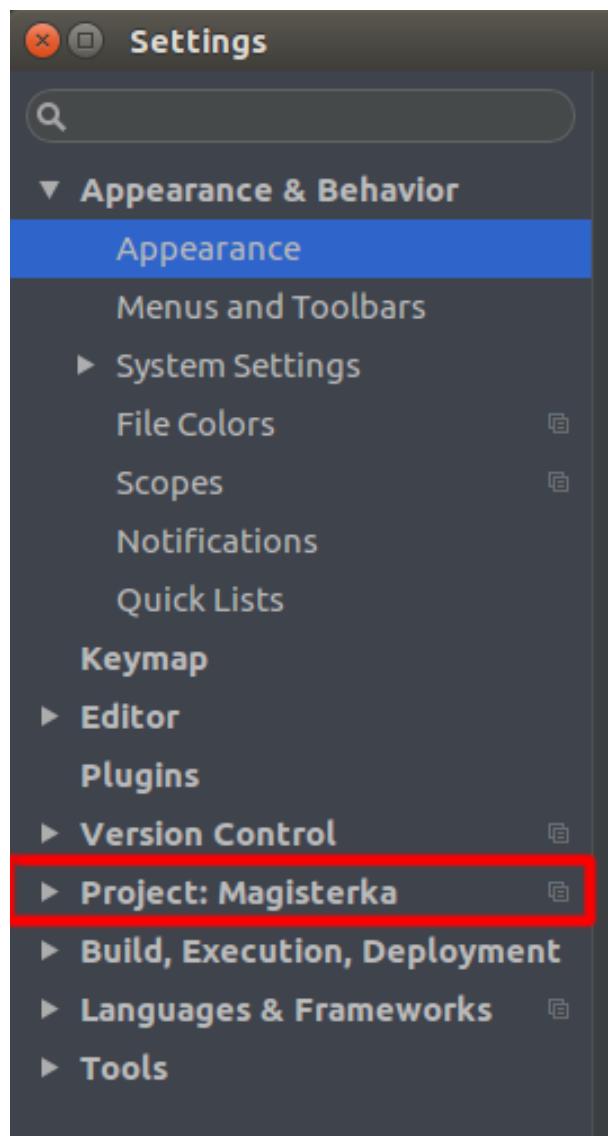
W nowo otwartym oknie w polu wyszukiwania, należy wpisać nazwę biblioteki, która ma zostać zainstalowana. Następnie należy nacisnąć przycisk "Install Package"



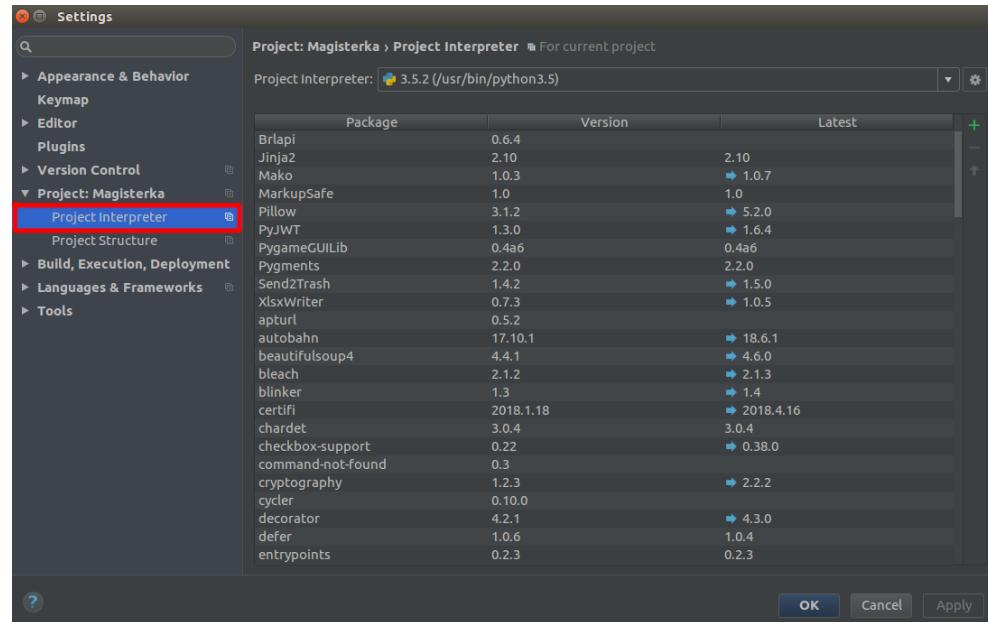
Rys. 5.11. zakładka "File"



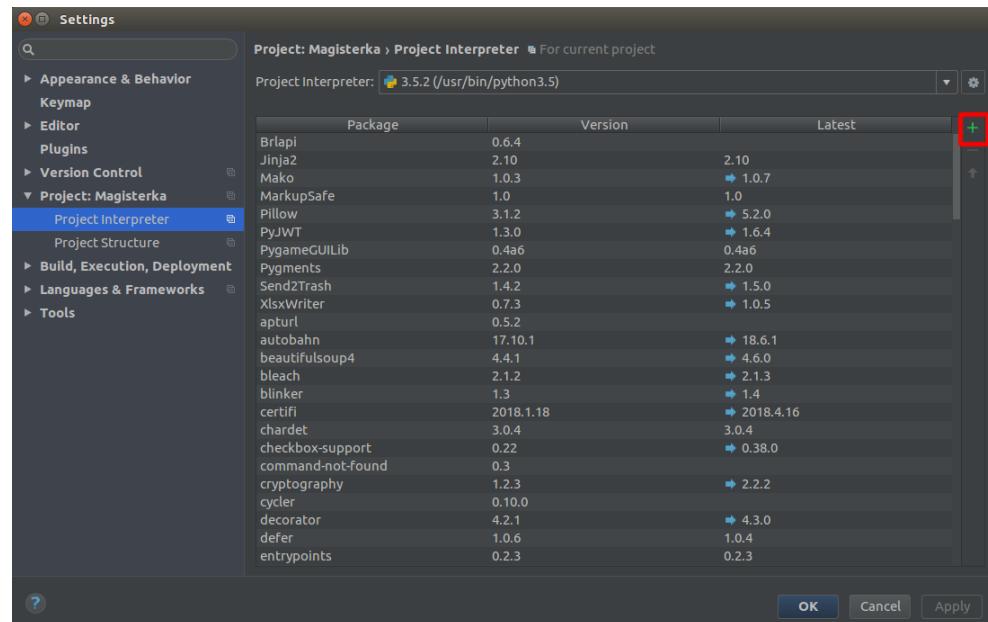
Rys. 5.12. zakładka "Settings"



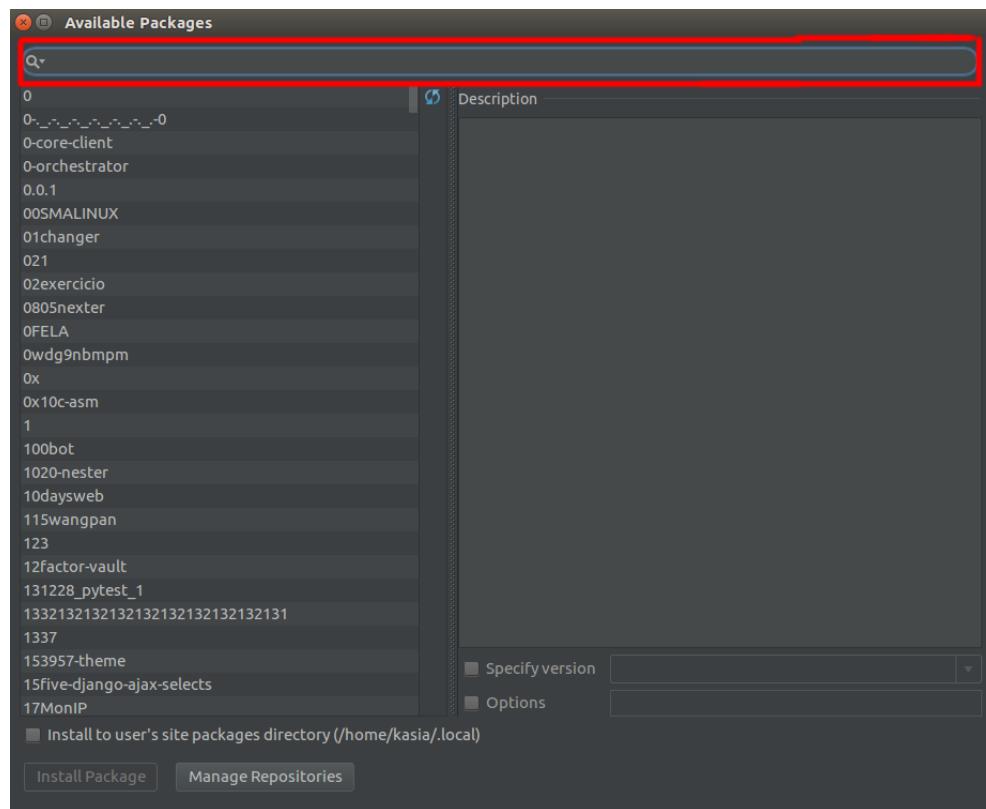
Rys. 5.13. Okno ustawień



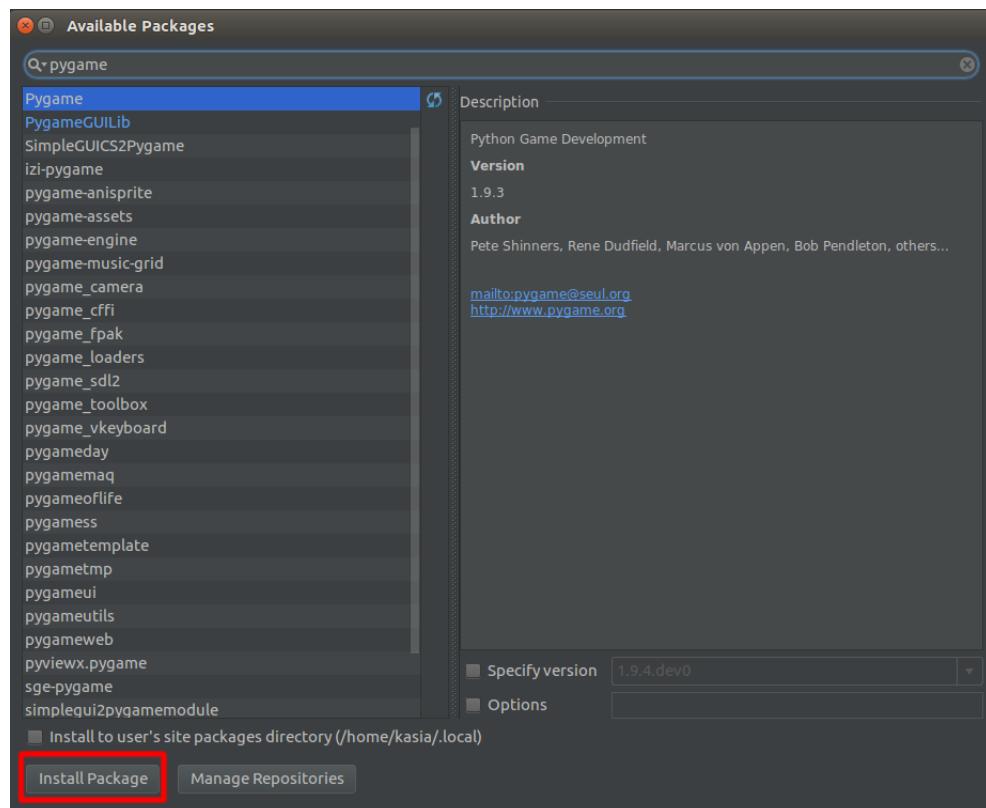
Rys. 5.14. Okno interpretera



Rys. 5.15. Ikona instalacji



Rys. 5.16. Wyszukiwanie bibliotek



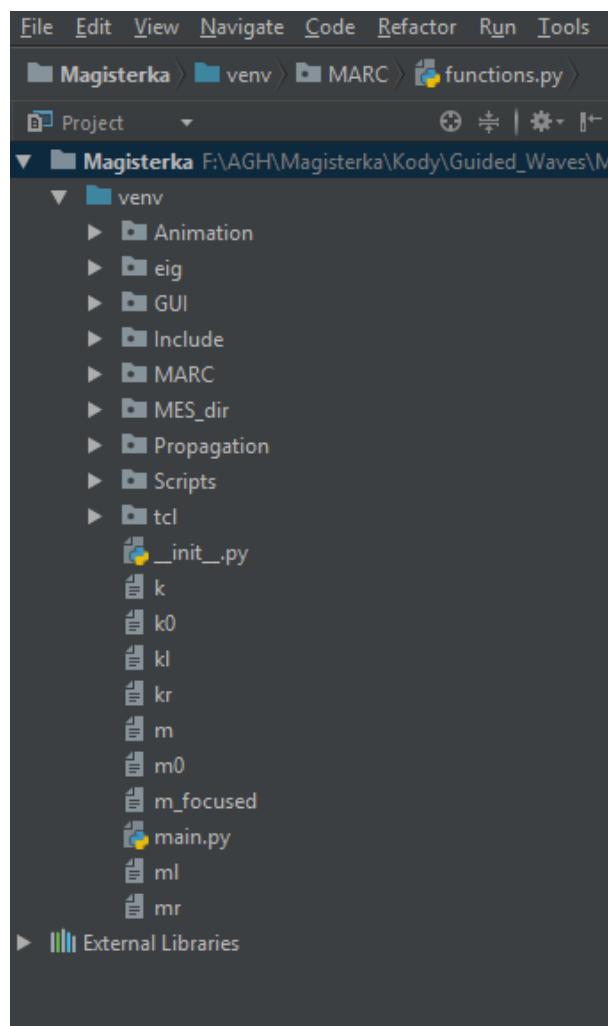
Rys. 5.17. Instalacja

5.3. Obliczenia MES i wyznaczanie krzywych dyspersji oraz wzbudzalności

BARTŁOMIEJ PIWOWARCZYK

W tej sekcji przedstawione są kolejno sposoby tworzenia siatki węzłów, budowy elementów skończonych, wyznaczania lokalnych i globalnych macierzy modelu MES pręta i wyznaczani krzywych dyspersji oraz wzbudzalności. Program umożliwia tworzenie modelu MES z wykorzystaniem elementów czworościennych oraz sześciennych. Krzywe dyspersji oraz wzbudzalności można wyznaczyć z obliczonego modelu, ale jest też opcja wczytania danych modelu z programu MARC.

Strukturę projektu przedstawia rysunek 5.18. Implementacja zagadnień związanych z obliczeniami MES oraz wyznaczaniem krzywych dyspersji z obliczonego modelu znajduje się w katalogu MES_dir. Katalog MARC zawiera funkcje umożliwiające wczytanie danych z programu MARC i na ich podstawie wyznaczeniu krzywych dyspersji.

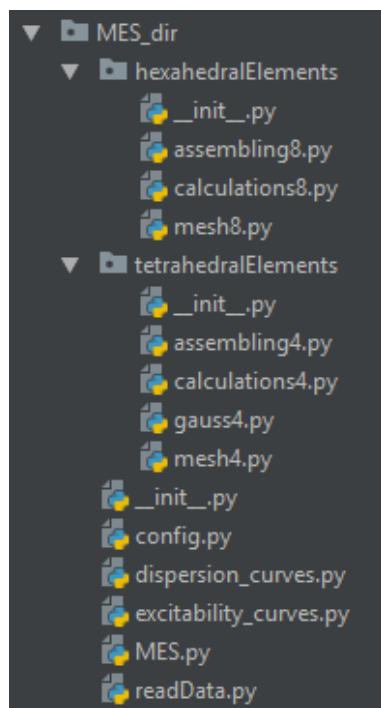


Rys. 5.18. Widok okna projektu

5.3.1. Elementy czworościennne

Zawartość katalogu MES dir przedstawia rysunek 5.19. Obliczenia dotyczące elementów czworościennych zawarte są w modułach katalogu tetrahedralElements. Poniżej znajdują się funkcje poszczególnych modułów, wraz z opisem ich zastosowania, argumentami wejściowymi oraz wyjściowymi.

Dane zbierane w postaci macierzy są najczęściej tablicami array z biblioteki NumPy, która służy do obliczeń numerycznych. W części modułów obliczenia są prowadzone na zmiennych symbolicznych z wykorzystaniem biblioteki SymPy. Macierze zmiennych symbolicznych są obiektami Matrix z tej biblioteki.



Rys. 5.19. Zawartość katalogu MES_dir

Moduł mesh4.

circlePlaneVertices(x, radius, numberOfPoints) - funkcja wykorzystywana przy tworzeniu siatki (w *circleMeshFull* oraz *circleMeshSparse*). Dodaje do siatki węzły na okręgu w jednej płaszczyźniej pręta, która znajduje się na długości x. Promień okręgu określa - radius, a ilość węzłów na okręgu - numberOfPoints.

circleMeshFull(radius, numberOfCircles, numberOfPoints) - funkcja tworzy siatkę na trzech płaszczyznach przesuniętych o 1 we współrzędnej x. Siatka zbudowana jest na każdej płaszczyźnie tak samo i zawiera węzeł centralny oraz okręgi z węzłami w ilości *numberOfCircles*. Na każdym okręgu liczba węzłów jest większa, aby zapewnić możliwie

równe odległości pomiędzy węzłami. Pierwszy okrąg zawiera liczbę węzłów `numberOfPoints`. Zwraca tablicę o wymiarach $n \times 3$, gdzie n to liczba węzłów. W kolumnach są kolejne współrzędne węzłów.

`circleMeshSparse(radius, numberOfCircles, numberOfPoints)` - jak wyżej, z tą różnicą, że każdy okrąg siatki ma tyle samo węzłów.

`triangulation(vertices)` - funkcja tworzy elementy skończone czworościenne. Przyjmuje macierz węzłów z powyższych funkcji - `vertices` i zwraca tablicę $e \times 4$, gdzie e to liczba elementów. W kolumnach zawarte są indeksy węzłów z macierzy wejściowej, które należą do danego elementu.

`correctVolumeSign(vertices, indices)` - funkcja przyjmuje macierz współrzędnych węzłów - `vertices` oraz indeksy węzłów dla każdego elementu skończonego - `indices`. Sprawdza czy objętość elementu skończonego obliczona za pomocą wyznacznika ma dodatnią wartość. Jeśli nie, to zmienia miejscami dwa indeksy elementu w macierzy zwróconej z `triangulation(vertices)`.

`drawPlane(vertices)` - funkcja przyjmuje macierz współrzędnych węzłów - `vertices` i rysuje ich układ na płaszczyźnie. Przykładowe układy przedstawione są na rysunku 5.20.

`drawBar(vertices)` - funkcja przyjmuje macierz współrzędnych węzłów - `vertices` i rysuje wszystkie węzły w rzucie izometrycznym. Przykład przedstawia rysunek 5.21.

`drawTriangulation(vertices, indices)` - funkcja przyjmuje macierz współrzędnych węzłów - `vertices` i macierz elementów skończonych - `indices`. Rysuje elementy skończone w rzucie izometrycznym. Przykład przedstawia rysunek 5.22.

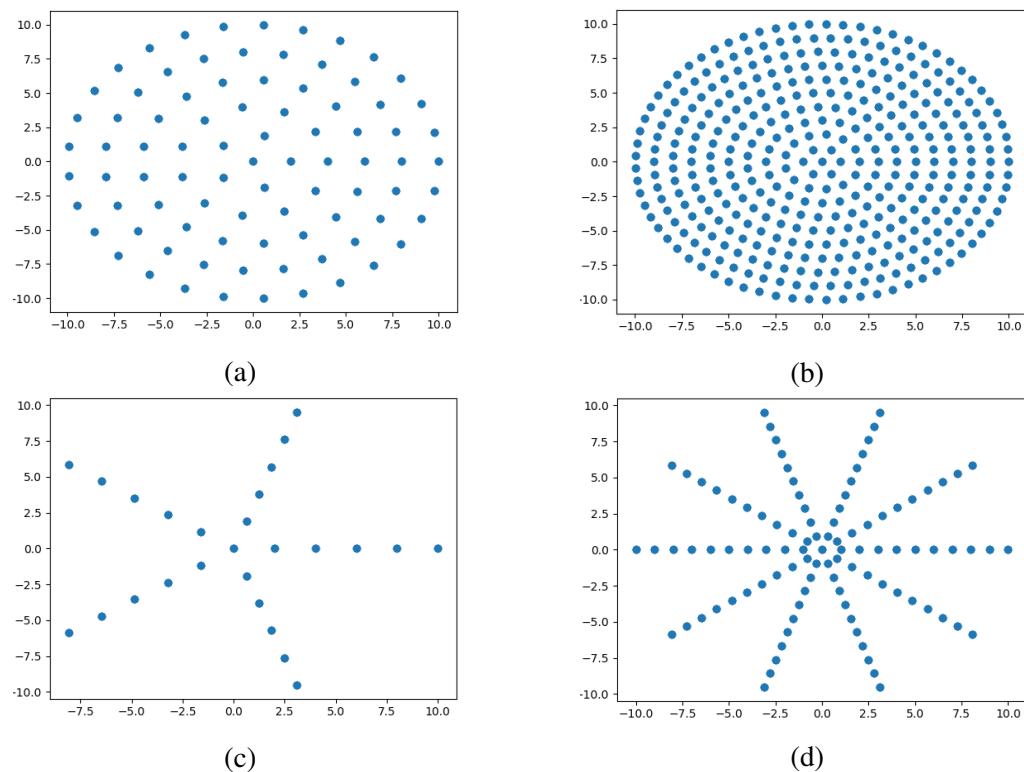
Moduł calculation4.

`pVector()` - funkcja zwraca macierz zmiennych symbolicznych z wektorem p dla elementu czworościenego.

`meMatrix(vertices, elementIndices)` - funkcja przyjmuje macierz współrzędnych węzłów - `vertices` oraz indeksy dla węzłów elementu skończonego - `elementIndices` i zwraca macierz M^e .

`meInvMatrix(vertices, elementIndices)` - jak powyżej, dodatkowo macierz wyjściowa jest odwrócona w stosunku do poprzedniej funkcji.

`shapeFunctions(vertices, elementIndices)` - funkcja przyjmuje macierz współrzędnych węzłów - `vertices` i indeksy dla węzłów elementu skończonego - `elementIndices`, a zwraca funkcje kształtu w formie tablicy zmiennych symbolicznych.



Rys. 5.20. Układ siatki węzłów na płaszczyźnie powstały z a) *circleMeshFull(10, 5, 5)* b) *circleMeshFull(10, 10, 10)* c) *circleMeshSparse(10, 5, 5)* d) *circleMeshSparse(10, 10, 10)*

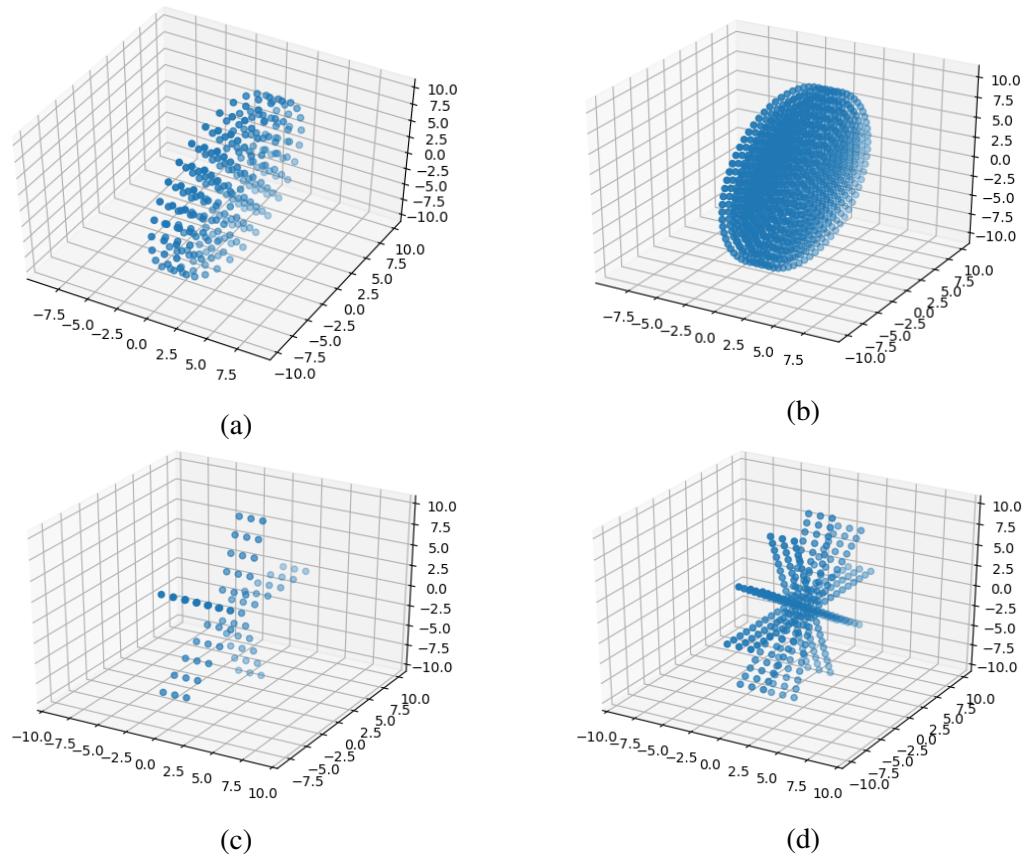
bMatrixFc(shapeFunctions) - funkcja przyjmuje tablicę funkcji kształtu w postaci symbolicznej - *shapeFunctions*, a zwraca macierz B w postaci tablicy numerycznej. Funkcje kształtu są dla elementów czworościennych liniowe więc ich pochodne są stałymi.

bMatrixNatural(shapeFunctions) - jak powyżej, ale dla tablicy funkcji kształtu we współrzędnych naturalnych.

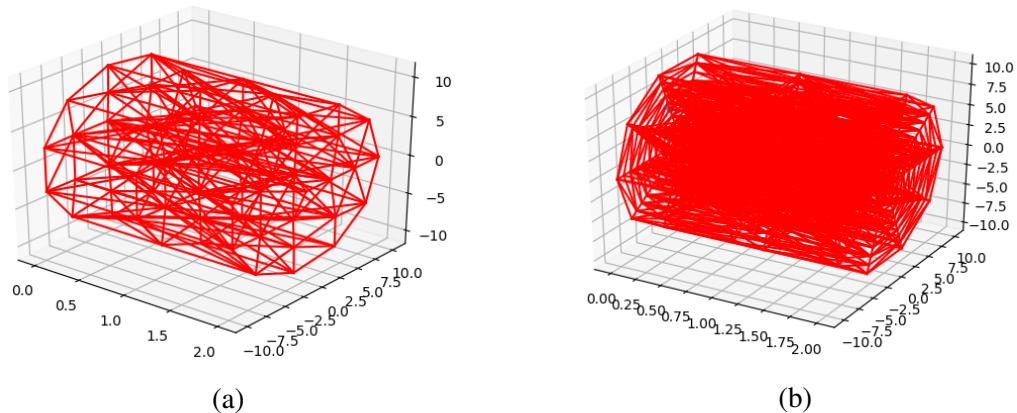
dMatrixFc(youngModulus, poissonCoeficient) - funkcja zwraca macierz D, obliczoną na podstawie modułu Younga - *youngModulus* oraz współczynnik Poissona - *poissonCoeficient*.

stiffLocalMatrix(shapeFunctions, vertices, elementIndices, youngModulus, poissonCoefficient) - funkcja przyjmuje jako argumenty funkcje kształtu w formie symbolicznej - *shapeFunctions*, macierz współrzędnych węzłów - *vertices*, indeksy węzłów elementu skończonego - *element indices* oraz moduł Younga - *youngModulus* i współczynnik Poissona - *poissonCoeficient*. Zwraca macierz sztywności dla elementu skończonego.

massLocalMatrix(density) - funkcja przyjmuje gęstość materiału - *density*. Zwraca macierz mas obliczoną we współrzędnych naturalnych bez uwzględnienia jakobianu. Na tym etapie nie jest to więc poprawnie wyznaczona macierz mas. Jakobian jest stały dla elementów czworościennych i uwzględniany jest na etapie agregacji. Pozwala to obliczyć



Rys. 5.21. Układ siatki węzłów pręta z a) $circleMeshFull(10, 5, 5)$ b) $circleMeshFull(10, 10, 10)$ c) $circleMeshSparse(10, 5, 5)$ d) $circleMeshSparse(10, 10, 10)$



Rys. 5.22. Triangulacja (elementy skończone) dla siatki a) $circleMeshFull(10, 3, 3)$ b) $circleMeshSparse(10, 10, 10)$

całkę 3.26 bez uwzględnienia jakobianu raz, a następnie mnożyć ją dla każdego elementu przez jakobian.

volume(elementVertices) - funkcja przyjmuje współrzędne węzłów elementu - elementVertices i oblicza objętość elementu z wykorzystaniem geometrii analitycznej.

volumeDet(vertices) - jak powyżej z tym, że objętość jest obliczana za pomocą wyznacznika macierzy M^e .

Moduł gauss4. Moduł zawiera funkcje pomocnicze do całkowania macierzy mas. Obliczanie macierzy sztywności nie wymaga całkowania, ponieważ macierz B jest stała.

shapeFunctionsNatural() - zwraca tablicę funkcji kształtu we współrzędnych naturalnych, w formie symbolicznej.

coordinateChangeModel(elementVertices, naturalShapeFc) - funkcja przyjmuje funkcje kształtu we współrzędnych naturalnych - naturalShapeFc oraz współrzędne węzłów elementu skońzonego - elementVertices i oblicza zależność współrzędnych rzeczywistych i naturalnych. Zwraca trzy wyrażenia symboliczne zawierające współrzędne naturalne. Przedstawiają one współrzędne rzeczywiste, kolejno x, y, z.

jacobian(elementVertices, naturalShapeFc) - funkcja przyjmuje współrzędne węzłów elementu skońzonego - elementVertices oraz funkcje kształtu we współrzędnych naturalnych - naturalShapeFc. Zwraca jakobian przekształcenia, który jest wykorzystywany w obliczaniu macierzy mas.

matrixToIntegrate(density) - funkcja przyjmuje gęstość materiału - density. Zwraca macierz podcałkową, do obliczania macierzy mas. Nie uwzględnia jakobianu, który jest stały. Wynik całkowania jest mnożony przez jakobian na etapie agregacji.

Moduł assembling4. Moduł zawiera funkcje do obliczania macierzy globalnych. Pozwala też na wizualizację rzadkości macierzy.

assembleGlobalStiffMatrix(vertices, indices, youngModulus, poissonCoefficient) - funkcja przyjmuje macierz współrzędnych węzłów konstrukcji - vertices, indeksy wszystkich elementów - indices, moduł Younga - youngModulus i współczynnik Poissona - poissonCoefficient. Zwraca globalną macierz sztywności.

drawMatrixSparsity(matrix) - funkcja przyjmuje macierz - matrix. Pozwala rysować rzadkość macierzy w postaci bitmapy, gdzie każdy element macierzy jest jednym pikselem.

assembleGlobalMassMatrix(vertices, indices, density) - funkcja przyjmuje macierz współrzędnych węzłów konstrukcji - vertices, indeksy węzłów wszystkich elementów skończonych - indices oraz gęstość materiału - density. Zwraca globalną macierz mas.

focusMatrixRows(matrix) - funkcja przyjmuje macierz - matrix. Zwraca macierz skupioną poprzez sumowanie elementów w wierszu i umieszczanie ich na diagonali. Wykorzystywana przy obliczeniach ze skupioną macierzą mas.

5.3.2. Elementy sześciennne

Obliczenia dotyczące elementów sześciennych zawarte są w modułach katalogu hexahedralElements. Poniżej znajdują się funkcje poszczególnych modułów, wraz z opisem ich zastosowania, argumentami wejściowymi oraz wyjściowymi.

Moduł mesh8.

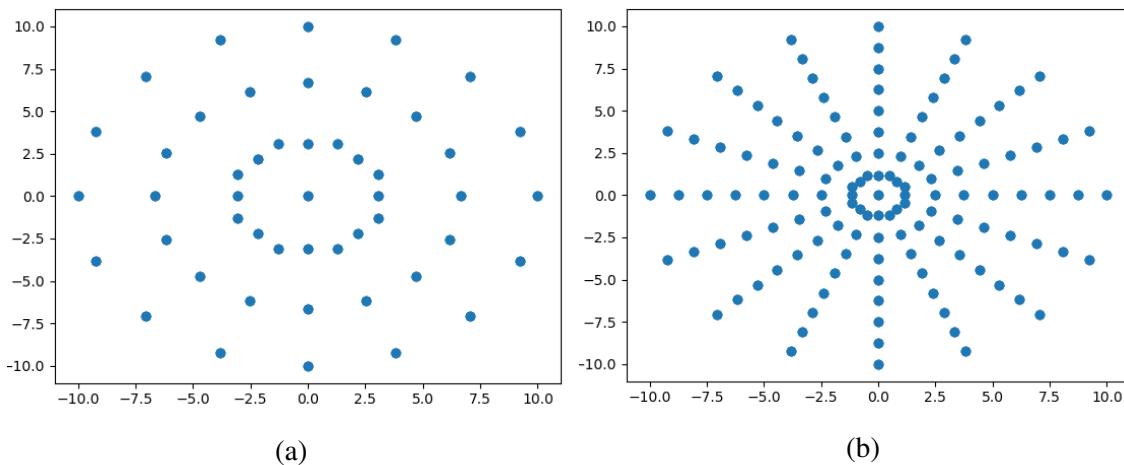
circleMeshFull(radius, firstCircle, addNodes, circles) - funkcja działa jak funkcja z modułu mesh4 z tą różnicą, że przyjmuje dodatkowy argument addNodes. Określa on ile punktów więcej ma być na każdym kolejnym okręgu siatki.

circleMeshSparse(radius, firstCircle, circles) - funkcja działa jak odpowiednik z modułu mesh4

createFiniteElements(vertices, pointsOnLastCircle, length, numberofPlanes) - funkcja przyjmuje jako argumenty macierz współrzędnych węzłów, liczbę punktów na ostatnim okręgu siatki, długość modelu pręta oraz liczbę płaszczyzn siatki. Tworzy elementy sześciennne w kilku etapach. Najpierw jedna z płaszczyzn dzielona jest na trójkąty. Następnie trójkąty są łączone w pary i w ten sposób powstają czworokąty. W większości konfiguracji siatki na brzegach pozostają puste miejsca z trójkątów, które nie mają pary. W takich miejscach dodawany jest dodatkowy punkt siatki i z trójkąta tworzony czworokąt. Następnie identyczne czworokąty są tworzone na kolejnych płaszczyznach i wzdłuż długości pręta tworzone są z nich sześciociany. Zwraca macierz e x 8, gdzie e to liczba elementów skończonych. W kolumnach są indeksy kolejnych punktów siatki.

brickMesh(radius, numberofPlanes, numberofCircles, numberofPointsOnCircle) - funkcja tworzy siatkę o kształcie wielokąta na pierwszym okręgu. Przyjmuje jako argumenty promień - radius, liczbę płaszczyzn siatki - numberofPlanes, liczbę okręgów na płaszczyźnie siatki - numberofCircles oraz ilość wierzchołków wielokąta wpisanego w okrąg - numberofPointsOnCircle. Zwraca tablicę o wymiarach n x 3, gdzie n to liczba węzłów. W kolumnach są kolejne współrzędne węzłów.

createBrickElements(brickVertices, numberofPlanes, numberofCircles, numberofPointsOnCircle) - funkcja przyjmuje macierz współrzędnych węzłów z funkcji *brickMesh* - brickVertices, liczbę płaszczyzn siatki - numberofPlanes, liczbę okręgów na każdej płaszczyźnie - numberofCircles oraz liczbę wierzchołków wielokąta wpisanego w każdy okrąg - numberofPointsOnCircle. Tworzy elementy sześciocienne na zadanej siatce.



Rys. 5.23. Układ siatki węzłów na płaszczyźnie powstały z a) *brickMesh(10, 3, 3, 16)* b) *brickMesh(10, 3, 8, 16)*

Zwraca macierz $e \times 8$, gdzie e to liczba elementów skończonych. W kolumnach są indeksy kolejnych punktów siatki.

drawPlane(vertices) - jak w *mesh4*. Przykładowe siatki z tego modułu są przedstawione na rysunku 5.23.

drawBar(vertices) - jak w *mesh4*

drawTetragons(vertices, indices) - funkcja przyjmuje jako argumenty macierz współrzędnych węzłów - *vertices* oraz macierz indeksów węzłów dla każdego elementu - *indices*. Rysuje układ czworokątów na płaszczyźnie. Przykłady są przedstawione na rysunku 5.24.

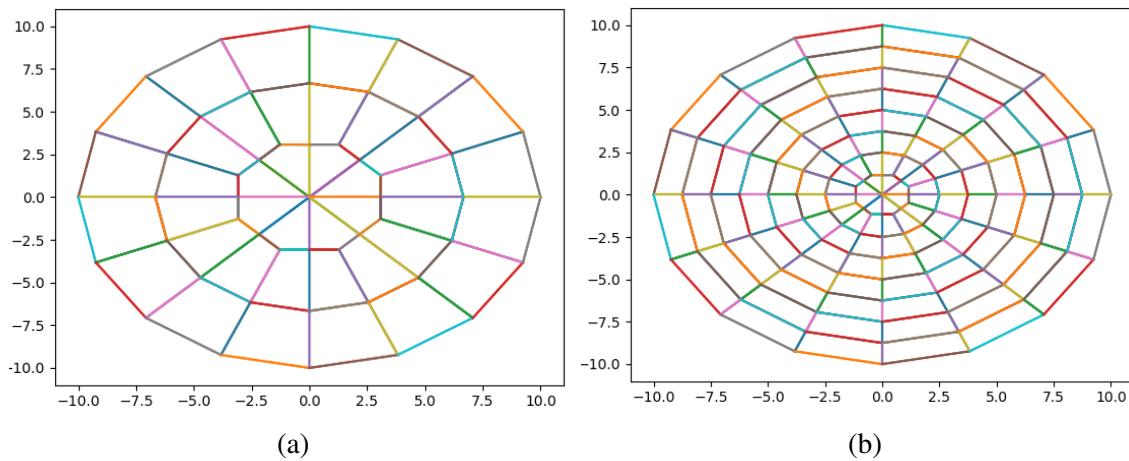
drawHexahedrons(vertices, indices) - funkcja przyjmuje jako argumenty macierz współrzędnych węzłów - *vertices* oraz macierz indeksów węzłów dla każdego elementu - *indices*. Rysuje model złożony z sześciocianów w rzucie izometrycznym. Przykład znajduje się na rysunku 5.25.

Moduł calculations8.

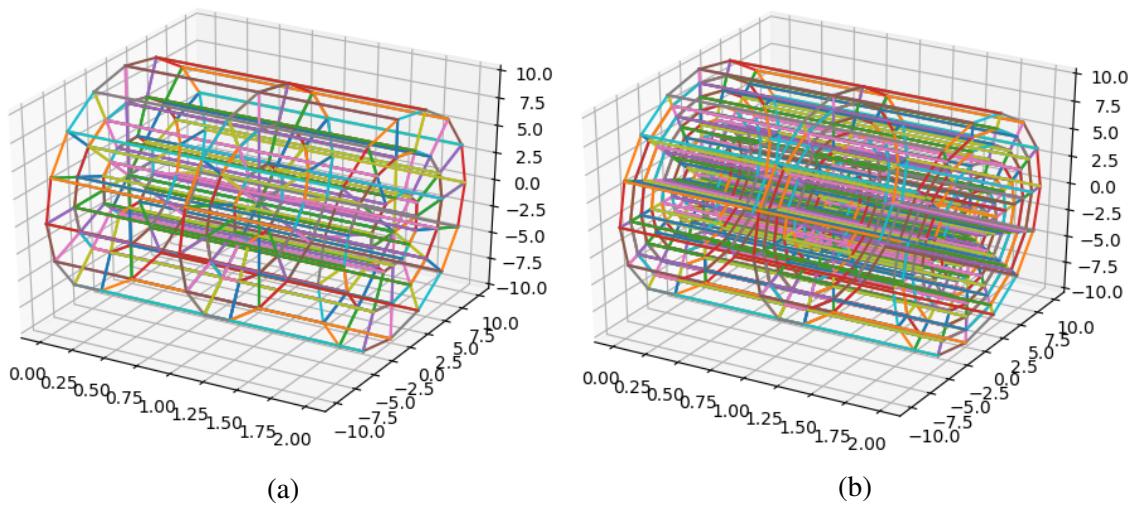
localStiffMatrix(elementVertices) - funkcja przyjmuje jako argumenty macierz współrzędnych węzłów elementu skończonego. Zwraca macierz sztywności elementu skończonego.

localMassMatrix(elementVertices) - funkcja przyjmuje jako argumenty macierz współrzędnych węzłów elementu skończonego - *elementVertices*. Zwraca macierz mas elementu skończonego.

Moduł assembling8.



Rys. 5.24. Czworokąty będące ścianą elementu skończonego na płaszczyźnie a) $brickMesh(10, 3, 3, 16)$ b) $brickMesh(10, 3, 8, 16)$



Rys. 5.25. Elementy skończone zbudowane na siatce a) $brickMesh(10, 3, 3, 16)$ b) $brickMesh(10, 3, 8, 16)$

assembleGlobalStiffMatrix(vertices, indices) - funkcja przyjmuje jako argument macierz współrzędnych węzłów konstrukcji - vertices oraz macierz indeksów węzłów elementów skończonych - indices. Zwraca macierz sztywności konstrukcji.

assembleGlobalMassMatrix(vertices, indices) - funkcja przyjmuje jako argument macierz współrzędnych węzłów konstrukcji - vertices oraz macierz indeksów węzłów elementów skończonych - indices. Zwraca macierz mas konstrukcji.

focuseMatrixRows(matrix) - funkcja przyjmuje macierz - matrix. Zwraca macierz skupioną poprzez sumowanie elementów w wierszu i umieszczanie ich na diagonali. Wykorzystywana przy obliczeniach ze skupioną macierzą mas.

drawMatrixSparsity(matrix) - funkcja przyjmuje macierz - matrix. Pozwala rysować rzadkość macierzy w postaci bitmapy, gdzie każdy element macierzy jest jednym pikselem.

5.3.3. Pozostałe moduły katalogu MES_dir

Moduł config.

W tym module przechowywane są dane wykorzystywane w innych częściach programu. Jego zawartość podana jest poniżej.

```
ROOT_DIR = os.path.dirname(os.path.abspath(__file__))
#sciezka do katalogu MES_dir ndof = 3 # liczba stopni swobody

kvect_min = 0
kvect_no_of_points = 0
kvect_max = 2*pi k = [] # globalna macierz sztywnosci

m = [] # globalna macierz mas
m.Focused_rows = [] #globalna macierz mas - skupiona wierszami
ml = []
m0 = []
mr = []
kl = []
k0 = []
kr = [] force = [] # Stale materialowe

young_mod = 70000
poisson_coef = 0.3
density = 2.7*1e-9 # Wyswietlanie siatki na plaszczyznie, siatki w 3D i triangulacji

show_plane = False
show_bar = False
```

```
show_elements = False
saveEigVectors = False
```

Pierwszym elementem jest ścieżka do katalogu pozwalająca wygodnie odnosić się do niego w funkcjach zapisujących i wczytujących dane z plików. Poniżej znajdują się wartości minimalna, maksymalna oraz liczba próbek wartości liczby falowej, dla których to wartości obliczane będą częstotliwości własne. Następnie kolejno zawarte są macierze i podmacierze modelu MES, siła wymuszająca wykorzystywana przy obliczaniu wzbudzalności, stałe materiałowe konstrukcji pręta i wartości logiczne określające czy wyświetlać siatkę lub element skończone modelu.

Moduł MES. W tym module zawarte są dwie funkcje, które są przykładami budowy modelu za pomocą elementów czworościennych oraz sześciościennych.

mes4(radius, numOfCircles, numOfPointsAtFirstCircle) - funkcja przyjmuje jako argumenty promień pręta - radius, liczbę okręgów na jednej płaszczyźnie siatki - numOfCircles oraz liczbę węzłów na pierwszym okręgu siatki - numOfPointsAtFirstCircle. Poniżej znajduje się całość kodu tej funkcji.

```
vertices = mesh4.circleMeshFull(radius, numOfCircles, numOfPointsAtFirstCircle)
if config.show_plane:
    mesh4.drawPlane(vertices)
if config.show_bar:
    mesh4.drawBar(vertices)

indices = mesh4.triangulation(vertices)
# mesh.draw_triangulation(vertices, indices)
if config.show_elements:
    mesh4.drawTriangulation(vertices, indices)

start = time.clock()
config.k = assembling4.assembleGlobalStiff_matrix(vertices, indices, config.young_mod, config.poisson_coef)
# assembling.drawMatrixSparsity(config.k)
print("Macierz sztywnosci gotowa")
print("Wykonywanie: ", time.clock() - start)

config.m = assembling4.assembleGlobalMassMatrix(vertices, indices, config.density)
config.m_focused_rows = assembling4.focusMatrixRows(config.m)
# assembling.drawMatrixSparsity(config.m)
print("Macierz mas gotowa")
print("wykonywanie: ", time.clock() - start)
```

W pierwszej linii uzyskiwana jest macierz współrzędnych węzłów siatki. Następnie siatka jest wyświetlana w na płaszczyźnie, bądź w rzucie izometrycznym jeśli wartości z modułu config mają wartości TRUE. Kolejnym etapem jest tworzenie elementów skończonych i warunkowe ich wyświetlanie. Wartość start przechowuje czas, w którym rozpoczyna się obliczanie macierzy sztywności - k. Po obliczeniu tej macierzy wyświetlany jest komunikat z czasem obliczeń. Podobnie poniżej w przypadku macierzy mas.

mes8(numberOfPlanes, radius, numberOfCircles, numberOfPointsOnCircle, addNodes) - funkcja przyjmuje jako argumenty liczbę płaszczyzn siatki - *numberOfPlanes*, promień pręta - *radius*, liczbę okręgów na jednej płaszczyźnie siatki - *numberOfCircles*, liczbę węzłów na pierwszym okręgu siatki - *numberOfPointsOnCircle* oraz liczbę dodatkowych punktów na każdym kolejnym okręgu siatki przy zastosowaniu odpowiedniego jej typu - *addNodes*. Poniżej znajduje się całość kodu tej funkcji.

```
# brickMesh(radius, numberOfPlanes, numberOfCircles, numberOfPointsOnCircle)
vertices = mesh8.brickMesh(radius, numberOfPlanes, numberOfCircles, numberOfPointsOnCircle)

# createBrickElements(brickVertices, numberOfPlanes, numberOfCircles, numberOfPointsOnCircle)
indices = mesh8.createBrickElements(vertices, 3, 3, 16)

start = time.clock()
config.k = assembling8.assembleGlobalStiffMatrix(vertices, indices)
print("Macierz sztywnosci gotowa")
print("Wykonywanie: ", time.clock() - start, "[s]")
print("Wykonywanie: ", (time.clock() - start)/3600, "[h]")
start = time.clock()

config.m = assembling8.assembleGlobalMassMatrix(vertices, indices, config.density)
config.m_focused_rows = assembling8.focused_matrix_rows(config.m)
print("Macierz mas gotowa")
print("wykonywanie: ", time.clock() - start, "[s]")
print("wykonywanie: ", (time.clock() - start)/3600, "[h]")
```

W tym przykładzie zastosowano siatkę brickMesh. Po wyznaczeniu współrzędnych węzłów oraz zbudowaniu elementów skończonych, obliczane są macierze mas i sztywności.

Moduł dispersion_curves. W tym module wyznaczone są punkty krzywych dyspersji, na podstawie macierzy mas i sztywności wyznaczonych z modelu MES.

getDataForEq() - funkcja wyznacza podmacierze mas i sztywności potrzebne do zastosowania wzoru 2.43. Macierze przechowywane są w module *config*. Dodatowo po

wyznaczeniu zapisywane są do plików tekstowych, tak więc nie ma potrzeba wyznaczania ich ponownie w celu innego wykorzystania.

findEig() - funkcja dla kolejnych wartości liczby falowej oblicza wartości i wektory własne pary macierzy, wyznaczonych jak we wzorze 2.43. Dodatkowo wektory własne zapisywane są w niej do pliku.

drawDispersioCurves(number_of_curves_to_draw=10, save_plot_to_file=False) - funkcja przyjmuje jako argument liczbę początkowych modów do wyświetlenia - *number_of_curves_to_draw* oraz wartość logiczną określającą czy zapisać wykres do pliku png - *save_plot_to_file*. Dodatkowo zapisuje wszystkie wartości własne w pliku tekstowym.

drawDispersioCurvesFromFile(number_of_curves_to_draw=10, save_plot_to_file=False) - jak powyżej z tym, że funkcja ta służy do rysowania krzywych dyspersji z wartości wczytywanych z plików tekstowych.

sortColumns(matrix) - funkcja służy do wstępnego sortowania wartości własnych. Sortuje je kolumnami od najmniejszej do największej. W każdej kolumnie zapisane są wartości własne dla jednej wartości liczby falowej. Sortowanie takie jest więc poprawne tylko dla modów początkowych, które się nie krzyżują. Bardziej zaawansowany algorytm sortowania jest wprowadzany na etapie obliczeń związanym z wykorzystywaniem krzywych dyspersji.

Moduł *readData*. W tym module zawarte są wszystkie funkcje służące do zapisu i wczytywania danych. Nie będą one z osobna omawiane ponieważ ich funkcja jest jasna. Poniżej omówiony jest sposób umieszczania danych w plikach.

Wszystkie dane umieszczone są w katalogu *eig*. Liczby falowe zapisane są w pliku *kvect*. Każda wartość znajduje się w osobnej linii.

Wartości własne zapisane są w pliku *omega*. Każda kolumna zawiera wartości własne wyznaczone dla jednej wartości liczby falowej.

Wektory własne zapisywane są w katalogu *eig*, w plikach mających w nazwie *eig_*, a następnie wartość liczby falowej. W każdym z takich plików zapisane są wartości własne i odpowiadające im wektory własne dla jednej wartości liczby falowej. W pierwszej linii znajduje się wartość własna, w drugiej odpowiadający jej wektor własny, a następnie pozostałe wartości i wektory własne w kolejnych liniach.

5.3.4. Moduł **main**

Przykładowy skrypt wyznaczający krzywe dyspersji, z pomocą wcześniej opisanych modułów, znajduje się w module **main** głównego katalogu projektu. Kod przedstawiony jest poniżej.

```
import sympy as sp
import numpy as np
from MES_dir import MES, config, dispersion_curves
from MARC import functions

# begin MES
x, y, z = sp.symbols('x, y, z')
if __name__ == "__main__":

    print("Wpisz wartość: ")
    print("1 - rysowanie krzywy dyspersji z wykorzystaniem MES")
    print("2 - rysowanie krzywych dyspersji z ostatnio policzonych danych")
    text = input()

    if text == '1':

        print("Wpisz wartość: ")
        print("4 - elementy czworościenne")
        print("8 - elemnty sześciennne")
        print("M - wczytanie macierzy z MARC i wykreślenie krzywych")
        text1 = input()

        # wektor liczby falowej
        config.kvect_min = 1e-10
        config.kvect_max = np.pi / 4
        config.kvect_no_of_points = 51

        # rysowanie wykresow

        config.show_plane = True
        config.show_bar = True
        config.show_elements = True

        # zapisywanie wektorow własnych saveEigVectors = True

        # obliczenia
        if text1 == '4':
            # parametry preta
            radius = 10
            num_of_circles = 4
            num_of_points_at_c1 = 4
            MES.mes4(radius, num_of_circles, num_of_points_at_c1)

        if text1 == '8':
```

```
radius = 10
numberOfPlanes = 3
firstCircle = 16 #for brickMesh should be 16
addNodes = 0
circles = 1
MES.mes8(numberOfPlanes, radius, circles, firstCircle, addNodes)

if text1 == 'M':
    config.k, config.m = functions.getStiffAndMassMatrix()
    dispersion_curves.drawDispercionCurves()
    print("koniec")

# rysowanie krzywych dyspersji z wcześniejszych obliczonych wartości
if text == '2':
    dispersion_curves.drawDispercionCurvesFromFile()
```

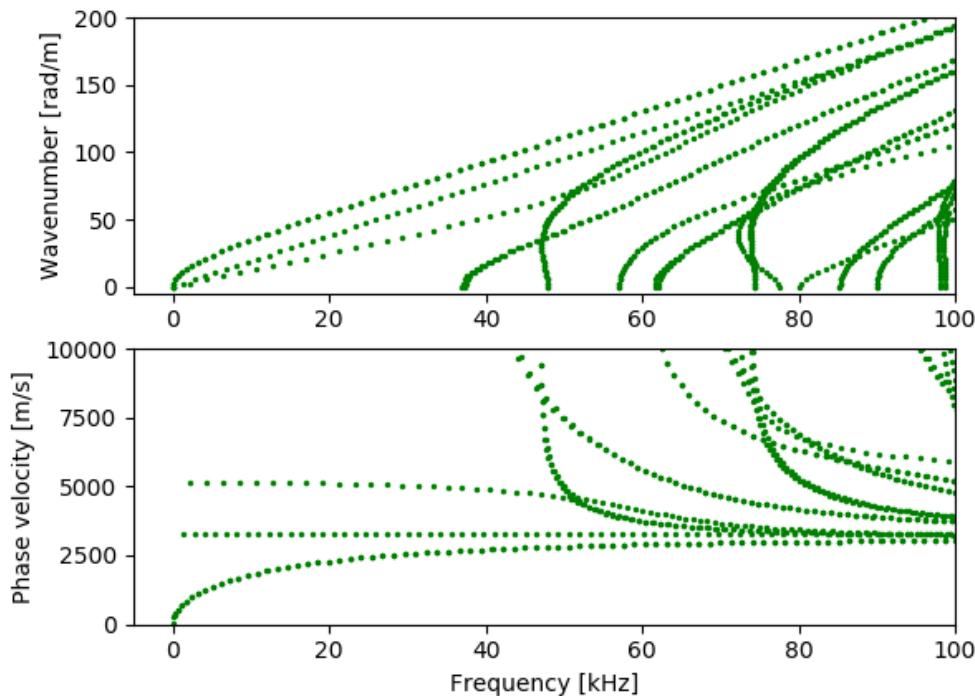
Pierwsze linijki zapewniają dostęp do funkcji z potrzebnych modułów programu oraz bibliotek Python-a. Następnie znajduje się definicja zmiennych symbolicznych, które są wykorzystywane w programie oraz zmienne określające czy wyświetlać wykresy rozmieszczenia węzłów siatki, elementów skończonych, a także czy zapisywać do plików wektory własne. Ostatnia z tych operacji jest czasochłonna i domyślnie jest wyłączona. W dalszej części następuje seria warunków, które pozwalają na wybór obliczania nowych wartości własnych (text=1), bądź skorzystana z wcześniejszych wyznaczonych do wykreślenia krzywych dyspersji (text=2). Wyboru dokonujemy poprzez wpisane odpowiedniew cyfry w konsoli programu.

Jeśli chcemy obliczyć nowe krzywe, to będziemy mogli skorzystać z budowania modelu MES przy pomocy elementów czworościennych (text1=4), elementów sześciennych (text1=8), bądź skorzystać z modelu wyznaczonego w programie MARC (text1=M) i zapisanego w postaci plików z rozszerzeniem dat w katalogu *MARC*. Niezależnie od wybranego sposobu dostarczania modelu, program w kolejnej fazie przejdzie do wyznaczania krzywych dyspersji.

Siatkę dla elementów skończonych możemy dostosować przez zmianę odpowiednich wartości przyjmowanych jako argumenty funkcji *mes4* oraz *mes8*. Dla zmiany rodzaju siatki należy wybrać inną funkcję do jej generacji w funkcji *mes4* lub *mes8*. Należy przy tym pamiętać żeby dostosować także funkcję budującą elementy skończone. Przykładowo dla siatki tworzonej przy pomocy *brickMesh*, powinna to być funkcja *createBrickElements*.

5.3.5. Wyniki - krzywe dyspersji

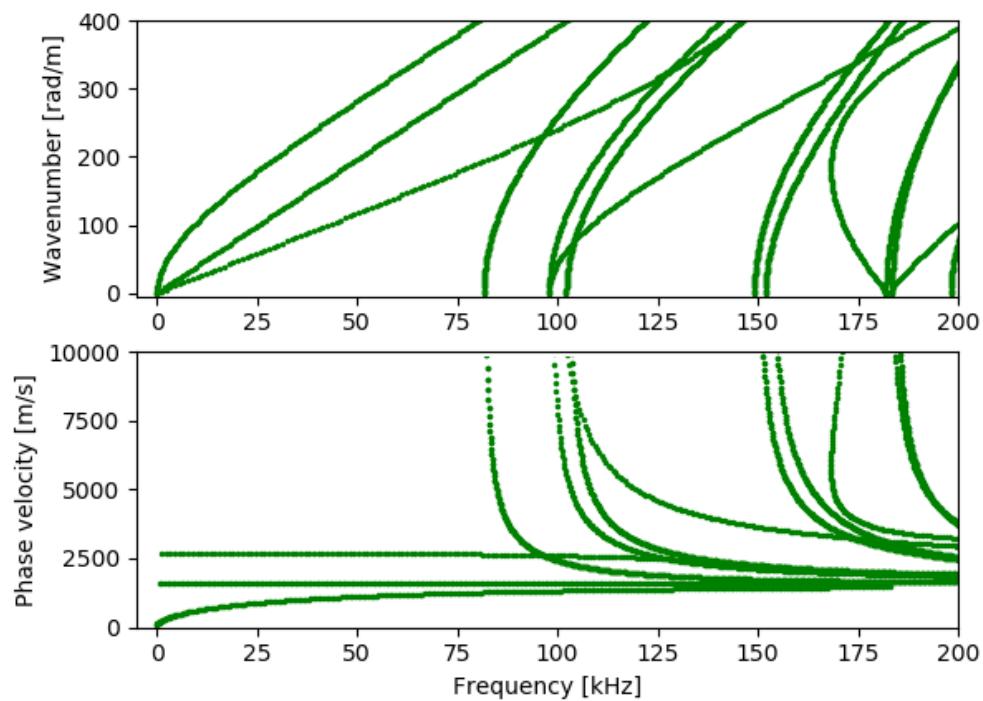
Rysunek 5.26 przedstawia krzywe dyspersji otrzymane z wykorzystaniem elementów czworościennych, a rysunek 5.27 krzywe otrzymane z wykorzystaniem elementów sześciennych.



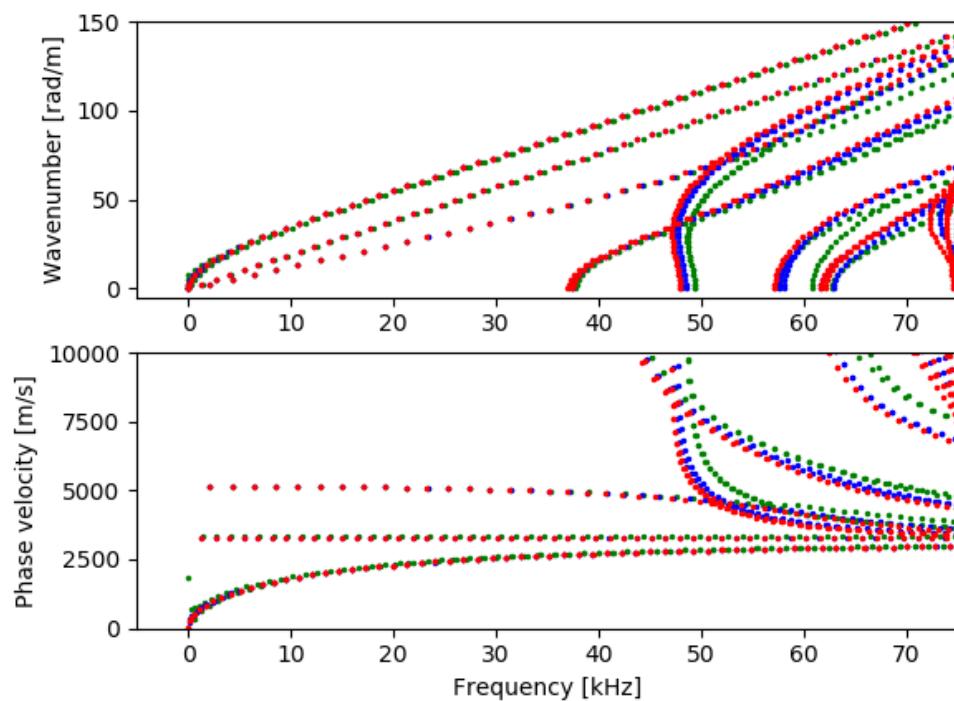
Rys. 5.26. Krzywe dyspersji otrzymane z wykorzystaniem elementów czworościennych dla stalowego pręta o promieniu 25mm. Pozostałe parametry symulacji to num_of_circles = 8 oraz num_of_points_at_c1 = 8

Wyniki różnią się do siebie częstotliwościami, w których pojawiają się kolejne postaci fali. Biorąc pod uwagę spodziewane wyniki, symulacja z wykorzystaniem elementów sześciennych wprowadza błąd na nieznanym etapie. Prędkości fali różnią się mocno od oczekiwanych.

W przypadku symulacji z elementami czworościennymi wyniki zbiegają do oczekiwanych wraz z zagęszczaniem siatki elementów. Przedstawia to rysunek 5.28. Widać na nim, że różnica pomiędzy wynikami z najrzadszą siatką, a wynikami ze średnią siatką, jest większa niż pomiędzy tą drugą, a siatką najgęstszą. Na tej podstawie można wywnioskować, że obliczenia przebiegają poprawnie.



Rys. 5.27. Krzywe dyspersji otrzymane z wykorzystaniem elementów sześciennych dla stalowego pręta o promieniu 25mm. Pozostałe parametry symulacji to firstCircle = 16, addNodes = 0, circles = 2

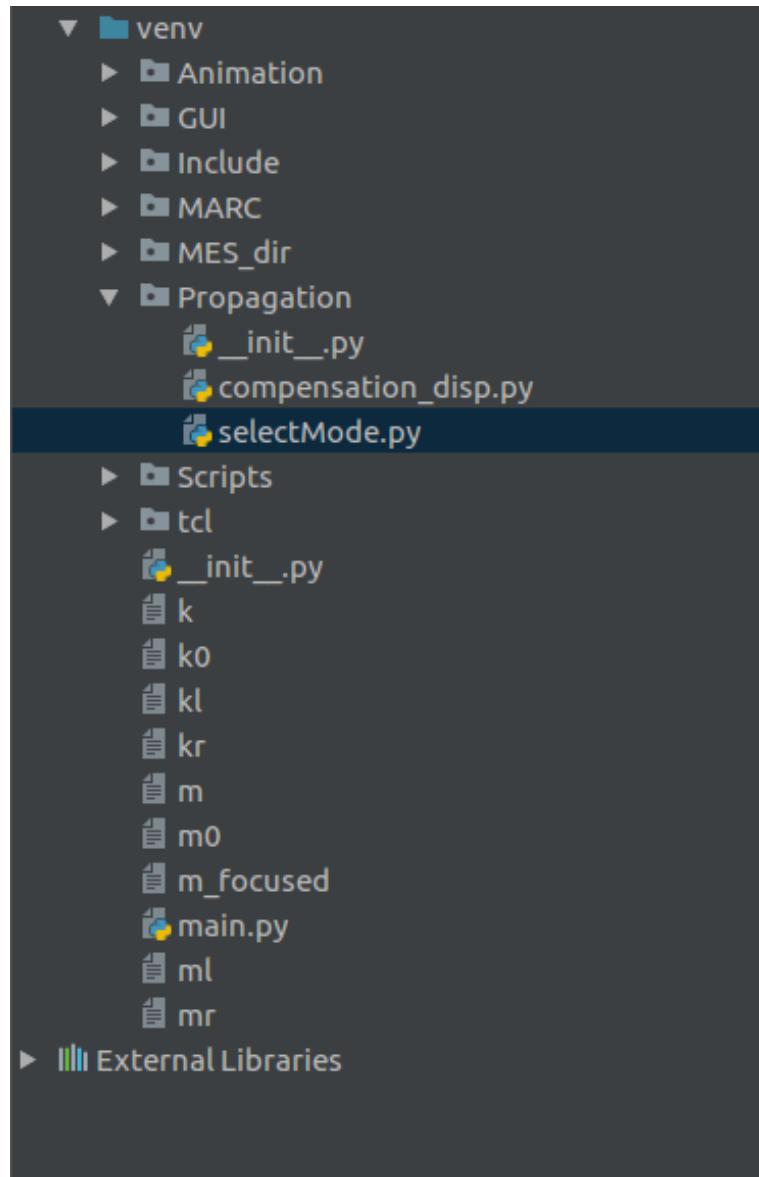


Rys. 5.28. Porównanie wyników z różną gęstością siatki. Kolor zielony - najrzadsza, niebieski - średnia, czerwony - najgęstsza

5.4. Segregacja krzywych dyspersji

KATARZYNA RUGIELŁO

Kod odpowiedzialny za segregację krzywych dyspersji wygenerowanych z solvera opisanego w poprzedniej sekcji znajduje się w pliku selectMode.py w katalogu o nazwie Propagation. Jego miejsce w drzewie projektu przedstawia rysunek 5.29. Jest to jedyna



Rys. 5.29. Lokalizacja pliku selectMode.py w drzewie projektu

część aplikacji napisana całkowicie obiektowo. Zastosowanie programowania obiektowego w przypadku tej części aplikacji znaczco ułatwiło późniejsze używanie stworzonego rozwiązania jak i uporządkowało sam kod. Plik selectMode został stworzony aby w sposób łatwy i szybki można było przeprowadzić agregację punktów do konkretnych

krzywych dyspersji a następnie w sposób łatwy i intuicyjny z nich korzystać. Samo generowanie obiektu przechowującego wszystkie krzywe dyspersji jest bardzo łatwe. Należy najpierw stworzyć obiekt typu `SelectMode`, jako argumenty podając ścieżki względne kolejno do pliku `kvect` oraz `omega`. Następnie należy wywołać metodę stworzonego właśnie obiektu o nazwie `selectMode()`. Od tego momentu nasz obiekt przechowuje posortowane krzywe dyspersji.

Podstawową jednostką są tu obiekty **klasy Point**. Klasa ta jak wszystkie omawiane w tej sekcji znajduje się w pliku `selectMode.py`. Klasa ta posiada jedynie konstruktor, przyjmujący dwa parametry: `w,k`. Obiekt ten reprezentuje pojedynczy punkt wygenerowany przez zaimplementowany solver MES. Wartość `w` jest wartością zespoloną i odpowiada wartości znajdującej się na osi x wykresu krzywej dyspersji, czyli częstości kątowej, natomiast `k` odpowiada wartości y tego wykresu i oznacza liczbę falową. Obydwa parametry przyjmują wartości domyślne wynoszące zero. **Klasa ta ma cztery pola:**

`w` - przechowujące wartość częstości w kHz

`wkat,real,part` - przechodzące część rzeczywistą podanej do konstruktora częstości

`wkat,complex` - przechowujące zespoloną częstość kątową podaną do konstruktora

`k` - przechowujące wartość liczby falowej podaną do konstruktora.

Klasa ta posiada tylko jedną metodę:

`printCoor()`, która nie przyjmuje żadnych argumentów. Jej funkcją jest wypisywanie w konsoli współrzędnych punktu w postaci: "w=[wartość w kHz], k=[wartość w rad/m]"

Kolejną, bardziej złożoną klasą, wykorzystywaną do przechowywania krzywych dyspersji jest **klasa Mode**. Jej głównym zadaniem jest przechowywanie zbioru punktów. Posiada konstruktor bezparametryowy oraz **następujące pola:**

`points` - tablica przechowująca zbiór punktów, domyślnie pusta

`minOmega` - minimalna przechowywana wartość ω wyrażona w kHz, póki tablica `points` jest pusta przyjmuje wartość `float('inf')`

`min_omega_kat` = minimalna przechowywana wartość ω wyrażona w $\frac{rad}{s}$, póki tablica `points` jest pusta przyjmuje wartość `float('inf')`

`allOmega` - tablica przechowująca wartości wszystkich częstości w postaci zespolonej, domyślnie pusta

`all_omega_khz` - tablica przechowująca wartości wszystkich częstości w kHz, domyślnie pusta

Klasa ta ma zaimplementowane następujące metody:

`addPoint(point)` funkcja ta przyjmuje jeden parametr. Jest nim obiekt klasy `Point`. Dodaje ona punkt do listy punktów (`self.points`), list `self.allOmega` i `self.all_omega_khz` oraz, jeśli to konieczne ustawia wartości pól `self.min_omega_kat` oraz `minOmega`.

delPoint(point) funkcja analogiczna znaczeniowo do *self.addPoint*, jako parametr przyjmuje obiekt klasy Point. Podany punkt jest wyszukiwany w tablicy *self.points* a następnie usuwany.

delDuplicates(pointlist) funkcja przyjmująca jako argument tablice obiektów typu Point. Jej zadaniem jest usunięcie z tablicy *self.points* wszystkich punktów z podanej listy.

quicksort(pocz, koniec) to funkcja przyjmująca dwa parametry, indeks początkowy i końcowy. Jej zadaniem jest posortowanie przechowywanych punktów rosnąco względem częstości kątowych. Argument *pocz* oznacza początkowy indeks tablicy, natomiast *koniec* indeks końcowy. Jest to funkcja działająca rekurencyjnie. Zasada działania tego sortowania opiera się na metodzie "dziel i zwyciężaj", której główną zasadą jest rozbijanie dużego problemu na skończoną ilość mniejszych podproblemów tak dugo, aż staną się one trywialne do rozwiązywania. W tym przypadku wybiera się tak zwany piwot wokół którego rozdziela się elementy do lewej i prawej tablicy. W lewej znajdują się elementy mniejsze w prawej większe. W ten sposób piwot znajduje się na swoim właściwym miejscu. Natomiast na lewej i prawej tablicy znów wywołuje się metoda quicksort. Postępuje się w ten sposób aż do uzyskania tablic jednoelementowych, które są już posortowane. Jest to bardzo dobry algorytm, zwłaszcza do sortowania dużej ilości danych.

findAngle(Ppoint) to funkcja przyjmująca jeden parametr - obiekt typu Point. Ma ona za zadanie wyznaczyć kąt pomiędzy dwoma wektorami. Pierwszy z nich stworzony jest przez dwa ostatnie punkty w tablicy *self.points* drugi natomiast to wektor mający początek w ostatniej punkcie tablicy *self.points* natomiast koniec w podanym w argumentie punkcie. Do obliczenia szukanego kąta wykorzystywana jest następująca zależność:

$$\cos \alpha = \frac{a \circ b}{|a| \cdot |b|} \quad (5.1)$$

$$\alpha = \arccos \frac{a \circ b}{|a| \cdot |b|} \quad (5.2)$$

Gdzie: $a \circ b$ - oznacza iloczyn skalarny wektorów a i b

$|a|, |b|$ - oznacza długość tych wektorów

Funkcja zwraca wartość tak obliczonego kąta.

findSmallestAngle(vPoint, dist) jest funkcją przyjmującą dwa argumenty. Pierwszy z nich to jednowymiarowa tablica obiektów klasy Point, natomiast drugi do odległość wyrażona w kHz, w jakiej mają znajdować się brane pod uwagę punkty. Funkcja przyjmuje wartość domyślną, która dobrana została heurystycznie. Funkcja bierze wszystkie punkty z podanej listy i wybiera spośród nich ten, który tworzy najmniejszy kąt z wektorem stworzonym z dwóch ostatnich punktów. Brane pod uwagę są tylko te punkty, które znajdują się odpowiednio blisko ostatniego punktu listy *self.points*. wartość drugiego

parametru jest ustawiona na domyślną. Jeśli w trakcie przeszukiwania danej tablicy punktów okazałoby się, że nie ma punktu w wyznaczonym otoczeniu, funkcja wywoływana jest rekurencyjnie z odpowiednio powiększonym zakresem otoczenia. Wartość zwracana to indeks punktu, najlepiej pasującego do już istniejącego zbioru punktów. Funkcja ta jest wykorzystywana bezpośrednio do segregacji krzywych dyspersji. Znalezienie punktu najbardziej odpowiadającego już istniejącej krzywej, a co za tym idzie tworzącego najmniejszy kąt i znajdującego się w odpowiednim sąsiedztwie jest kluczowym zadaniem pozwalającym na segregację punktów.

findPointWithK(k) przyjmuje jeden argument, k. Wyszukuje w tablicy *self.points* wszystkie punkty, których wartość pola k jest równa zadanej wartości. Zwraca tablicę tych punktów.

findPoint(points, omega) przyjmuje dwa argumenty: points i omega. Points to dwuelementowa tablica zawierająca obiekty klasy Point. Funkcja na podstawie podanych dwóch punktów interpoluje prostą i raz zwraca wartość tej prostej w punkcie omega.

findPointWithGivenK(points, k) przyjmuje dwa argumenty: points oraz k. działa analogicznie do funkcji *self.findPoint* lecz zamiast zwracać wartość funkcji na podstawie jej argumentu, zwraca argument na podstawie podanej wartości k, zwracana wartość wyrażona jest w kHz.

findPointWithGivenK, ad_s(points, k) ta funkcja również przyjmuje dwa argumenty, a jej działanie jest identyczne jak działanie funkcji *self.findPointWithGivenK*, z tą różnicą, iż zwracany wynik wyrażony jest w $\frac{rad}{s}$

findKWithGivenOmega_kHz(omega_kHz) przyjmuje jeden parametr, omega, będący częstotliwością wyrażoną w kHz. Oblicza wartość przechowywanej krzywej dyspersji dla podanej częstotliwości. Zwraca liczbę falową odpowiadającą podanej omedze.

Klasa Mode wykorzystywana jest zarówno do przechowywania zbioru punktów (na początek przechowuje cały zbiór wygenerowanych przez solver punktów), jak również na późniejszym etapie obiekty typu Mode przechowują pojedyncze krzywe dyspersji.

Kolejną klasą w stworzonej strukturze jest **klasa Data**. Posiada ona konstruktor bezparametrowy oraz **jedno pole**:

modeTable - jest to tablica przechowująca obiekty typu Mode, przechowujące punkty należące do kolejnych krzywych dyspersji.

Klasa ta posiada również jedną metodę:

addMode(mode) - przyjmuje obiekt typu mode i dodaje go do tablicy *self.modeTable*

Najbardziej nadzczną klasą jest **klasa SelectMode**. Posiada ona konstruktor przyjmujący trzy parametry:

kvect_{path} - ścieżka do wygenerowanego przez solver pliku kvect,

omega_{path} - ścieżka do wygenerowanego przez solver pliku omega,

rows - liczba kolumn w pliku omega, które chcemy wczytać, domyślnie ustawiona na cały plik czyli 426 kolumn.

Klasa ta posiada 5 pól. Są to:

eigpath - przechowująca podaną ścieżkę do pliku kvect

omegopath - przechowująca podaną ścieżkę do pliku omega

rows - przechowująca liczbę kolumn do odczytu

AllModes - obiekt typu Data. Na początku nie przechowujący żadnych danych

k_v - wektor wartości liczby falowej. Domyślnie pusty

W klasie tej zostały zaimplementowane trzy metody:

getMode(number), która jako argument przyjmuje liczbę całkowitą, natomiast zwraca obiekt typu Mode. Jej zadaniem jest z obiektu przechowującego wszystkie krzywe dyspersji, wybrać tę o podanym numerze i zwrócić ją w postaci obiektu klasy Mode.

plotmodes(num_of_modes), która jako argument przyjmuje liczbę całkowitą, określającą ile pierwszych krzywych dyspersji chcemy wyświetlić. Ta funkcja niczego nie zwraca, jedynie generuje wykres żądanych krzywych dyspersji.

selectMode() jest funkcją nie przyjmującą żadnych parametrów. Na podstawie podanych do konstruktora danych wczytuje wektor liczb falowych z pliku kvect. następnie tworzy obiekt typu Mode, *AllPoints*, do którego następnie wczytuje wszystkie punkty wygenerowane przez solver. Następnie tworzone są dwa obiekty klasy Mode: *MinKTable* oraz *MinKTable2*. Pierwsza z nich przechowuje punkty o wartości k równej najmniejszej wartości występującej w wektorze liczb falowych, natomiast druga przechowuje punkty, których wartość liczby falowej jest drugą najmniejszą wartością z wektora liczb falowych.

Te wybrane punkty, uszeregowane względem omega stanowią zestaw dwóch pierwszych punktów kolejnych krzywych dyspersji. Następnym krokiem jest usunięcie ich z wektora *self.AllPoints*. Ostatnim etapem segregacji jest wybieranie kolejnych punktów z wektora *self.AllPoints* o aktualnie najniższej wartości liczby falowej, oraz przydzielanie ich do odpowiednich modów, używając funkcji *findSmallestAngle*. Wszystkie przydzielone punkty są usuwane z wektora *self.AllPoints*. Procedura jest powtarzana do momentu aż wektor ten stanie się pusty.

Listing przykładowego kodu, agregującego punkty do krzywych dyspersji, oraz rysowania wybranej krzywej:

```
Mody = SelectedMode('../eig/kvect','../eig/omega')
```

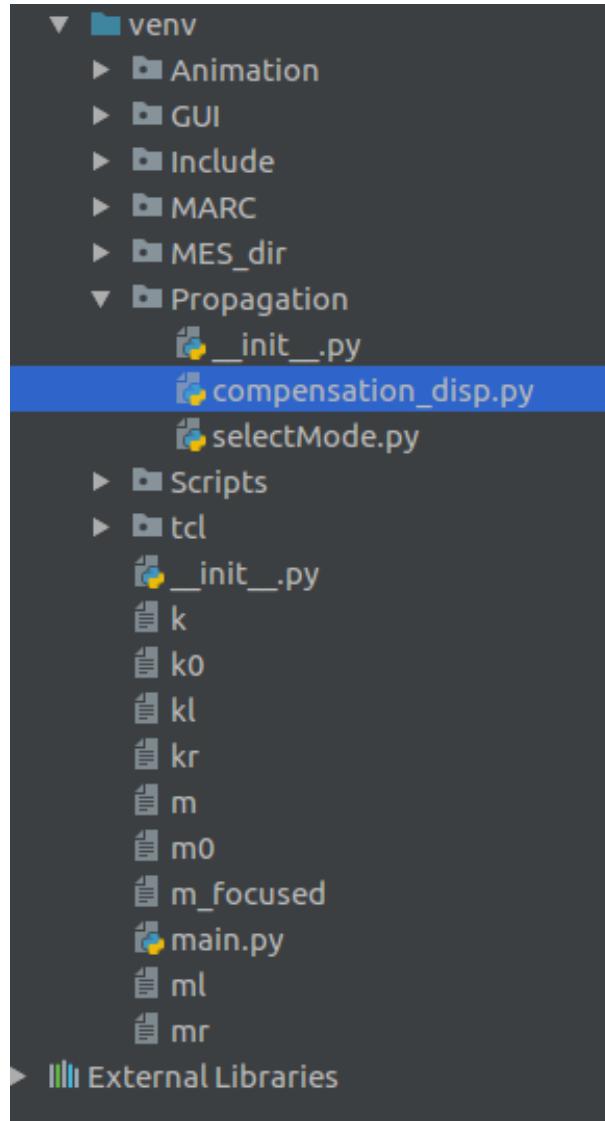
```
Mody.selectMode()
```

```
Mody.plotmodes(50)
```

5.5. Metody kompensacji dyspersji - Katarzyna Rugełło

KATARZYNA RUGEŁŁO

Wszystkie zaimplementowane metody związane z kompensacją dyspersji znajdują się w pliku *compensation_disp.py* w folderze Propagation. Lokalizację pliku przedstawia rysunek 5.30



Rys. 5.30. Lokalizacja pliku *compensation_disp.py* w drzewie projektu

Trzy podstawowe metody znajdujące się w tym pliku odpowiadają trzem opisanym wcześniej metodom kompensacji. Są to:

linear_mapping_compensation(signal, number_of_modes, disp_curves) funkcja kompensująca dyspersję przy pomocy liniowego przybliżenia szeregu Taylora, opisana w rozdziale 4.4. Funkcja przyjmuje trzy argumenty. Pierwszy z nich, *signal*, jest dwuelementową tablicą reprezentującą sygnał, który ma zostać skompensowany.

signal[0] to wektor czasu tego sygnału, natomiast *signal[1]* to wektor wartości sygnału w kolejnych chwilach czasowych. Konwencja przechowywania sygnału w postaci dwuelementowej tablicy jest zachowana w całym pliku. Kolejnym przyjmowanym argumentem, *number_of_modes*, jest to liczba całkowita, będąca indeksem krzywej dyspersji, która powinna zostać użyta do kompensacji dyspersji. Ostatnim argumentem, *disp_curves*, jest obiekt klasy *SelectedMode* przechowującym zaagregowane krzywe dyspersji. Wartością zwracaną jest skompensowany sygnał w postaci dwuelementowej tablicy [*wektor_czasu*, *wektor_wartoci*].

time_reverse_compensation(signal, distance, numbers_of_modes, disp_curves) funkcja generująca sygnał, którego zadaniem jest skompensowanie się na zadanej odległości przy założeniu propagacji wybranych trybów fali. Metoda ta opisana została w rozdziale 4.3. Funkcja przyjmuje cztery parametry. Pierwszy z nich, *signal* jest sygnałem, który użytkownik chce otrzymać po propagacji. Jego format jest analogiczny jak w poprzednio omawianej funkcji. Kolejny argumentem jest *distance*. Jest to liczba, wyrażająca w metrach odległość po jakiej sygnał powinien się skompensować. Następny argument to *numbers_of_modes*. Jest to jednowymiarowa tablica liczb całkowitych. Zawiera ona indeksy krzywych dyspersji, tych postaci fali, które mają propagować w przecie. Funkcja zwraca sygnał, który przepropagowany przez pręt, o zadaną odległość sam się skompensuje. Ostatnim argumentem jest obiekt przechowujący krzywe dyspersji.

mapping_from_time_to_distance(dispersion, dispersion_curves, propagated_modes, need_to_pad = False) jest funkcją kompensującą rozproszony sygnał przy pomocy mapowania funkcji w dziedzinie czasu na dziedzinę odległości, opisaną w rozdziale 4.5. Przyjmuje cztery argumenty w tym ostatni jest argumentem o wartości domyślnej równej False. Pierwszy z nich *dispersion* jest sygnałem, który należy skompensować. Drugi argument to obiekt, przechowujący krzywe dyspersji. Następnie jednowymiarowa tablica liczb całkowitych, przekazująca do funkcji indeksy postaci fali, które propagowały wbadanym obiekcie. Ostatni parametr jest opcjonalny. Domyślnie przyjmuje on wartość False. W aplikacji funkcja generująca przepropagowany sygnał zapewnia jego wydłużenie poprzez dodanie zer. Z tego powodu domyślnie argument ten jest ustawiony na wartość False. Jeśli jednak wyniki z funkcji nie dają dobrych rezultatów, powstały sygnał nie jest skompensowany. Należy ustawić te flagę na wartość True. Funkcja zwraca dwuelementową tablicę w postaci [*wektor_odlegoci*, *wartoci_funkcji*]

Pozostałe funkcje zawarte w pliku są używane przez zaprezentowane już funkcje. Są to:

find_accurate_len(actual_len, factor = 8) - jest to funkcja, która znajduje porządaną długość wektora. Jak zostało to napisane w [21], najlepiej aby sygnał wyłożony zerami po wydłużeniu miał liczbę próbek równą jakiejś potęze dwójki. Pierwszy argument to obecna długość sygnału, natomiast drugi parametr, przyjmujący domyślnie

wartość 8 mówi ile razy użytkownik chce wydłużyć sygnał. Funkcja zwraca długość sygnału o długości, która jest potęgą dwójki nie mniejszą od pierwotnej długości pomnożonej przez współczynnik *factor*.

pad_timetraces_zeroes(time_vector, signal_vector, multi = 8) - funkcja, której zadaniem jest uzupełnić podany sygnał odpowiednią ilością zer. Pierwszym argumentem jest wektor czasu sygnału, drugim wartością sygnału w czasie. Ostatnim przekazywanym parametrem jest liczba całkowita, mówiąca ile razy chcemy wydłużyć podany sygnał. Domyślnie przyjmuje wartość 8. Wartość parametru *multi* jest przekazywana do funkcji *find_accurate_len* jako argument *factor*. Funkcja zwraca sygnał wyłożony w formacie [*wyduony_wektor_czasu, wartoci_wyduonego_sygnau*]

calculate_n(k_Nyquista, delta_k, factor = 1.1) - funkcja obliczająca wartość n zgodnie ze wzorem 4.35. Przyjmowane przez nią argumenty to liczba falowa Nyquista, krok liczby falowej oraz współczynnik, który domyślnie przyjmuje wartość 1.1. Ponieważ nierówność 4.35 jest ostra, współczynnik ten musi być większy od 1. W przypadku podania do funkcji wartości mniejszej niż 1 wartość ta zostanie automatycznie ustawiona na 1.1. Wartością zwracaną jest wartość liczby n spełniającą wymaganą nierówność

calculate_k_nyquist(dispercion_curves, dt, factor = 1.1) - funkcja obliczająca wartość liczby falowej Nyquista. Korzystający z nierówności 4.34. Jako argumenty przyjmuje obiekt zawierający krzywe dysperji, krok czasowy oraz współczynnik, który domyślnie przyjmuje wartość 1.1. Tak jak w przypadku poprzedniej funkcji, w przypadku podania wartości mniejszej od 1 zostanie ona ustawiona na wartość domyślną, tak samo jak w przypadku nie podania wartości tego argumentu. Wartością zwracaną, jest wyliczona wartość liczby falowej Nyquista.

calculate_delta_k(max_v_gr, signal_duration, factor = 0.9) - funkcja wyliczająca potrzebny krok liczby falowej zgodnie z zależnością 4.33. Jako argumenty przyjmuje maksymalną prędkość grupową, długość trwania sygnału oraz współczynnik. Ostatni argument ma wartość domyślną równą 0.9. Wynosi ona tyle zarówno w przypadku, gdy argument nie zostanie podany jawnie do funkcji jak i w przypadku w którym podana wartość będzie większa lub równa 1. Wartością zwracaną jest obliczona wartość Δk

calculate_delta_x(k_Nyquista) - funkcja wyliczająca wartość Δx . Jako parametr przyjmuje liczbę falową Nyquista natomiast zwraca obliczoną wartość Δx

find_max_k(mode, k_vect, max_omega_kHz) - funkcja pobiera jako argumenty, obiekt klasy Mode, reprezentujący wybraną krzywą dyspersji, wektor liczb falowych oraz wartość częstotliwości. Na ich podstawie oblicza wartość krzywej dyspersji i ją zwraca. Funkcja ma na celu znalezienie wartości liczby falowej dla największej wartości częstotliwości występującej w sygnale wzbudzającym. W przypadku, w którym podana częstotliwość nie wzbudzi rzadanej postaci. Funkcja zwróci wartość -1

find_omega_in_dispercion_curves(mode, temp_k, k_vect) - metoda przyjmująca trzy argumenty: obiekt klasy Mode, reprezentujący krzywą dyspersji,

wartość na krzywej dyspersji dla której chcemy odnaleźć odpowiadającą częstotliwość oraz wektor liczb falowych. Na podstawie podanej wartości liczby falowej obliczana jest poszukiwana wartość częstotliwości. Wartość zwracana jest poszukiwaną wartością wyrażoną w kHz.

find_omega_in_dispercion_curves,ad_s(mode,temp_k,k_vect) - funkcja analogiczna do *find_omega_in_dispercion_curves*, jedyna różnica to wartość zwracana, która jest wyrażona w $\frac{rad}{s}$

find_value_by_omega_in_G_w(G_w,freq_sampling_kHz,omega) - funkcja, której zadaniem jest znalezienie wartości w widmie sygnału $G(w)$ na podstawie podanej w kHz częstotliwości. Pierwszym przyjmowanym argumentem, jest widmo sygnału, z którego użytkownik chce wyciągnąć wartość. Przyjmuje on postać jednowymiarowej tablicy zawierającej kolejne wartości widma. Drugim parametrem jest wektor częstotliwości, wyrażonych w kHz którym odpowiadają kolejne elementy G_w . Ostatnim parametrem funkcji jest częstotliwość dla której użytkownik chce poznać wartość widna, wyrażoną w kHz. Metoda jako wynik zwraca wartość odczytaną z widma sygnału.

calculate_group_velocity(mode,k_sampling_rad_m,ind,k_vect) - funkcja oblicza prędkość grupową w funkcji częstotliwości wybranej postaci fali w zadanym punkcie. Jako argumenty przyjmuje kolejno: obiekt klasy Mode przechowujący wybraną krzywą dyspersji. Wektor przechowujący spróbowane liczby falowe, indeks punktu w którym należy obliczyć prędkość grupową oraz wektor liczb falowych odpowiadający żądanemu trybowi dali. Wartością zwracaną jest wartość prędkości grupowej wybranej postaci fali w wybranej częstotliwości.

calculate_mean_mode(dispercion_curves,numbers_of_propagated_modes) - funkcja której zadaniem jest wyznaczenie średniej krzywej dyspersji z podanych krzywych. Jako argumenty przyjmuje kolejno: obiekt klasy SelectMode przechowujący wszystkie krzywe dyspersji oraz jednowymiarową tablicę liczb całkowitych stanowiących indeksy krzywych, które użytkownik chce uśrednić. Wartością zwracaną jest obiekt klasy Mode będący średnią krzywą dyspersji ze wszystkich wymienionych w *number_of_propagated_modes*

time_reverse(signal) - funkcja przyjmująca jako argument sygnał w postaci *[wektor_czasu,wektor_wartoci]* zwracający ten sam sygnał odwrócony w czasie w postaci analogicznej dwuelementowej tablicy.

Funkcją ostatnią w omawianym pliku jest funkcja służąca do symulacji propagacji fali prowadzonej w precie opisany przez wygenerowane krzywe dyspersji. jest to funkcja *wave_length_propagation(signal,numbers_of_modes,disp_curves,distance_m,F_PADZEROES,mult = 8)*. Przyjmuje ona 6 argumentów w tym ostatni przyjmuje wartość domyślną równą 8. Pierwszy z nich to sygnał w postaci dwuelementowej tablicy *[wektor_czas,wektor_wartoci]*. Kolejnym argumentem jest jednowymiarowa tablica zawierająca liczby całkowite, stanowiące indeksy postaci fali, które mają zostać

zasymulowane w propagacji. Następnym argumentem jest obiekt klasy `SelectMode`, przechowujący wszystkie krzywe dyspersji. Kolejnym argumentem jest odległość o jaką użytkownik chce zasymulować propagację, podana w metrach. Argument `F_PADZEROS` jest flagą przyjmującą wartości typu `True` lub `False`, informującą o tym czy użytkownik chce wydłużyć wprowadzany sygnał poprzez wyłożenie go zerami. Ostatni argument przekazuje informację ile razy należy wydłużyć sygnał. Wartością zwracaną jest przepropagowany sygnał w postaci dwuelementowej tablicy.

Poniższy listing kodu pokazuje przykład użycia opisanych funkcji do kompensacji dyspersji.

```

KrzyweDyspersji = selectMode.SelectedMode('../eig/kvect', '../eig/omega')
KrzyweDyspersji.selectMode()
dist = 3#w metrach
signal_array, time_x_freq = Anim_dyspersji.get_chirp()
signal = wavelength_propagation([time_x_freq[0], signal_array[3]], [0, 1, 2, 3],
KrzyweDyspersji, dist, True, 100)
signal_after_compensation = mapping_from_time_to_distance(signal,
KrzyweDyspersji, [0, 1, 2, 3])
Taylor_compensation = linear_mapping_compensation(signal, [0, 1, 2, 3],
KrzyweDyspersji)
inversed = time_reverse_compensation([time_x_freq[0], signal_array[3]])

```

5.6. Graficzny interfejs użytkownika - Katarzyna Rugełło

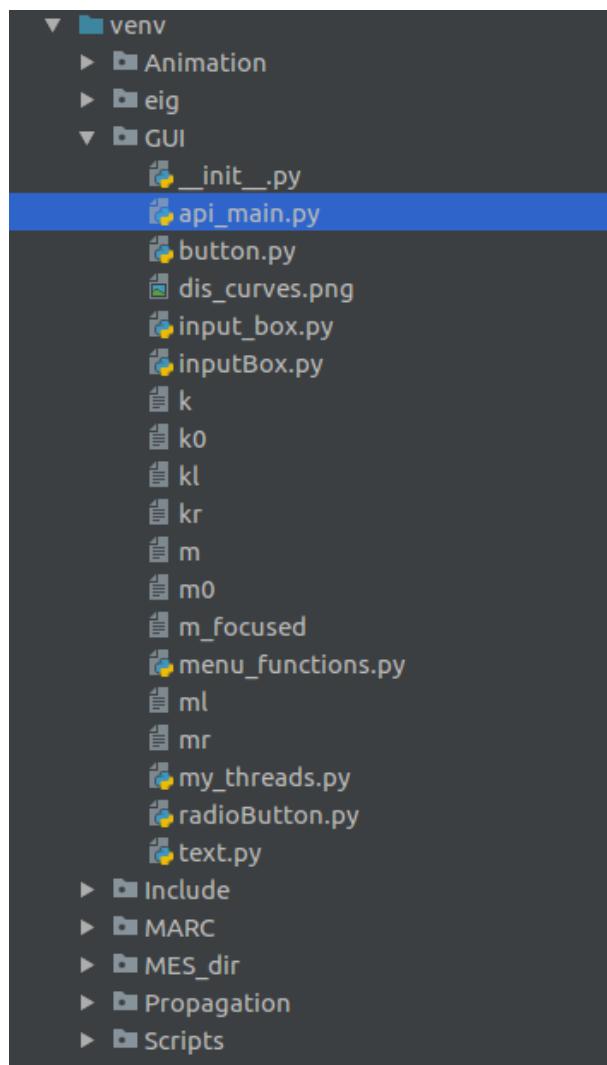
KATARZYNA RUGIELŁO

5.6.1. Opis dostępnych funkcji

W ramach niniejszej pracy zaimplementowany został graficzny interfejs użytkownika. Umożliwia on obsługę większości opisanych we wcześniejszych sekcjach funkcji. Pozwala na korzystanie z opracowanego oprogramowania bez znajomości języka Python. Uruchamianie interfejsu odbywa się z poziomu IDE, w tym przypadku z Pycharma. W drzewie projektu należy odnaleźć katalog o nazwie `GUI`, następnie plik `api_main.py`. Rysunek 5.31

Po uruchomieniu programu `api_main.py` wyświetlany zostaje graficzny interfejs. Rysunek 5.32 przedstawia wygląd interfejsu zaraz po uruchomieniu.

Po pierwsze użytkownik proszony jest o wybranie, czy chce korzystać z ostatnio wygenerowanych danych czy chce wygenerować nowe dane do symulacji. Nowe dane można



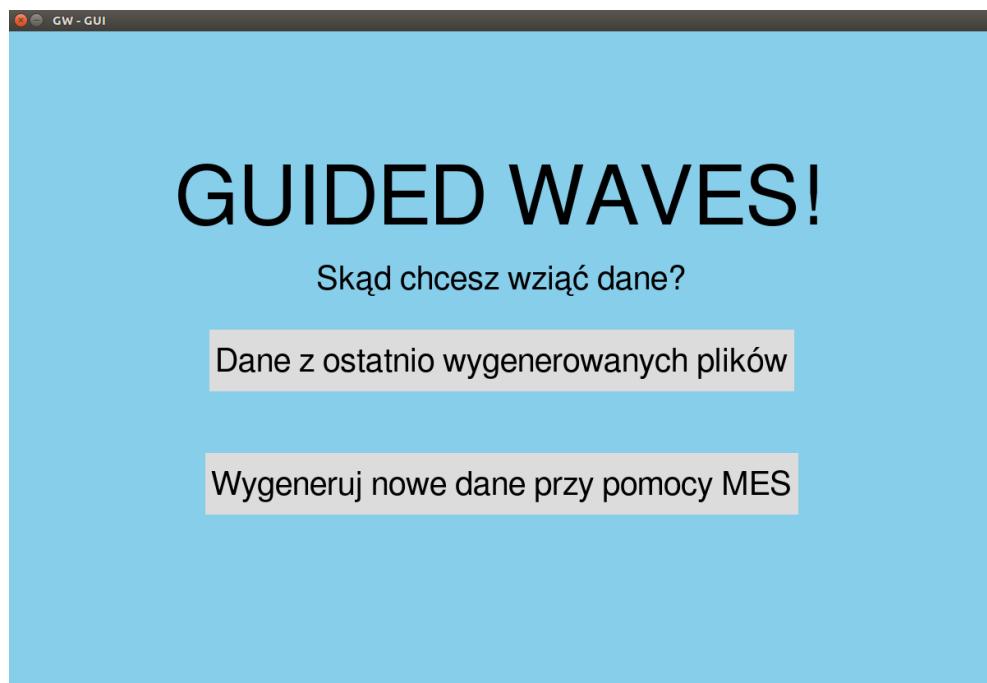
Rys. 5.31. Położenie *api_main.py* w drzewie projektu

również wygenerować później w dowolnym momencie. Po wybraniu opcji "Dane z ostatnio wygenerowanych plików" wyświetlany zostaje komunikat informujący o ładowaniu danych do programu (Rysunek 5.33)

W tym momencie, wygenerowane przez solver dane są wczytywane do programu. A krzywe dyspersji agregowane do odpowiednich obiektów tak jak zostało to opisane w poprzednim podrzędziale. Kiedy ładowanie zostanie zakończone wyświetlany zostaje napis "Krzywe dyspersji" oraz wyświetlane są wczytane krzywe. Przedstawia to rysunek 5.34 5.35

Aby przejść dalej należy zamknąć okienko przedstawiające krzywe dyspersji. Następnie wyświetlane jest główne menu interfejsu. Przedstawia je rysunek 5.36

W tej części do wyboru jest pięć opcji. Pierwsza z nich wyświetla okno z krzywymi dyspersji z rysunku 5.35. Opcja "Wygeneruj nowe dane" przenosi użytkownika do okna z rysunku 5.37. Okno to zostanie również wyświetcone po wybraniu z okna przedstawionego na rysunku 5.32 opcji "Wygeneruj nowe dane"

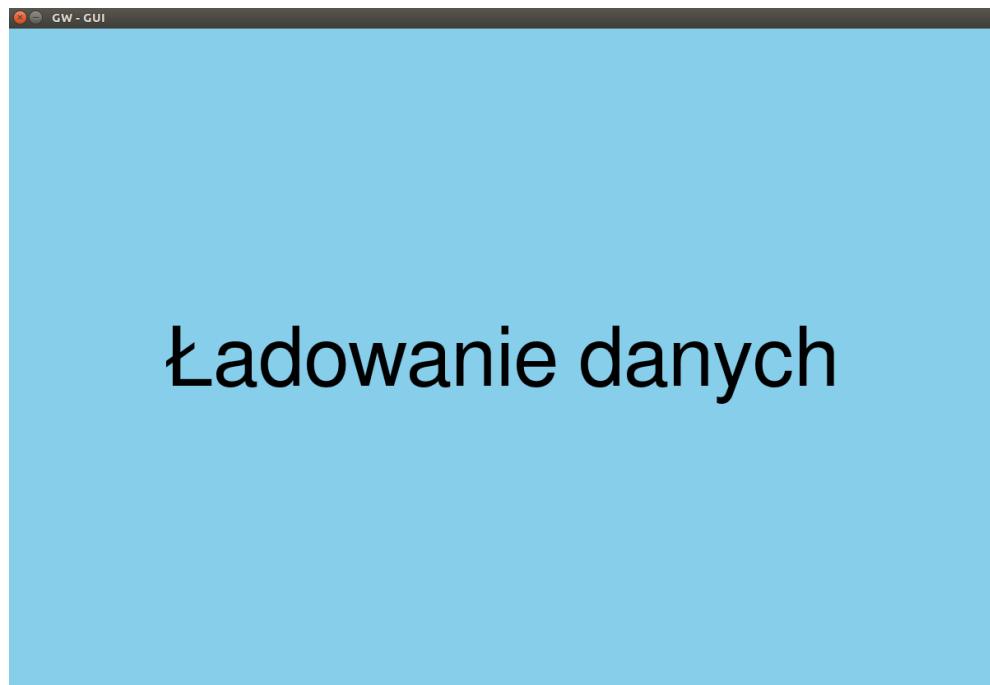


Rys. 5.32. Wybór źródła danych do symulacji

W tym miejscu można wybrać rodzaj siatki, jaką użytkownik chce wygenerować. Ustawić wszystkie parametry oraz zaznaczyć czy chce się wyświetlać benerowaną siatkę zarówno na płaszczyźnie, jak i w postaci pręta czy też elementów skończonych. Po wybraniu interesujących parametrów oraz zatwierdzeniu wyboru poprzez naciśnięcie przycisku "Oblicz" pojawia się ekran informujący, że obliczenia trwają. Przedstawia to rysunek 5.38. Ważną kwestią podczas wpisywania parametrów jest podawanie liczb tylko przy pomocy klawiszy znajdujących się nad klawiaturą. Klawiatura numeryczna nie jest obsługiwana, a jej użycie może spowodować błąd oraz zakończenie działania programu. Znacznik szary oznacza, iż dana opcja jest nieaktywna, natomiast znacznik zielony oznacza aktywną opcję.

Gdy siatka zostanie już wygenerowana program wraca do głównego menu z rysunku 5.36. Kolejne trzy dostępne opcje odpowiadają opisany już metodom kompensacji dyspersji. W całym interfejsie sygnałem propagującym oraz będącym kompensowanym jest sygnał linear chirp pomnożonym przez okno Hanninga z rysunku 4.8. Po wybraniu opcji "Symulacja kompensacji metodą mapowania na dziedzinę odległości" pojawia się okno z rysunku 5.39.

W tym momencie użytkownik wybiera sygnał, jaki chce wygenerować do symulacji. Zostanie on następnie skompensowany wybraną metodą. Gdy wygenerowany sygnał zostanie skompensowany jest on wyświetlany. Po zamknięciu okienka z wyświetlonymi wynikami, program wraca do głównego menu. Po wybraniu opcji "Symulacja kompensacji metodą mapowania liniowego" program realizuje algorytm analogiczny do właśnie opisanego. Jedyna różnica to metoda jaką kompensowany jest sygnał. Wybranie opcji



Rys. 5.33. Ekran ładowania danych

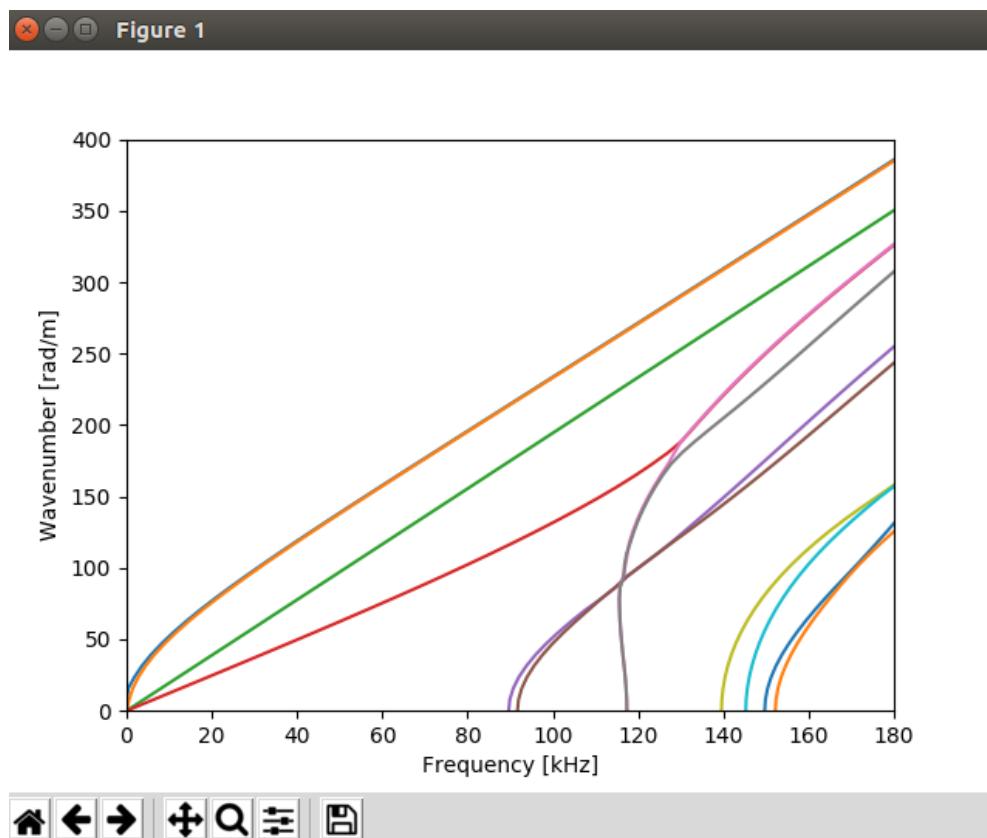
Śymulacja sygnału kompensującego się na zadanej odległości otwiera okno przedstawione na rysunku 5.40

Po wpisaniu odpowiednich parametrów oraz wykonaniu stosownych obliczeń wyświetlany jest wygenerowany sygnał. Po zamknięciu okna wyświetlającego wyniki pojawia się okno pozwalające na ustawienie danych propagacji, tak aby można było zaobserwować zachowanie wygenerowanego sygnału po propagacji (rys. 5.41)

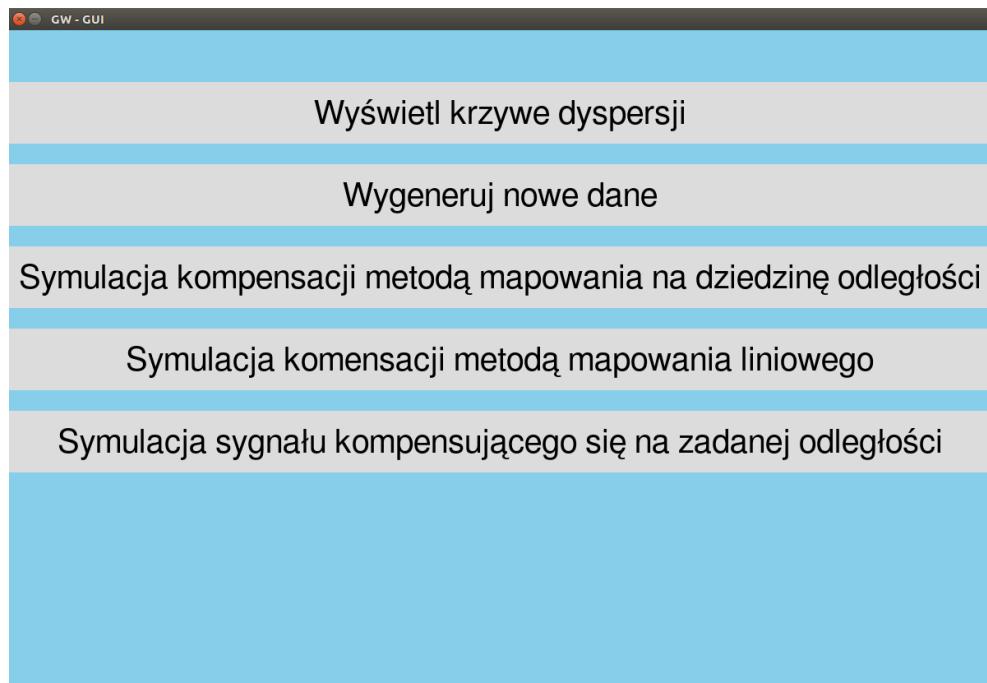
Po skończonych obliczeniach wyświetlany jest otrzymany sygnał.



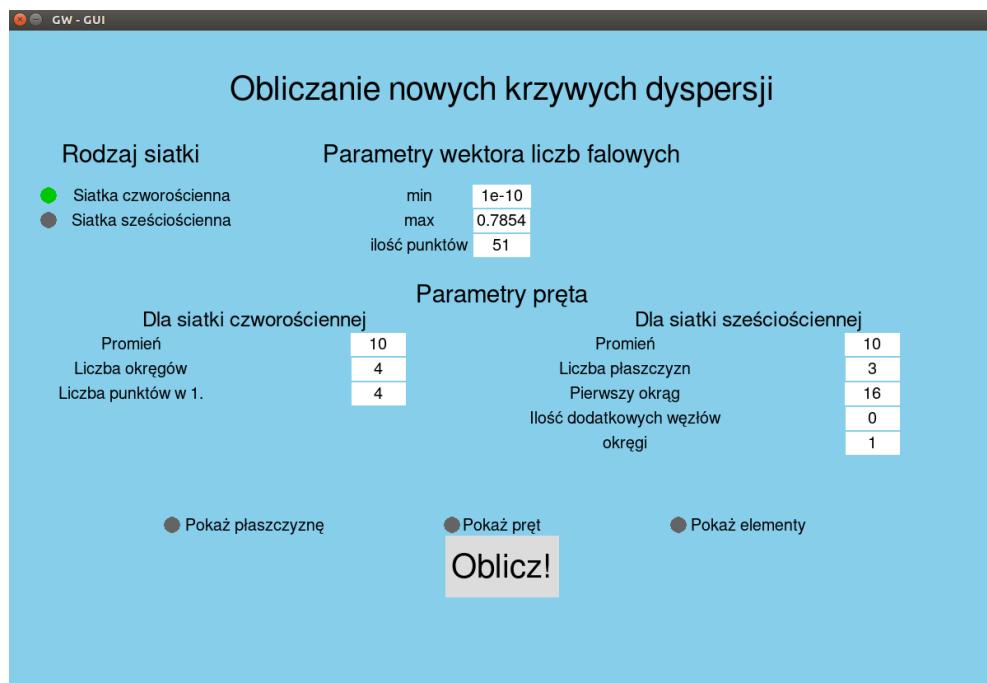
Rys. 5.34. Krzywe dyspersji



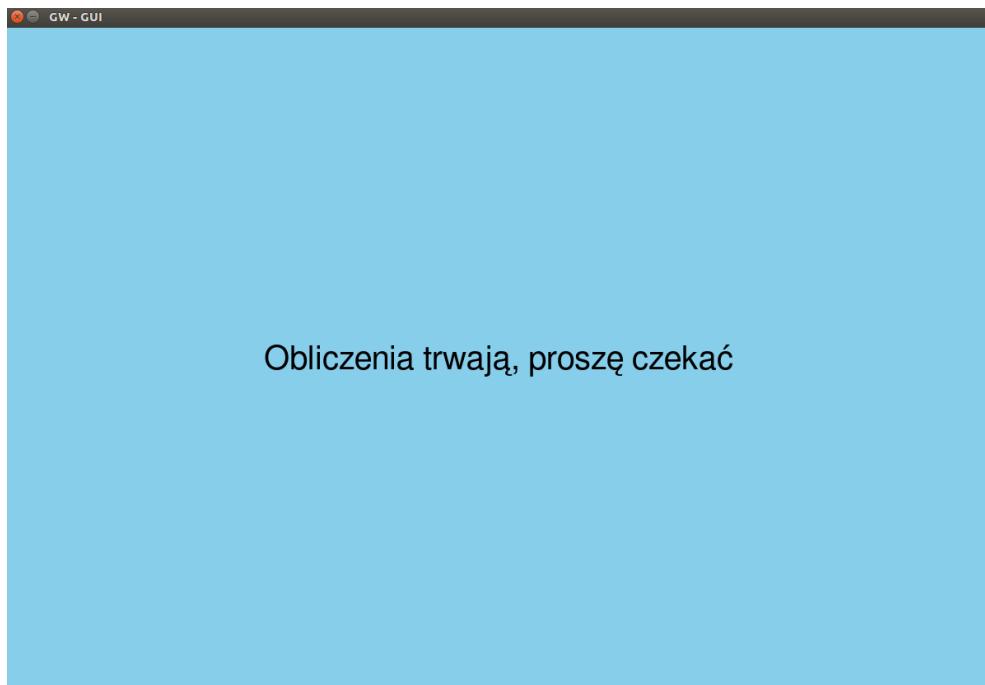
Rys. 5.35. Krzywe dyspersji, pręta aluminiowego o średnicy 10mm



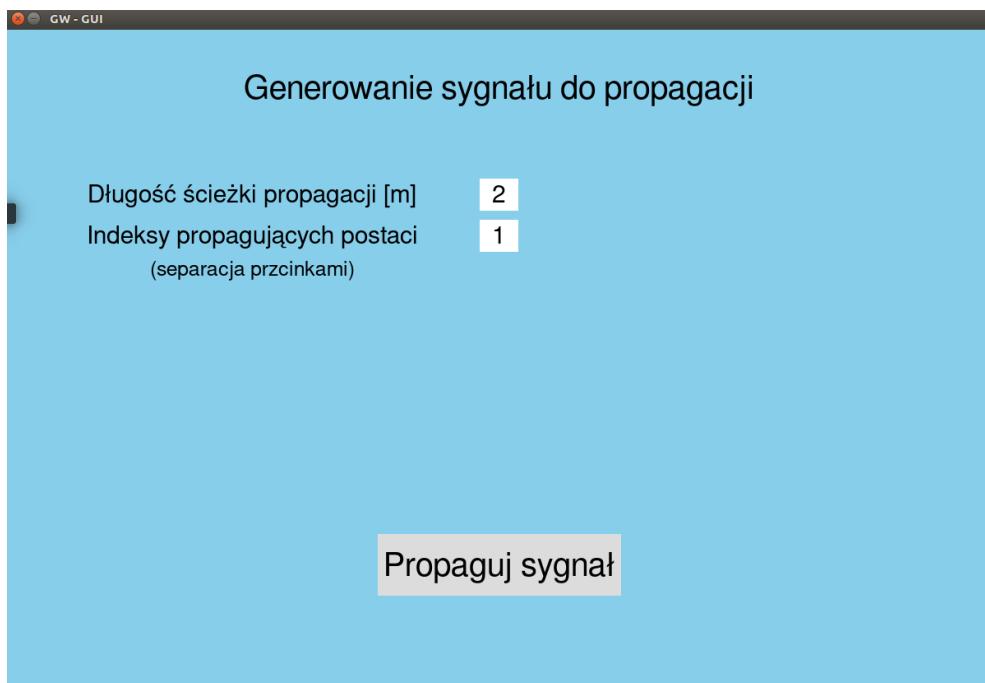
Rys. 5.36. Główne menu programu



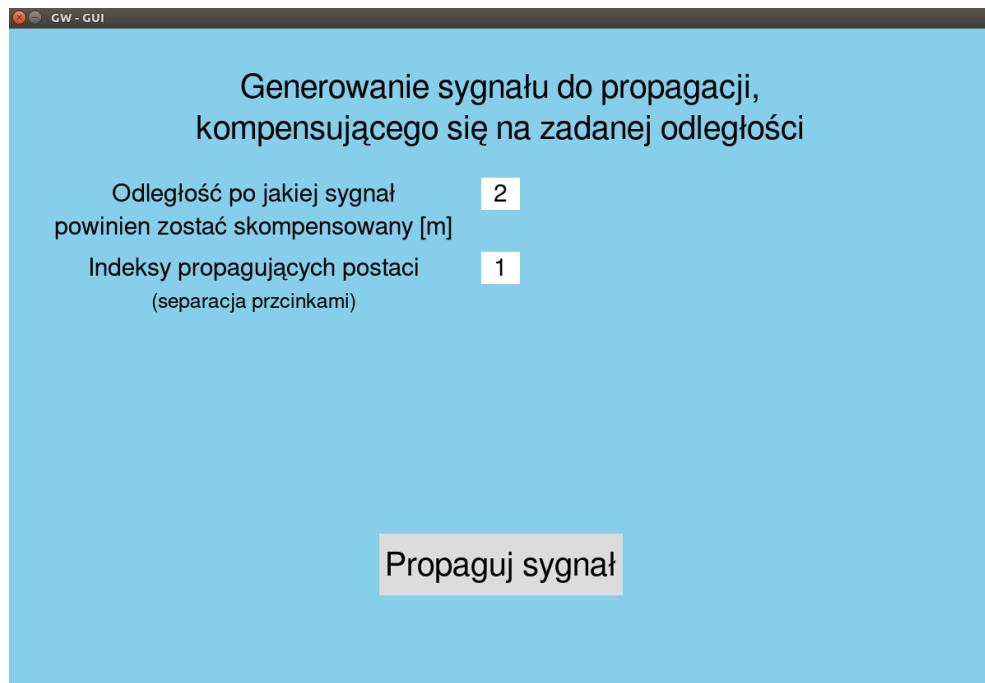
Rys. 5.37. Ekran ustawień siatki



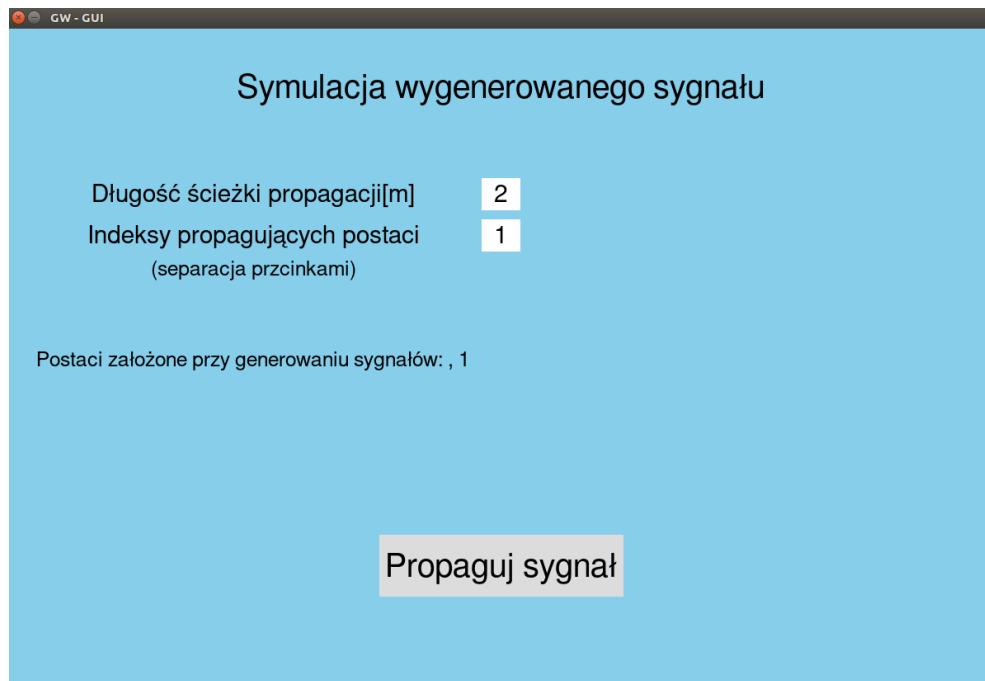
Rys. 5.38. okno programu informujące o trwających obliczeniach



Rys. 5.39. okno programu informujące o trwających obliczeniach



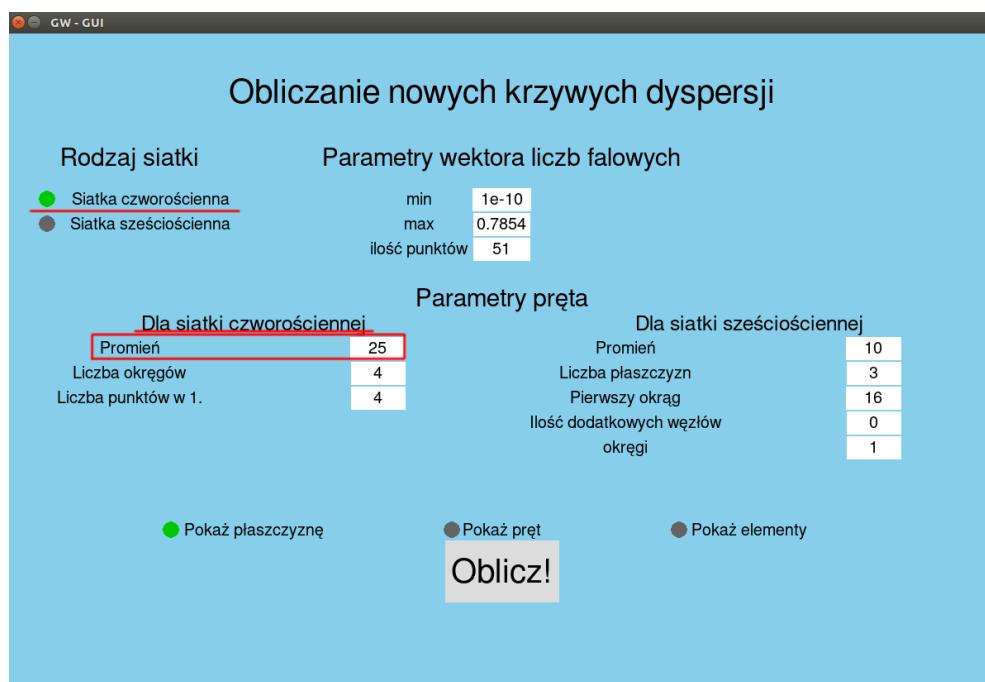
Rys. 5.40. okno programu informujące o trwających obliczeniach



Rys. 5.41. okno programu informujące o trwających obliczeniach

5.6.2. Przykład zastosowania do wygenerowania danych dla zadanego pręta

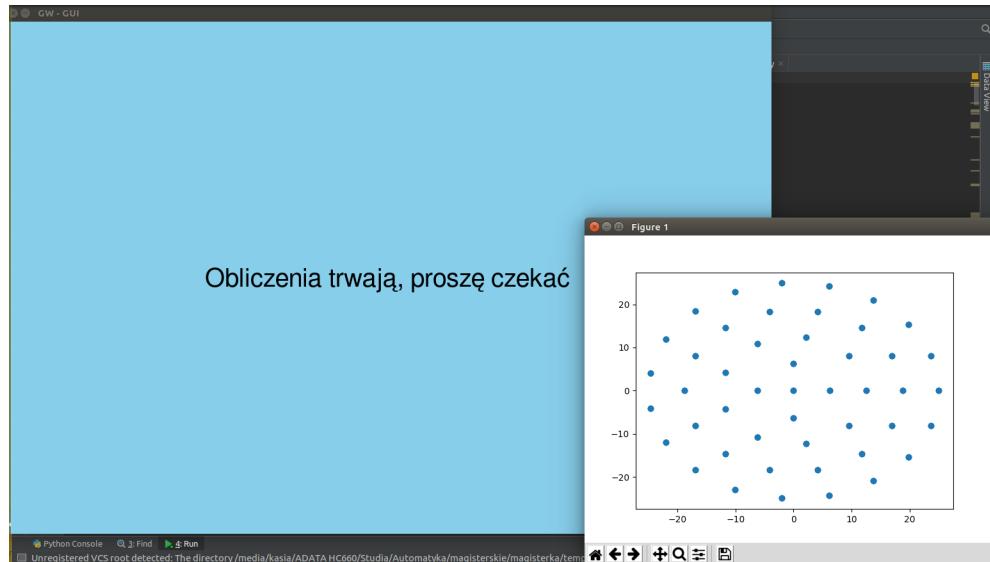
W niniejszej sekcji został zaprezentowany proces generowania danych dla pręta ze stali czarnej o średnicy 25 mm oraz długości 2m. Przed uruchomieniem programu należy w pliku *config.py*, znajdującym się w folderze *MES_dir* ustawić dane materiału pręta, dla którego chcemy przeprowadzić symulację. Konieczne jest podanie wartości modułu Younga, gęstości oraz współczynnika Poissona. Należy te wartości przypisać do odpowiednich zmiennych w pliku. Są to kolejno: *young_mod*, *density*, *poisson_coef*. Gdy odpowiednie wartości zostaną ustawione należy uruchomić program *api_main.py* w menu przedstawionym na rysunku 5.32 należy nacisnąć przycisk „Wygeneruj nowe dane przy pomocy MES”. Wyświetlone zostanie okno ustawień siatki MES generowanej przez program. W tym miejscu ustawiana zostaje średnica pręta, dla którego ma zostać wygenerowany model. Przyjmowaną jednostką jest milimetr, dlatego w polu średnica należy wpisać wartość 25, ilustruje to rysunek 5.42.



Rys. 5.42. okno programu do ustawień siatki MES

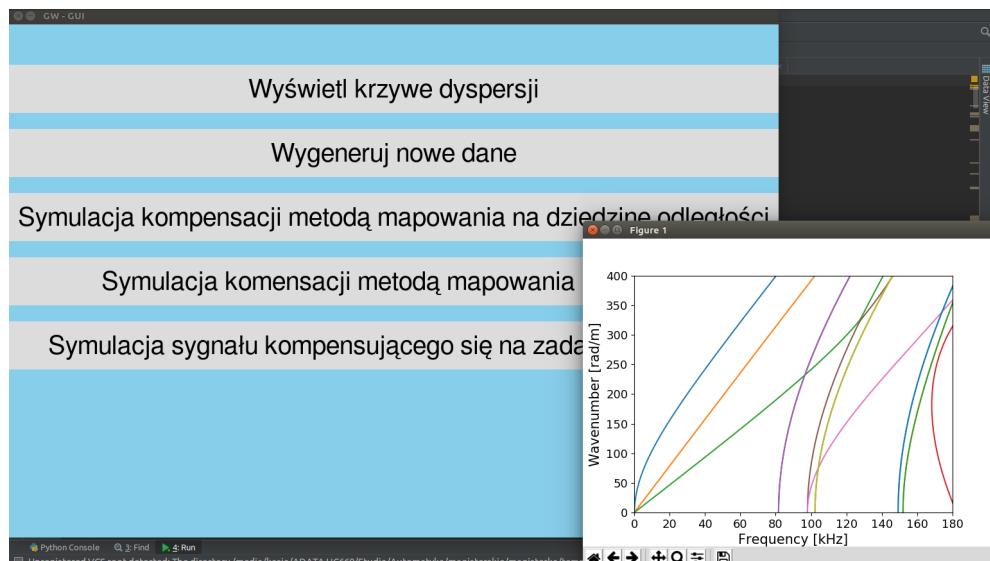
Warto zaznaczyć, iż jeśli wybrana została siatka czworościenna należy ustawić jedynie parametry w sekcji „Dla siatki czworościennej”. Parametry z drugiej sekcji nie będą brane pod uwagę. W przypadku wyboru siatki szesciościennej sytuacja jest analogiczna. Na rysunku 5.42 zaznaczona została opcja „Pokaż płaszczyznę”. Spowoduje ona, iż w trakcie trwania obliczeń, z momencie gdy zostanie wygenerowana płaszczyzna, zostanie one również wyświetlona do podglądu użytkownikowi. Przedstawia to rysunek 5.43. Gdyby zostały zaznaczone również dwie pozostałe opcje, wówczas kolejno zostałyby wyświetlane jeszcze: model pręta stworzony z wygenerowanych punktów oraz model pręta

z narysowaną siatką elementów skończonych. Aby rozpocząć obliczenia należy wcisnąć przycisk „Oblicz”.



Rys. 5.43. Okno obliczeń, z wyświetlona wygenerowaną płaszczyzną

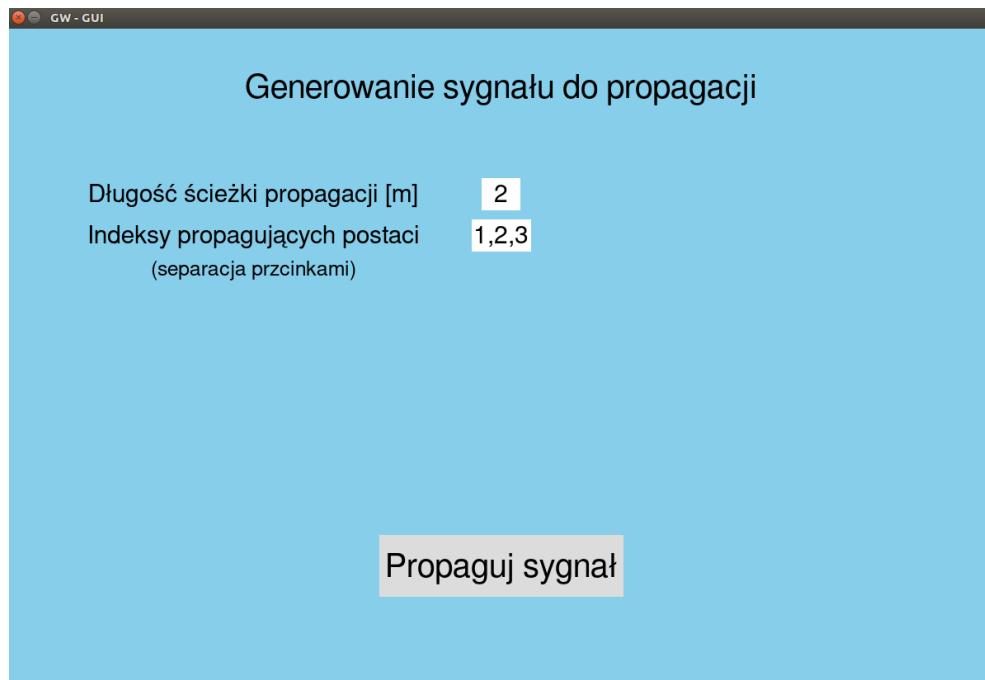
Ważne jest aby po oględzinach uzyskanych wyników zamknąć każde dodatkowo pojawiające się okno, przedstawiające siatkę, pręt lub też pręt z elementami skończonymi, ponieważ bez tego program nie przejdzie do kolejnego okna. Gdy obliczenia zostaną zakończona pojawi się okno przedstawione na rysunku 5.36. W tym momencie nowe dane zostały wygenerowane i są przechowywane w pamięci programu. Możliwy jest teraz podgląd wygenerowanych krzywych dyspersji poprzez przycisk „Wyświetl krzywe dyspersji”. Przedstawione zostało to na rysunku 5.44.



Rys. 5.44. Okno programu z wyświetlonymi krzywymi dyspersji

Aby móc korzystać z pozostałych funkcji programu należy zamknąć okienko z krzywymi.

Na tak wygenerowanych danych można testować zaimplementowane metody kompensacji dyspersji. Na przykład metodę mapowania sygnału z dziedziny czasu na dziedzinę odległości. Aby to zrobić należy nacisnąć przycisk „Symulacja kompensacji metodą mapowania z dziedziny czasu na dziedzinę odległości”. Następnie wyświetlony zostanie ekran służący do generowania sygnału omówionego na początku rozdziału 4., który przeprowadził zadaną odległość. Aby zasymulować sygnał, który został wprowadzony do założonego na początku sekcji pręta, przeprowadził jego długość, wynoszącą 2 metry i został odebrany na jego końcu, jako długość ścieżki propagacji należy wpisać 2. Przyjęto, że propagować będą trzy pierwsze postaci fali prowadzonej. Powyższe ustawienia ilustruje rysunek 5.45



Rys. 5.45. Okno programu przedstawiające ustawienia generowanego sygnału

Gdy obliczenia zostaną zakończone, wyświetlane zostanie okno z wynikami kompensacji. Po jego zamknięciu znów pojawi się główne menu, z którego poziomu możliwa jest symulacja kompensacji opracowanymi metodami, wyświetlenie aktualnie używanych krzywych dyspersji oraz wygenerowanie nowych danych.

6. Podsumowanie

KATARZYNA RUGIEŁŁO

Celem niniejszej pracy było stworzenie aplikacji, pozwalającej na symulację fali prowadzonej w długim stalowym pręcie o zadanych parametrach. Stworzenie opisywanej aplikacji wymagało dokładnego zrozumienia poruszanych zagadnień. Niezbędne było pogłębienie wiedzy w dziedzinie liniowej teorii sprężystości, zapoznanie się z rodzajami fal sprężystych, zrozumienie ich natury oraz zjawiska dyspersji. Kolejnym etapem było zapoznanie się z zagadnieniami metody elementów skończonych. Pierwszym krokiem w realizacji aplikacji było opracowanie oprogramowania wyliczającego krzywe dyspersji na podstawie zadanych parametrów. Aby to osiągnąć, niezbędne było zapoznanie się z zasadami budowy funkcji kształtu. Następnie niezbędne było zaimplementowanie algorytmów wyznaczających macierze mas oraz sztywności pojedynczych elementów skończonych. Kolejnym etapem było opracowanie algorytmów agregacji globalnych macierzy mas i sztywności. Zadaniem kolejnego algorytmu było wyznaczenie częstości własnych pręta na podstawie modelu MES. Stworzona aplikacja daje użytkownikowi możliwość własnoręcznego ustawienia niemal każdego parametru. Do dyspozycji są dwa rodzaje kształtu elementu skońzonego: elementy czworościenne oraz sześcioczienne. Istnieje również możliwość wczytania wartości z zewnętrznego programu o nazwie MARC. Wygenerowane krzywe przechowują informacje niezbędne do symulacji propagacji fali prowadzonej.

Kolejnym etapem było opracowanie algorytmu do symulacji propagacji fali. Aby było to możliwe zaimplementowany został algorytm segregacji danych wygenerowanych przez solver. W drugiej części pracy omówione zostały wybrane metody kompensacji dyspersji, które również zostały zaimplementowane i stanowią część prezentowanej aplikacji. Każda z proponowanych metod ma zarówno wady jak i zalety. Ważną część pracy stanowi zestawienie wyników uzyskanych z symulacji przy pomocy niniejszej aplikacji. Każda z metod została opisana od strony teoretycznej. Opisana została również ich implementacja numeryczna. Rozdział piąty poświęcony jest programistycznej stronie pracy. Opisana została tam instalacja niezbędnych narzędzi, umożliwiających sprawne korzystanie z opracowanego rozwiązania. Rozdział ten zawiera również listing ważniejszych części kodu, opis wszystkich funkcji, niezbędnych do korzystania z aplikacji wraz z przykładami

użycia oraz instrukcje pozwalającą na korzystanie ze stworzonego graficznego interfejsu użytkownika. Interfejs ten stanowi zwieńczenie aplikacji. Umożliwia korzystanie ze stworzonej aplikacji nawet osobom nie znających języka programowania python. Stworzona w ramach pracy aplikacja jest kompleksowa oraz wyposażona w wiele potrzebnych funkcji. Zarówno aplikacja jak i graficzny interfejs pozostawiona jest w postaci otwartego kodu.

Istnieje wiele możliwości dalszego rozwoju projektu. Z pewnością możliwa jest optymalizacja zastosowanych algorytmów, tak aby czas obliczeń uległ skróceniu. Dalszym krokiem rozwoju tego projektu z pewnością byłoby przetestowanie wyników symulacji na realnych obiektach, oraz weryfikacja oczekiwanych wyników z tymi uzyskanymi drogą symulacji. Kolejną możliwością rozwoju jest usprawnienie działania graficznego interfejsu oraz rozszerzenie symulacji o uwzględnienie tłumienia sygnału oraz charakterystyk wzbudzalności.

Podsumowując, w ramach niniejszej pracy w pełni zostały zrealizowane postawione na początku cele i założenia. Aplikacja działa w pełni sprawnie, jednak pozostaje ogromny obszar możliwego rozszerzenia powstałej pracy.

Bibliografia

- [1] J. L. Rose, “Ultrasonic guided waves in solid media,” pp. 1–39, 77–106, 120–133, 155–171, 2014.
- [2] T. STEPINSKI and K.-J. MATSSON, “Rock bolt inspection by means of rbt instrument,” *19 th World Conference on Non-Destructive Testing*, 2016.
- [3] A. S. Wolny, “Wytrzymałość materiałów,” pp. 1–41, 2002.
- [4] O. S. Z. Nazarchuk, V. Skalskyi, “Acoustic emission, methodology and applications,” pp. 29–69, 2017.
- [5] F. A. Amirkulova, “Dispersion relations for elastic waves in plates and rods,” pp. 29–69, 2011.
- [6] P. H. O. Cervena, “The problems at investigation of state of stress of thick orthotropic plate,” *Applied and Computational Mechanics*, vol. 1, no. 1, pp. 11–20, 2007.
- [7] L. Y. Z. Tian, “Lamb wave frequency–wavenumber analysis and decomposition,” *Journal of Intelligent Material Systems and Structures*, vol. 25, pp. 1107 – 1123, 2014.
- [8] G. S. G. Valsamosa, F. Casadeia, “A numerical study of wave dispersion curves in cylindrical rods with circular cross-section,” *Applied and Computational Mechanics*, vol. 7, p. 99–114, 2013.
- [9] P. P. P. Kijanka, W. J. Staszewski, “Generalised semi-analytical method for excitability curves calculation and numerical modal amplitude analysis for lamb waves,” *Structural Control and Health Monitoring*, vol. 25, 2018.
- [10] L. L. F. Treyssède, “Numerical and analytical calculation of modal excitability for elastic wave generation in lossy waveguides,” *The Journal of the Acoustical Society of America*, vol. 133, pp. 3827–3837, 2013.
- [11] K. J. Kwasniewski, I. Dominik, “A self-excited acoustical system for stress measurement in a cement plant,” *Mechanics and Control*, vol. 31, no. 1, pp. 29–34, 2012.

- [12] K. J. Kwasniewski, I. Dominik, “Harmonic analysis of self-excited acoustical system for stress changes measurement in compressed steel structural section,” *Solid State Phenomena*, vol. 198, pp. 639–644, 2013.
- [13] <https://www.americanpiezo.com> 2018.
- [14] http://www.moskat.pl/szkola/fizyka/drgania_i_fale.php?id=wielkosci_opisujace_fale 2018.
- [15] <https://fitz6.wordpress.com/2018/05/16/wednesday-may-16th-north-america-earthquakes-and-test-postponed-until-tomorrow/types-of seismic1/> 2018.
- [16] W. Śródka, “Trzy lekcje metody elementów skończonych,” 2004.
- [17] H. P. Gavin, “Structural element stiffness, mass, and damping matrices,” 2018.
- [18] J. Pointer, “Understanding accuracy and discretization error in an fea model,”
- [19] R. R. E. Wang, T. Nelson, “Back to elements - tetrahedra vs. hexahedra,” 2018.
- [20] A. K. Mal, P. C. Xu, and Y. Barcohen, “Leaky lamb waves for the ultrasonic non-destructive evaluation of adhesive bonds,” *J. Eng. Mater. Technol. Trans. ASME*, vol. 112, no. 3, p. 255–259, 1990.
- [21] P. D. Wilcox, “A rapid signal processing technique to remove the effect of dispersion from guided wave signals,” *ieee transactions on ultrasonics, ferroelectrics, and frequency control*, vol. 50, pp. 419–427, kwiecień 2003.
- [22] P. Wilcox, M. Lowe, and P. Cawley, “The effect of dispersion on long-range inspection using ultrasonic guided waves,” *NDT E Internat.*, vol. 34, no. 1, pp. 1–9, 2001.
- [23] J. Lin, H. Jiadong, L. Zeng, and L. Zhi, “Excitation waveform design for lamb wave pulse compression,” *IEEE Transactions on Ultrasonics Ferroelectrics and Frequency Control*, vol. 63, pp. 165–177, listopad 2015.
- [24] D. Alleyne, T. Pialucha, and P. Cawley, “A signal regeneration technique for long-range propagation of dispersive lamb waves,” *Ultrasonics*, vol. 31, pp. 201–204, maj 1993.
- [25] L. Liu and F. G. Yuan, “A linear mapping technique for dispersion removal of lamb waves,” *Structural Health Monitoring*, vol. 9, no. 1, pp. 1–12, 2010.