Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Електронної та біомедичної інженерії
(повна назва)

Кафедра Фізичних основ електронної техніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

РОЗРОБЛЕННЯ ГРАФІЧНОГО ІНТЕРФЕЙСУ ПАКЕТА MIT ELECTROMAGNETIC EQUATION PROPAGATION ДЛЯ ЗАДАЧІ ВИМІРЮВАННЯ ПАРАМЕТРІВ ГАУСОВИХ ПУЧКІВ
(тема)

Виконав:
студент 2 курсу, групи ФТОІм-21-1
Новицький В. В.
(прізвище, ініціали)

Спеціальність 152 «Метрологія та інформаційно-вимірювальна техніка»
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма «Фотоніка та оптоінформатика»
(повна назва освітньої програми)

Керівник проф. каф. ФОЕТ Одаренко Є. М.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Гнатенко О. С.
(підпис) (прізвище, ініціали)

2022 р.

Харківський національний університет радіоелектроніки

Факультет Електронної та біомедичної інженерії
(повна назва)
Кафедра Фізичних основ електронної техніки
(повна назва)
Рівень вищої освіти другий (магістерський)
Спеціальність 152 «Метрологія та інформаційно-вимірювальна техніка»
(код і повна назва)
Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма Фотоніка та оптоінформатика
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 20 ___ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Новицькому Владиславу Віталійовичу
(прізвище, ім'я, по батькові)

1.Тема роботи Розроблення графічного інтерфейсу пакета MIT Electromagnetic Equation Propagation для задачі вимірювання параметрів гаусових пучків

затверджена наказом університету від « 04 » листопада 2022 р. № 1444 Ст
2.Термін подання студентом роботи до екзаменаційної комісії 19 грудня 2022 р.
3. Вихідні дані до роботи роботи чисельний метод скінченних різниць в часові області (FDTD); пакет електромагнітного моделювання MEEP; закономірності розповсюдження випромінювання гаусових пучків; засоби розроблення графічних інтерфейсів

4. Перелік питань, що потрібно опрацювати в роботі
1 Ознайомлення з основними принципами використання пакету MEEP.
2 Розгляд гаусових пучків. 3 Розроблення базових тестових завдань для апробації графічної оболонки. 4 Розроблення програмного забезпечення для використання пакету MEEP

5. Перелік графічного матеріалу із зазначенням кресленików, схем, плакатів, комп'ютерних ілюстрацій

Установка експериментальна. Схема оптична структурна – А4 .

Кресленик деталі підставка регульована– А4 – 1шт.

Кресленик деталі стіл оптичний – А4 – 1 шт.

Демонстраційний матеріал – 13 шт.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Інформаційно-тематичний пошук та огляд літературних джерел про гаусові пучки	01.09.22–02.10.22	Виконано
2	Розробка серверного та графічного додатків для використання пакету MEEP	03.10.22–03.11.22	Виконано
3	Виконання чисельних розрахунків гаусових пучків	04.11.22–08.11.22	Виконано
4	Аналіз результатів розрахунків	09.11.22–14.11.22	Виконано
5	Оформлення пояснювальної записки	15.11.22–02.12.22	Виконано
6	Оформлення графічних та демонстраційних матеріалів	03.12.22–11.12.22	Виконано
7	Проходження нормоконтролю і отримання рецензії	12.12.22–15.12.22	Виконано
8	Проходження перевірки на плагіат	16.12.22–17.12.22	Виконано
9	Підготовка та захист кваліфікаційної роботи	18.12.22–20.12.22	

Дата видачі завдання 01 вересня 2022 р.

Студент _____
(підпис)

Керівник роботи _____ проф. каф. ФОЕТ Одаренко Є. М.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 53 с., 41 рис., 2 додатки, 23 джерела.

ПРОГРАМУВАННЯ, СЕРВЕРНИЙ ДОДАТОК, ПАКЕТИ ДО PYTHON, ОБРОБКА ЗОБРАЖЕННЬ, СИСТЕМА УПРАВЛІННЯ ПАКЕТАМИ, СИМУЛЯЦІЯ ЕЛЕКТРОМАГНІТНИХ ЯВИЩ, ГАУСОВІ ПУЧКИ

Об’єкт дослідження – пакет для симуляції електромагнітних явищ MEEP та його використання для вимірювання параметрів гаусових пучків.

Метою кваліфікаційної роботи є розробка серверного додатку та графічного інтерфейсу для використання пакету MEEP який використовується для симуляції електромагнітних явищ методом скінченних різниць в часовій області.

Методи дослідження – аналітичні та чисельні за допомогою мови програмування Python 3.7 та пакетів до неї. Також була використана мова програмування Dart та Flutter SDK до неї.

Під час проектування та розробки серверного додатку були використанні пакети: система управління пакетами Conda, фреймворк веб-додатків Flask, доповнення до Flask Flask-RESTplus, пакет для обробки та використання зображень imageio, пакет для симуляції електромагнітних явищ MEEP.

Під час проектування та розробки графічного інтерфейсу були використані пакети: FlutterFlow.

ABSTRACT

Explanatory note of qualifying work: 53 p., 41 drawings, 2 applications, 23 sources.

PROGRAMMING, SERVER APP, PYTHON PACKAGES, IMAGE PROCESSING, PACKAGE MANAGEMENT SYSTEM, ELECTROMAGNETIC SIMULATION, GAUSSIAN BEAM

The object of study is software package for electromagnetics simulation via the finite-difference time-domain (FDTD) MEEP and its usage for measuring parameters of Gaussian beams.

The purpose of the qualifying work is a development of a server application and graphical interface for the use of the MEEP package which is used to simulate electromagnetic phenomena by the finite difference method in the time domain.

Research methods – analytical and interesting with usage of Python 3.7 programming language and packages for it. Also was used Dart programming language and Flutter SDK.

During the design of the server application, the following packages were used: Conda package management system, Flask web application framework, additions to Flask Flask-RESTplus, imageio image processing and use package, MEER electromagnetic phenomena simulation package.

During the design of graphic interface, the following packages was used: FlutterFlow.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

BEM – Boundary Element Method – методи граничних елементів;

FDTD – finite-difference time-domain – метод кінцевих різниць в часовій області;

FEM – Finite Element Method – методи скінченних елементів;

MEEP – MIT Electromagnetic Equation Propagation;

MIT – Massachusetts Institute of Technology;

MPB – MIT Photonic Bands;

PML – Perfectly Matched Layer –ідеально узгоджені шари.

ВСТУП

Сучасні розроблені інструменти для моделювання розповсюдження електромагнітних хвиль дуже потужні, проте у них є одна дуже велика проблема. Ця проблема – це відсутність простого та зрозумілого графічного інтерфейсу, який полегшив би роботу з такими інструментами та заохотив нових людей спробувати пакети. Наразі у цих інструментів є можливість працювати з ними зазвичай за допомогою таких мов програмування як Python та Scheme, проте обидва потребують хоча б базового розуміння та знань програмування. Також, встановлення та використання цих інструментів потребує розуміння таких операційних систем як Linux.

Зважаючи всі фактори описані вище, розробка графічного інтерфейсу та серверного додатку для моделювання розповсюдження електромагнітних хвиль є актуальною проблемою.

Графічний інтерфейс дозволяє взаємодіяти із серверним додатком, що дозволяє спростити експерименти такого типу та дати починаючим науковцям та студентам використовувати або розвивати подібний інструмент що призведе до більшого та кращого розвитку пов’язаних з цією технологією наукових робіт.

За допомогою розробленої програми виконано розрахунковий проект, який дозволяє обчислювати характеристики Гаусового хвильового пучка при його розповсюджені в нелінійному середовищі.

1 ГАУСОВІ ПУЧКИ ТА ПАКЕТ MEEP

1.1 Гаусові пучки

Когерентне випромінювання, що генерується лазерами, є вузькими пучками, поперечні розміри яких, проте, набагато більше довжини хвилі. Тому дифракційна розбіжність таких пучків порівняно невелика, та його амплітуда повільно змінюється з поздовжньої координатою. Такі світлові пучки добре описуються гаусовими пучками, в яких амплітуда в поперечній площині змінюється за законом Гауса-Ерміта, а фазова поверхня викривляється, але в міру поширення пучка. Зараз доведено, що гаусові пучки найбільш просто і повно описують властивості лазерних світлових пучків і типи коливань (моди) відкритих резонаторів ОКГ (рис. 1.1) .

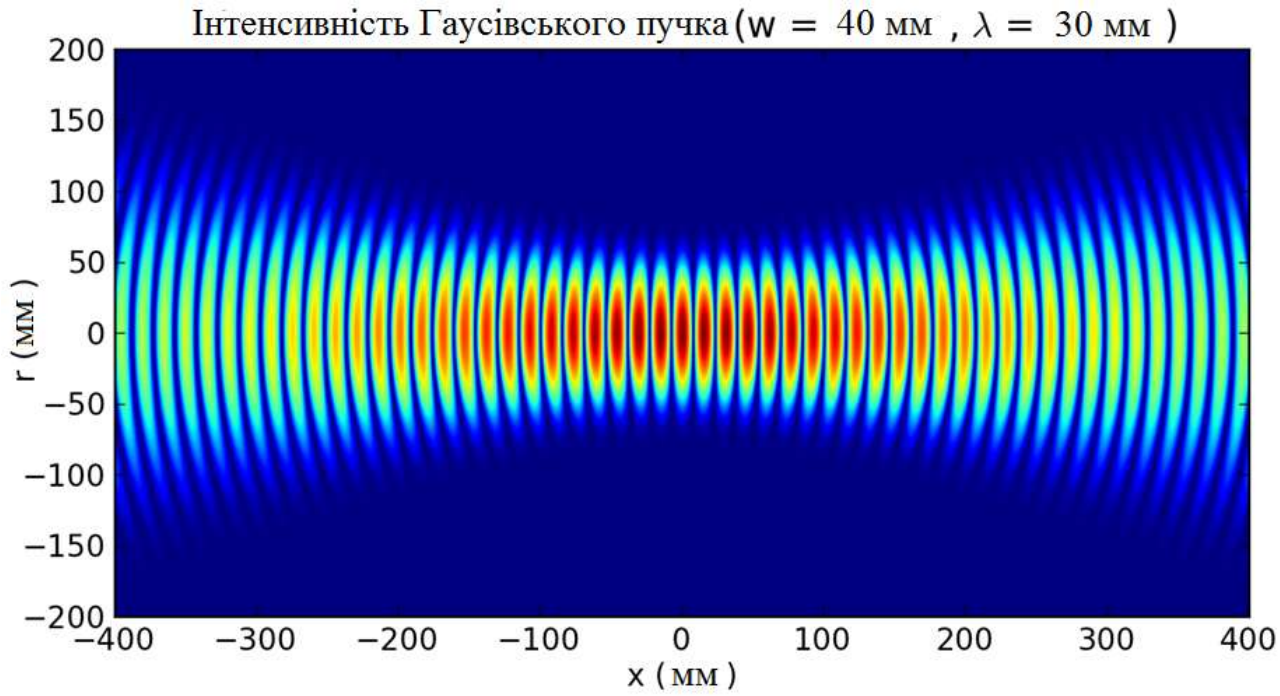


Рисунок 1.1 – Гаусовий пучок

Основні параметри гаусового пучка:

− ширина пучка або «розмір плями»;

− діапазон Релею та параметр конфокальності;

− радіус кривизни.

Розмір плями $\acute{\omega}(z)$ для гаусового пучка поширюється у вільному просторі буде мати мінімальне значення в одному місці променя, званому перетяжкою, наочно показано на рис.1.2 [1]. Для променя з довжиною хвилі на відстані від перетяжки вздовж променя зміна розміру плями описується:

$$w(z) = w_0 \sqrt{1 + (\frac{z}{z_R})^2}, \quad \#(1.1)$$

де початок осі збігається з перетяжкою променя.

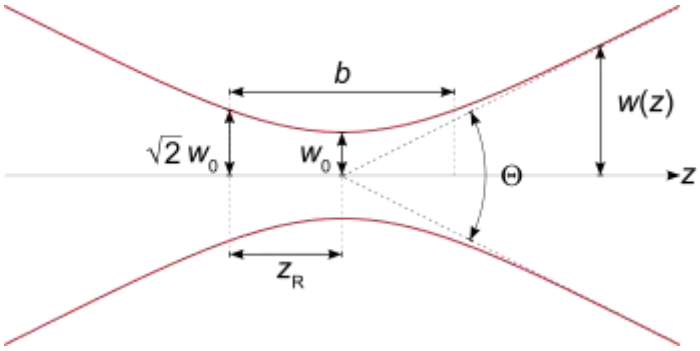


Рисунок 1.2 – Поздовжній переріз гаусового пучка

Інтенсивність випромінювання у поздовжньому перерізі гаусівського пучка, що фактично є квадратом амплітуди поля, є нерівномірною координатною функцією (рис. 1.3).

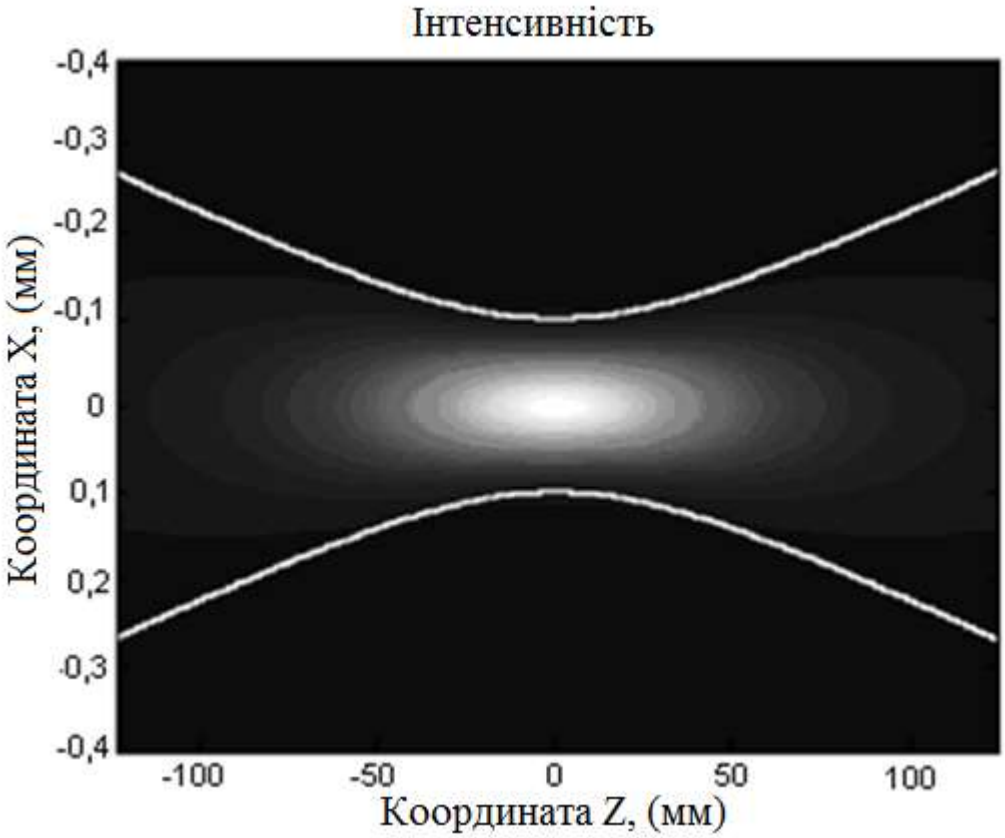


Рисунок 1.3– Розподіл інтенсивності випромінювання у поздовжньому перерізі гаусівського пучка

Вочевидь, що максимум поля реалізується у сфері, де поперечні розміри пучка мінімальні. Це область перетяжки. Саме тому її іноді називають псевдофокус пучка . На відстані від перетяжки, що дорівнює діапазону Релея, розмір плями дорівнює:

$$\omega(\pm z_R) = \omega_0\sqrt{2}. \#(1.2)$$

Відстань між двома такими точками називається параметром конфокальності або глибиною фокусу променя:

$$b = 2z_R = \frac{2\pi {\omega_0}^2}{\lambda}. \#(1.3)$$

Мода TEMooq конфокального резонатора – це сферична хвиля, що йде з центру і має гаусовий розподіл інтенсивності в площині, перпендикулярній напряму поширення. При цьому радіус кривизни сферичного хвильового фронту в міру поширення змінюється за законом:

$$R = z + (k\omega_0{}^2)^2/z, \#(1.4)$$

на великих відстанях від початку координат ($z >> k\omega_0{}^2 = 1/2$) збігається з відстанню від резонатора до фронту хвилі: $R \approx z$.

Це означає, що в дальній зоні хвильовий фронт гауссова пучка наближається до хвильового фронту сферичної хвилі, що розповсюджується з точки, розташованої на осі пучка в місці фокальної його перетяжки. При $z = \frac{l}{2}$ радіус $R = l$, тобто, як і слід було очікувати, на поверхні дзеркала хвильовий фронт збігається зі сферичною поверхнею дзеркала. На рис.1.4 показано огинаюча пучка в резонаторі і хвильові фронти.

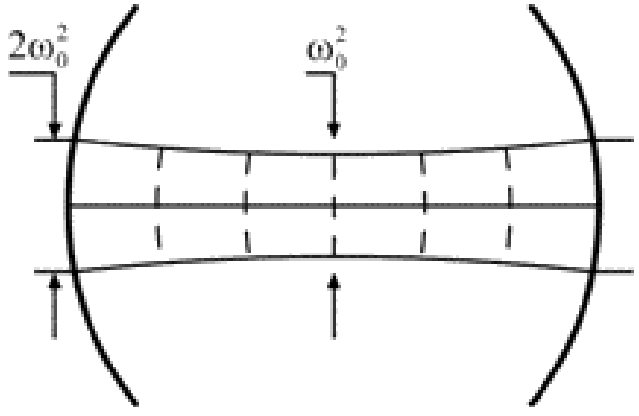


Рисунок 1.4 – Огинаюча інтенсивність гаусового пучка в конфокальному резонаторі та хвильові фронти

Разом з тим дуже важливо, що при $z \to 0,\ R \to \infty$. Площина симетрії резонатора або, що те саме, його фокальна площина є поверхнею постійної фази. Це означає, що у фокальній перетяжці хвиля є плоскою, але просторово обмеженою ефективним розміром ω_0. Саме цей розмір визначає розбіжність моди TEM_{ooq} [2].

На великій відстані від резонатора ($z >> k\omega_0^2 = l/2$) ширина $\omega = \frac{z}{k\omega_0}$, чому відповідає кутова розбіжність:

$$\Theta = \frac{\omega}{z} = 1/k\omega_0. \#(1.5)$$

В результаті основна частина енергії гаусового пучка зосереджена в тілесному куті:

$$\Omega = \pi\Theta^2 = \frac{\lambda}{l}. \#(1.6)$$

Таким чином, розбіжність лазерного випромінювання в основній моді визначається не поперечним, а поздовжнім розміром лазерного резонатора. Це є наслідком того, що найменшим ефективним отвором, на якому відбувається дифракція випромінювання гаусового пучка, що вільно розповсюджується, є фокальний переріз його каустики. Дифракційна розбіжність визначається ставленням довжини хвилі λ до ширини розповсюдження інтенсивності в області перетяжки ω_0.

Потужність P у поперечному перерізі пучка радіусом r дорівнює:

$$\mathrm{P(r,z)} = P_0\left[1 - \exp\left(\frac{-2r^2}{w^2(z)}\right)\right], \#(1.7)$$

де $P_0 = \frac{1}{2}\pi I_0 w_0^2$ – загальна потужність у пучку.

Пікова інтенсивність на відстані z від перетяжки обчислюється за допомогою правила Лопіталя як межа потужності в поперечному перерізі радіусом r, поділеної на площу цього перерізу πr^2:

$$\mathrm{I}(0,z) = \lim_{r\to 0} \frac{P_0\left[1-\exp\left(\frac{-2r^2}{w^2(z)}\right)\right]}{\pi r^2} = \frac{P_0}{\pi}\lim_{r\to 0}\frac{\left[-(-2)(2r)\exp\left(\frac{-2r^2}{w^2(z)}\right)\right]}{w^2(z)(2r)} = \frac{2P_0}{\pi w^2(z)}. \#(1.8)$$

Пікова інтенсивність дорівнює подвоєної середньої інтенсивності, отриманої шляхом поділу загальної потужності на площу перерізу радіусом $r = w(z)$.

Гаусівські пучки мають велике значення завдяки їх унікальним властивостям:

– дані пучки мають гаусівський профіль розподілу інтенсивності у будь-якому поперечному перерізі пучка; змінюється тільки радіус пучка;

– гаусівський пучок залишається гаусівським також після проходження через оптичні елементи простих видів (наприклад, лінзи без відхилень);

–- гаусівські промені – самоузгоджений розподіл поля самого низького порядку в оптичних резонаторах (моди резонатора) при умові, що немає жодних внутрішньорезонаторних елементів, які викликають спотворення пучка. З цієї причини, вихідні пучки багатьох лазерів є гаусівськими;

– є так звані моди вищих порядків, наприклад ерміт-гаусівського типу. Вони мають складніший розподіл поля.

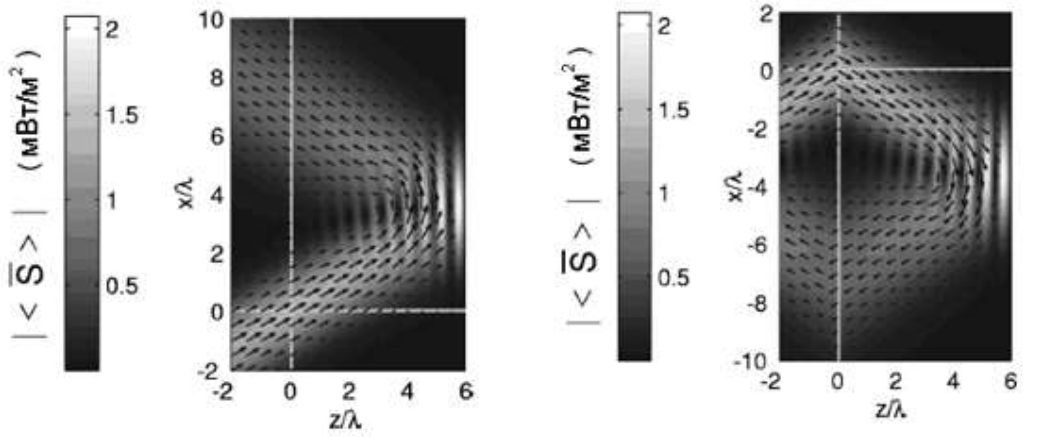
1.2 Вектор Пойнтінга

Вектор Пойнтінга (також вектор Умова-Пойнтінга) – вектор густини потоку енергії електромагнітного поля. Вектор Пойнтінга можна визначити через векторний добуток двох векторів:

$$\vec{S} = \left[\vec{E} \times \vec{H}\right], \#(1.9)$$

де $\vec{E}$ і $\vec{H}$ – вектора напруженості електричного та магнітного полів, відповідно .

На рис. 1.5 показано напрям вектора Пойнтінга в падаючому і відбитому пучках при різних діелектричних та магнітних складових.



а) $\varepsilon_1 = \varepsilon_0,\ \mu_1 = \mu_0$ б) $\varepsilon_1 = -\varepsilon_0,\ \mu_1 = -\mu_0,$ ($d = \lambda$)

Рисунок 1.5 – Розподіл щільності потоку енергії падаючого та відбитого гаусових пучків з різними параметрами

У роботі Jin Au Kong, Bae Ian Wu розглядається бічне зміщення гаусового пучка на межі поділу середовищ з негативною постійною діелектричною та магнітною проникністю (рис. 1.5).

1.3 Багатовимірні гаусові пучки

Так як поперечні розміри для еліптичного пучка по різних координатних осях різняться, то цікавий розгляд залежностей ширини пучка і радіуса кривизни його хвильового фронту від поздовжньої координати.

Ці залежності представлені на рис. 1.6. Слід зазначити існування точок перетину кривих, що відповідають різним поперечним координатним осям. Оскільки у цих точках збігаються поперечні розміри пучка (рис. 1.6, а), то реалізується кругова форма просторового розподілу інтенсивності випромінювання. Перетин кривих (рис. 1.6, б) відповідають сферичному хвильовому фронту пучка, оскільки в цих точках рівні радіуси кривизни хвильового.

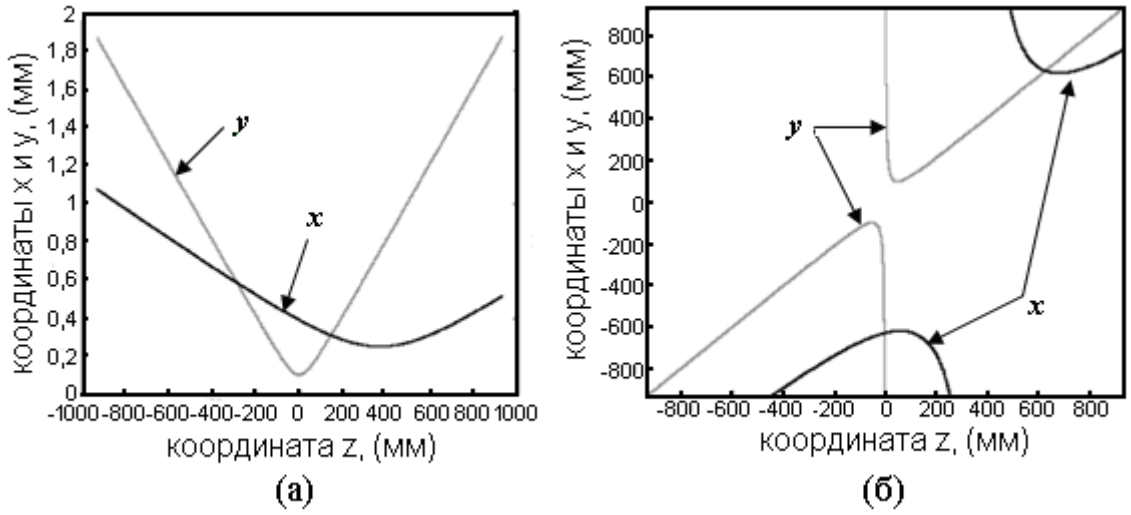


Рисунок 1.6 – Залежність ширини гауссовського пучка від радіуса кривизни його хвильового фронту та від продольної координати фронту по взаємно перпендикулярним поперечним напрямкам

Важливим результатом цієї побудови є відсутність просторового збігу реалізації кругової форми розподілу інтенсивності випромінювання та сферичного хвильового фронту [3].

На рис. 1.7 представлено тривимірну реалізацію залежності ширини хвильового пучка, що володіє неортогональним астигматизмом. Для формування такого хвильового пучка зазвичай застосовують пари циліндричних або тороїдальних лінз, у яких характеристичні осі повернені на деякий кут (відмінний від нуля і 900) відносно один одного.

На підставі наведених вище співвідношень можна отримати вираз для кутової розбіжності гаусового пучка:

$$\theta = \lim_{z\to\infty}\left(\frac{2w(z)}{z}\right) = \lim_{z\to\infty}\left(\frac{2w_0}{z}\sqrt{1+\left(\frac{z}{b}\right)^2}\right) = \frac{2w_0}{b} = \frac{2\lambda}{\pi w_0}. \#(1.10)$$

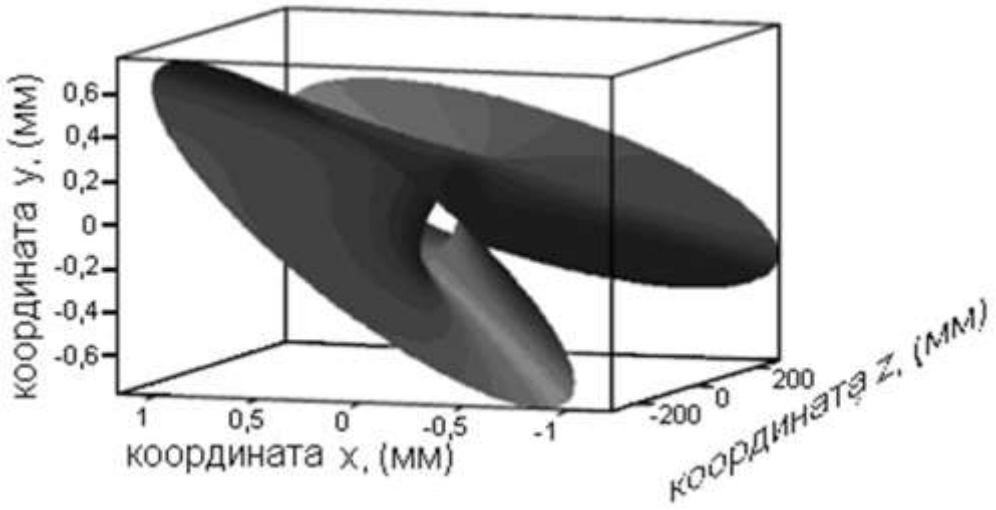


Рисунок 1.7 – Тривимірне уявлення координатної залежності ширини гаусового пучка

Видно, що цю розбіжність можна розуміти як дифракції, що з'явилася в результаті, на отворі з радіусом порядку w_0.

Гаусов пучок буде власною хвилею резонатора у разі, якщо дві його хвильові поверхні співпадуть із поверхнями дзеркал резонатора. Радіус

гауссового пучка і кривизна його хвильової поверхні описуються одним комплекснозначним параметром:

$$\frac{1}{q} = \frac{1}{R} + i\frac{\lambda}{\pi {w_0}^2}. \#(1.11)$$

та має чудову властивість, що він перетворюється при проходженні світла через оптичні елементи так само, як радіус геометричного хвильового фронту променів в геометричній оптиці, тобто за законом:

$$q = \frac{Aq_1 + B}{Cq_1 + D}, \#(1.12)$$

де A, B, C, D – елементи променевої матриці передачі оптичного елемента;

q_1, q_2 – параметри гаусового пучка на вході та виході оптичного елемента.

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix}_z = \begin{pmatrix} 1 & z \\ 0 & 1 \end{pmatrix}. \#(1.13)$$

Поширення у вільному просторі на відстань можна трактувати як проходження через оптичний елемент із матрицею.

1.4 Пакет MEEP

Одним з найпопулярніших інструментів для обчислювання у класичному електромагнетизмі є алгоритм кінцевої різниці часової області (FDTD), який ділить простір і час на регулярну сітку і моделює еволюцію рівнянь Максвелла. MIT створив безкоштовну реалізацію з відкритим кодом алгоритму FDTD: MEEP (абревіатура MIT Electromagnetic Equation Propagation) (рис 1.8). MEEP виступає повнофункціональним, зокрема:

– довільні анізотропні, нелінійні та дисперсійні електричні та магнітні середовища;

– різноманітність граничних умов, включаючи симетрії та ідеально узгоджені шари (PML);

– паралелізм розподіленої пам'яті;

– декартові (1d / 2d / 3d) та циліндричні координати;

– гнучкі обчислення виводу та поля [4].

Він також характеризується деякими незвичайними функціями, такими як вдосконалена обробка сигналів для аналізу резонансних режимів, точне усереднення субпікселів, вирішувач частотної області, який використовує код часової області, повна сценарійність та інтегровані засоби оптимізації.

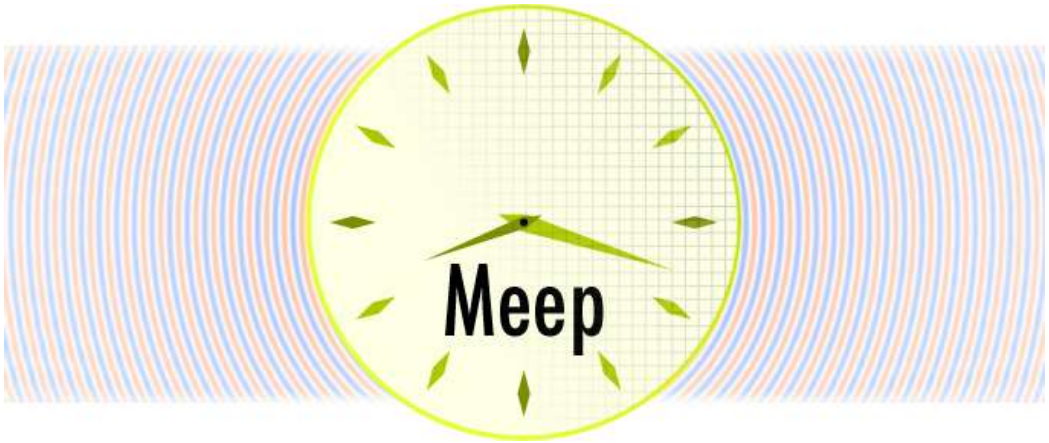


Рисунок 1.8 – Емблема пакету MEEP [5]

Зараз замість розбору відомого алгоритму FDTD (який детально висвітлений в іншому місці), ми вирішили зосередити увагу на конкретних дизайнерських рішеннях, які містилися в розробці MEEP, мотивація яких може не проглядатися з описів FDTD підручника, напруженість між абстракцією та продуктивність у реалізаціях FDTD, а також нетрадиційні або специфічні особливості цього програмного забезпечення.

Навіщо пропонувати впровадження ще однієї програми FDTD? Можна придбати десятки комерційних програмних пакетів FDTD. Однак потреби

досліджень часто вимагають гнучкості. Це гарантується доступом до вихідного коду (й ослаблених обмежень ліцензування для швидкого перенесення на нові кластери та суперкомп'ютери). Зв'язок з іншими дослідниками фотоніки повідомляє про те, що велика частина груп у кінцевому підсумку створює власний код FDTD, щоб задовольнити свої потреби (рис 1.9) (групи в MIT використовували принаймні три різнобічні власні впровадження FDTD протягом останніх 15 років), а це примножує зусилля, що не приносить користі.

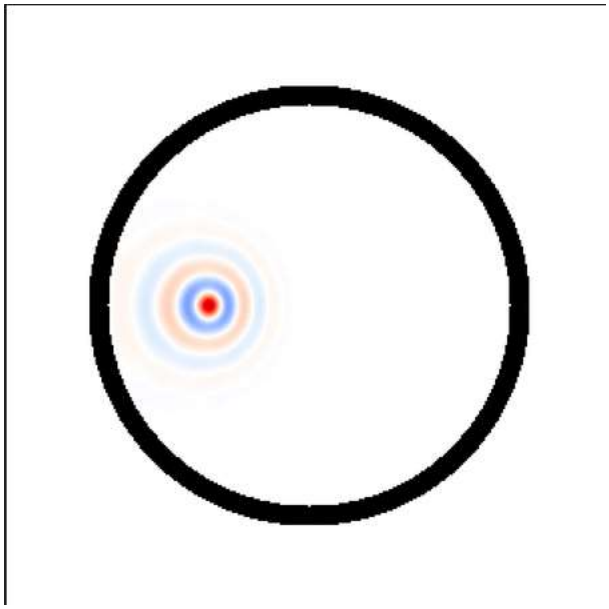


Рисунок 1.9 – Приклад результату моделювання за допомогою пакету MEEP

Більшість з них не було показано громадськості. До того ж кілька інших програм з вільним програмним забезпеченням FDTD, які були доступні до завантаження, коли MEEP вперше вийшов у 2006 році, виявилися недостатньо повнофункціональними для цілей широкого суспільства. З того моменту MEEP цитується в більш ніж 100 публікаціях журналів, а його завантаження здійснюється понад 10 000 разів, що служить підтвердженням попиту на такий пакет (рис 1.10).

Алгоритми FDTD, звичайно, є лише одним із багатьох числових інструментів, які були розроблені в обчислювальному електромагнетизмі, і,

скоріше всього, стануть здаватися найзвичайнісінькими в порівнянні з іншими складними методами, серед яких методи скінченних елементів (FEM) з високою точністю порядку та/або адаптивні неструктуровані сітки або навіть кардинально різні підходи, такі як методи граничних елементів (BEM), які дискретизують лише інтерфейси між однорідними матеріалами, а не обсягами. Кожен інструмент, звичайно, може похвалитися плюсами, й у нього також знаходяться мінуси, і ми не віримо, що будь-який з них є правильним рішенням. Наприклад, неоднорідні неструктуровані сітки FEM відрізняються кардинальними перевагами для металевих конструкцій, де довжини хвиль мікрометрів можуть поєднуватися з нанометровими глибинами шкіри.

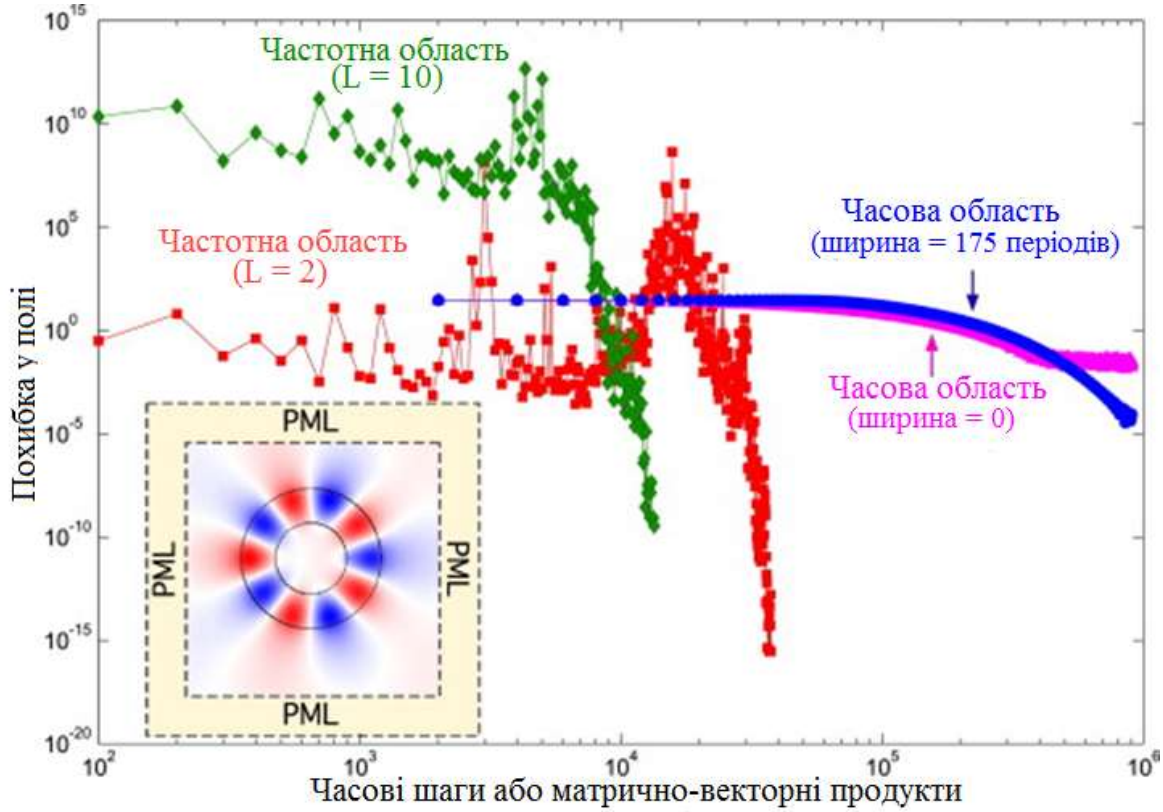


Рисунок 1.10 – Приклад складання графіків за допомогою пакету MEEP [6]

Під іншим поглядом цю гнучкість можливо досягти ціною значної складності програмного забезпечення, що може не приносити користь для діелектричних приладів на інфрачервоних довжинах хвиль (наприклад, в

інтегрованій оптиці або волокнах), де показник заломлення (і, отже, типова роздільна здатність) змінюється на менше чотирьох коефіцієнтів між матеріалами, а ось несуттєві особливості, поміж яких шорсткість поверхні, можуть бути точно оброблені методами пертурбації. BEM, засновані на формулах інтегральних рівнянь електромагнетизму, особливо потужні для розсіяння задач, що стосуються малих об'єктів у великому обсязі, оскільки обсяг не потрібно дискретизувати і не потрібні штучні "поглинаючі межі".

Крім того, BEM пропонують ряд обмежень: не виключено, що вони вимагатимуть штучних поглиначів для інтерфейсів, що розширюються до нескінченності (таких як вхідні / вихідні хвилеводи) будь-яка зміна функції Гріна (наприклад, введення анізотропних матеріалів, накладення періодичних або симетричних граничних умов або перехід з трьох на два виміри) вимагає повторної реалізації великих частин програмного забезпечення (наприклад, інтеграція окремих панелей та швидкі розв'язувачі) замість суто локальних змін, як у FDTD або FEM; постійно мінливі (на відміну від кусково-постійних) матеріали неефективні; і рішення у часовій області (а не в частотній області, що є недостатнім для нелінійних або активних систем, в яких частота не зберігається) з BEM вимагає дорогого вирішувача, який є нелокальним як у часі, так і в просторі. І тоді, звичайно, з'являються спеціалізовані інструменти, які вирішують лише певний тип електромагнітних проблем, таких як наше власне програмне забезпечення MPB, яке робить обчислення лише власні режими (наприклад, хвилеводні режими), які вважаються потужними та надійними у своїй галузі, але не служать замінником моделювання Максвелла загального призначення. З переваг FDTD варто відзначити простоту, загальність і надійність: просто впровадити повні рівняння Максвелла, які залежать від часу, для майже довільних матеріалів (включаючи нелінійні, анізотропні, дисперсійні та мінливі в часі матеріали) та широкий спектр граничних умов, можна в короткий час експериментувати з новою фізикою, яка з'єднана з рівняннями Максвелла (такими як популяції збуджених атомів для генерації), і алгоритм нескладно паралелізується для роботи на кластери або

суперкомп'ютери. Ця простота особливо приваблює дослідників, основним завданням яких є вивчення нових взаємодій фізичних процесів, для яких час програмістів та навчання нових студентів набагато дорожчий, ніж комп'ютерний.

1.5 Граничні умови та симетрії

Зважаючи на те, що дозволяється моделювати лише кінцеву область простору, моделювання завжди має закінчуватися з деякими граничними умовами (рис). Три основні типи закінчень підтримуються в MEEP: періодичні межі Блоха, металеві стінки та поглинаючі шари PML (рис 1.11). Крім того, можна використати симетрії задачі для подальшого зменшення обчислювальних вимог.

За звичайних періодичних меж у комірці розміром L компоненти поля задовольняють $f(x+L) = f(x)$. Блохова періодичність – це узагальнення, де $f(x+L) = e^{ik_x L} f(x)$ для деякого хвильового вектора Блоха k. Це може бути використано для вирішення режимів хвилеводів, решіток тощо, як у MPB.

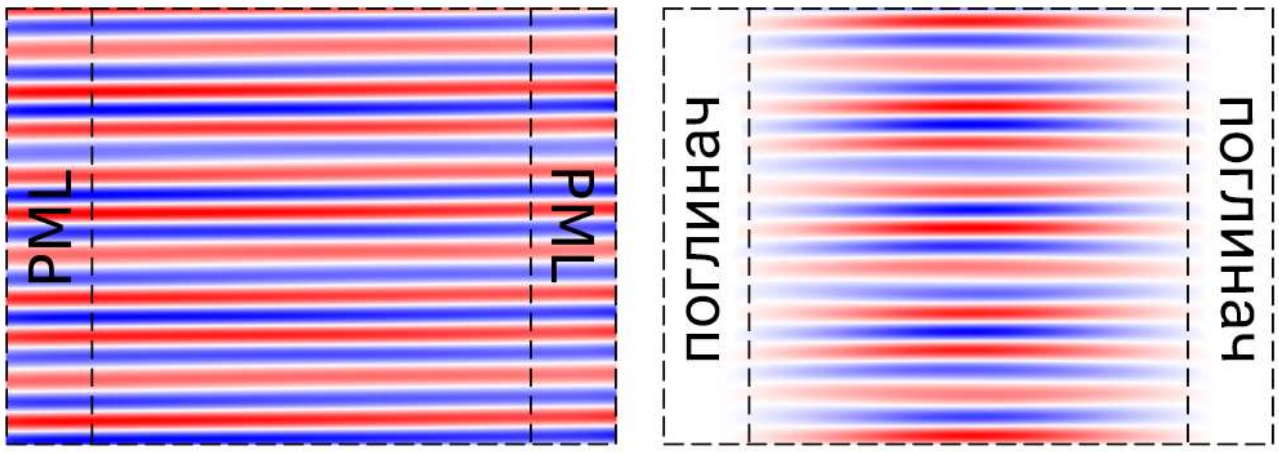


Рисунок 1.11 – Приклади різних граничних умов [7]

Металева стінка виступає ще простішою граничною умовою, де поля просто змушені дорівнювати нулю на кордонах, ніби клітину оточує ідеальний метал (нульове поглинання, нульова глибина шкіри). Слід підкреслити, що у вас є можливість розмістити ідеальні металеві матеріали в якому завгодно місці в обчислювальній комірці, наприклад для імітації металевих порожнин довільної форми. Для моделювання відкритих граничних умов бажано, щоб поглинання кордонами поширювалося на всі хвилі, що падають на них, без відображень. Це реалізовано з терміном, котрий називається ідеально узгоджені шари (PML). PML не є граничною умовою – швидше, це спеціальний поглинальний матеріал, що знаходиться неподалік від кордонів. PML насправді є вигаданим (нефізичним) матеріалом, який розрахований на нульове відображення на своєму інтерфейсі. Однак PML є безрефлексійною в теоретичній безперервній системі, у фактично дискретизованій системі є деякі невеликі відображення, які надають системі недосконалості. Саме тому PML завжди надає деяку кінцеву товщину, коли поглинання поступово включається. Інший спосіб зменшення обчислювальної комірки – це симетрія. Скажімо, ви знаєте, що ваша система має дзеркальну площину симетрії (як у структурі, так і в поточних джерелах), тоді ви можете заощадити коефіцієнт два, лише змоделювавши половину структури, а отримавши другу половину за допомогою дзеркального відображення. MEEP може застосувати декілька різновидів дзеркальної та обертової симетрій - вона розроблена так, що симетрія є суто оптимізацією, і крім зазначення симетрії, ваше обчислення налаштовано точно так само.

1.6 Методи скінченної різниці в часовій області

Методи FDTD поділяють простір і час на кінцеву прямокутну сітку. MEEP докладає зусиль максимально приховати цю дискретність від користувача, але існує кілька наслідків дискретизації. Щоб дискретувати рівняння з точністю другого порядку, методи FDTD зберігають різні

компоненти поля в різних місцях сітки. Така дискретизація відома під назвою Йі (рис 1.12). Як результат, MEEP повинен інтерполювати компоненти поля до загальної точки, коли потрібно об'єднати, порівняти або вивести компоненти поля (наприклад, при обчисленні густини енергії або потоку). В основному, немає потреби занадто турбуватися про цю інтерполяцію, адже вона є автоматичною. Однак, оскільки це проста лінійна інтерполяція, тоді як E і D можуть бути розривними через межі діелектрика, це означає, що інтерпольовані поля E і D можуть бути менш точними, ніж можна було очікувати прямо навколо діелектричних інтерфейсів.

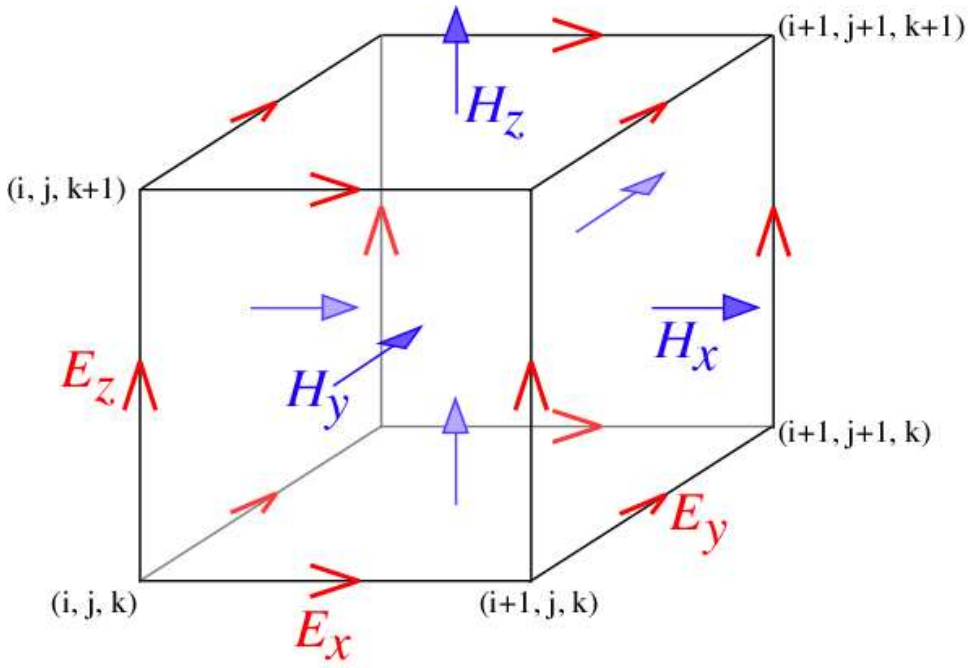


Рисунок 1.12 – Решітка Йі [8]

1.7 Ілюзія безперервності

Враховуючи використання FDTD дискретизованого простору і часу, MEEP робить спроби зберегти ілюзію, що використовується безперервна система. На початку моделювання треба вказати просторову роздільну

здатність, надалі ви, як правило, працюєте в безперервних координатах у вибраних одиницях.

Наприклад, ви вказуєте діелектричну функцію як функцію $\varepsilon(x)$ безперервного x, або як сукупність твердих геометричних об'єктів, таких як “Сфера”, “Циліндр” та ін., а перед MEEP стоїть завдання з'ясувати те, як вони повинні бути представлені на дискретній сітці. Або якщо ви хочете вказати точку “Джерело”, ви просто вказуєте точку *x*, де ви хочете, щоб джерело знаходилося – MEEP визначить найближчі точки сітки до x і додасть струми до цих точок, зважені відповідно до їх відстані від *x* (рис 1.13). При постійній зміні *X*, сила струму в MEEP також буде схильна до постійних змін, змінюючи вагу. Якщо задати потік Пойнтінга через певний прямокутник, то MEEP буде лінійно інтерполювати значення поля з сітки на цей прямокутник. Зрізи масивів полів та матеріалів безперервно інтерполюються під час переміщення положення зрізу.

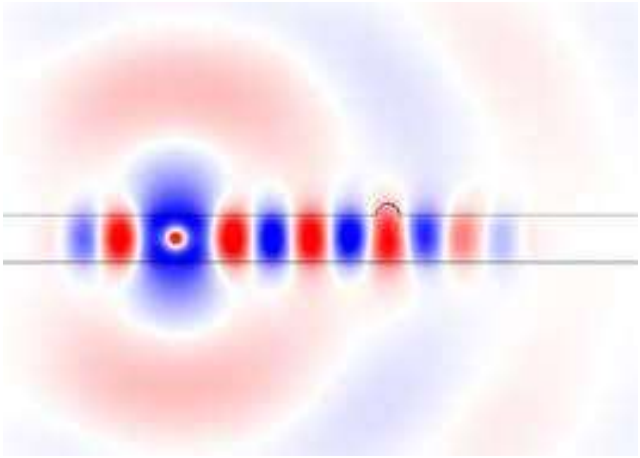


Рисунок 1.13 – Моделювання розповсюдження електромагнітних хвиль у нескінченному хвилеводі [9]

Загалом, філософія інтерфейсу MEEP полягає у всепроникаючій інтерполяції, тому при постійній зміні будь-якого виходу відповідь імітації IP буде змінюватися також безперервно, і в підсумку він буде максимально

швидко і плавно сходитися до безперервного рішення при збільшенні просторового дозволу.

Так, функція ε, яку внутрішньо використовує MEEP – це не просто дискретно вибіркова версія $\varepsilon(x)$, що була визначена користувачем. Кожна точка сітки є середнє значення ε в оточуючому пікселі. Субпіксельне згладжування MEEP зроблено спеціально для того, щоб мінімізувати "сходи" та інші помилки, що виникли в результаті різких інтерфейсів.

1.8 Інші чисельні методи в обчислювальній електромагнетиці

FDTD, звичайно, не єдиний чисельний метод в обчислювальній електромагнетиці, і не завжди є найбільш підходящим. Необхідно мати у своєму арсеналі кілька інструментів і потім зупинитися на найбільш зручному для кожного завданні (рис. 1.14). Наприклад, хоча FDTD може бути використаний для обчислення електромагнітних власних режимів, у структурах оминаючи втрати, часто швидше, простіше та надійніше використовувати спеціалізований власний метод вирішення, такий як MPB.

Для обчислення схеми поля чи відгуку структури на одній частоті може бути більш ефективним безпосереднє вирішення відповідного лінійного рівняння, а не ітерація в часі. Однак, стосовно випадків, коли мова йде про великі відмінності в масштабі (наприклад, з металами з невеликою глибиною), може статися, що бажано віддавати перевагу методу, що дає можливість змінювати роздільну здатність у різних просторових областях, поміж яких – кінцевий елемент або граничний метод елемента. Методи граничних елементів особливо потужні, коли у вас є велике відношення об'єму до поверхні, наприклад, для розрахунків розсіювання на невеликих об'єктах у великому (тобто нескінченному розмірі) обсязі.

Силою методів часової області є їх здатність отримувати весь частотний спектр відповідей (або власні частоти) в одному моделюванні, перетворюючи відповідь Фур'є на короткий імпульс або використовуючи більш складні методи

обробки сигналів, такі як Harminv. Методи скінченних елементів також можуть бути використані для полів, що розвиваються в часі, але вони зазнають серйозних недоліків у порівнянні з методами скінченних різниць: методи скінченних елементів для стабільності, як правило, повинні використовувати якусь форму неявного кроку в часі, де вони повинні інвертувати матриця (вирішити лінійну систему) на кожному часовому кроці.

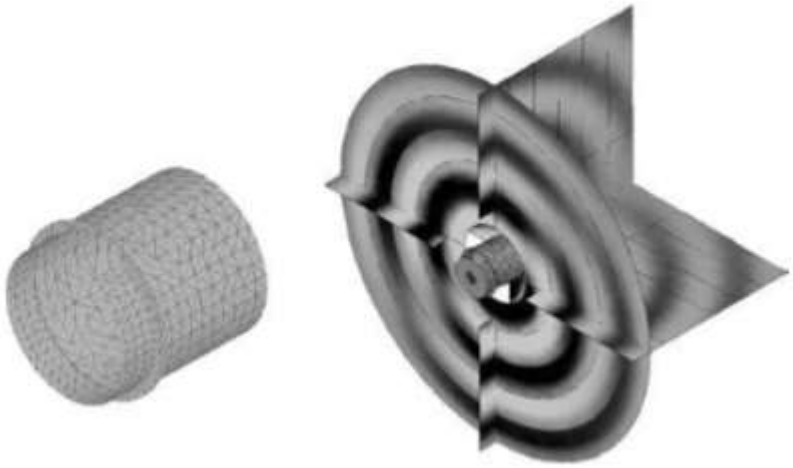


Рисунок 1.14 – Зв’язана електромагнітна-механічно-акустична модель електродвигуна постійного струму розроблена за використання методів граничних елементів [10]

2 СЕРВЕРНИЙ ТА КЛІЄНТСЬКИЙ ДОДАТКИ ДЛЯ ВИКОРИСТАННЯ ПАКЕТУ MEEP

2.1 Проблеми існуючих інтерфейсів для пакету MEEP

Графічні інтерфейси для пакету MEEP вже існують у сьогоденні. Однак більшість з них має деякі перепони для незнайомих з застосуванням або встановленням таких наукових пакетів. Також програмування не є опцією для широкого загалу. Конкретно для пакету MEEP існує декілька проблем: необхідність встановлення цього інструменту на свій пристрій та виключно консольний або програмний інтерфейс. Такий сервіс як Nanohub.org має в собі пакет MEEP (рис. 2.1) для використання без необхідності встановлення, прямо в браузері. Однак, навіть зважаючи на те, що проблему того, що треба встановити пакет MEEP Nanohub.org вирішує, залишається інша проблема у вигляді відсутності графічного інтерфейсу. Nanohub.org дає можливість запускати скріпти написані за допомогою однієї з доступних для пакету MEEP мов програмування. Це потребує від користувача володіння однією із таких мов. Підтримувані мови, Python, Scheme, C++ не є найбільш поширеними або найлегшими в освоюванні.

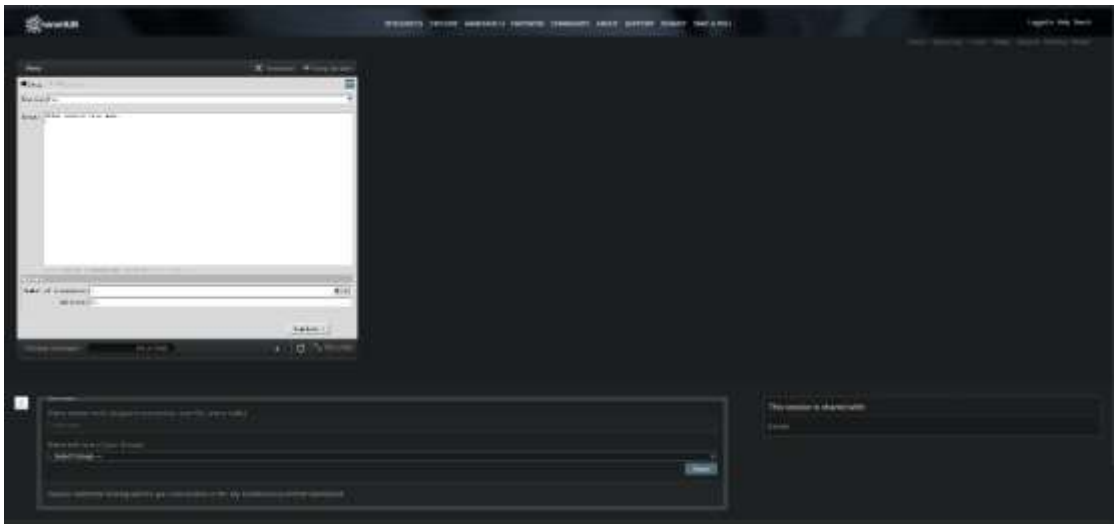


Рисунок 2.1 – Приклад графічного інтерфейсу на сайті nanohub.org [11]

Розглянувши проблему встановлення треба розібрати проблему графічних інтерфейсів. Існують декілька варіантів графічних інтерфейсів для пакету MEEP. Проте тут знову повертається проблема встановлення пакету на пристрій, оскільки ці пакети викликають команди з програмного інтерфейсу додатку який повинен бути встановлений на пристрої. Таким чином і вже існуючі графічні інтерфейси не є оптимальним інструментом для використання непідготовленою людиною.

Для чого використовувати MEEP непідготовленій людині? Пакет MEEP це прекрасний інструмент для навчання майбутніх інженерів, оскільки сьогодні технології для симуляції широко використовуються у реальних компаніях та процесах. І знання такого пакету допоможуть швидше та ефективніше пристосовуватись до інших пакетів або програм. Також такий пакет може використовуватися у наукових роботах і студенти зможуть допомагати керівникам не маючи всіх необхідних навичок для використання програмного інтерфейсу додатку пакету MEEP.

Щоб отримати подібний застосунок потрібно розробити дві його частини: серверний додаток який буде моделювати та графічний інтерфейс, що буде приймати введені величини від користувача.

2.2 Серверний додаток

Для того щоб реалізувати пакет MEEP для серверного використання був використана бібліотека Flask. Flask – це мікровеб-фреймворк, написаний на Python. Його класифікують як мікрофреймворк, оскільки він не потребує спеціальних інструментів чи бібліотек. У ньому немає рівня абстракції бази даних, перевірки форми або будь-яких інших компонентів, де вже існуючі бібліотеки сторонніх розробників забезпечують загальні функції. Однак Flask підтримує розширення, які можуть додавати функції додатків так, ніби вони реалізовані в самому Flask. Існують розширення для об’єктно-реляційних картографів, перевірки форм, обробки завантажень, різних відкритих технологій автентифікації та кількох поширених інструментів, пов’язаних із

фреймворком. Програми, які використовують фреймворк Flask, включають Pinterest і LinkedIn. Файл починається з імпорту необхідних бібліотек (рис. 2.2).

```
from __future__ import annotations
from os import path
from definitions import root_dir
from typing import Union
import meep as mp
from app.meep_api.models.image_transformer import ImageTransformer
from io import BytesIO
from matplotlib import pyplot, figure
from flask import json
```

Рисунок 2.2 – Імпорт бібліотек [12]

Flask пропонує, але не застосовує будь-яких залежностей та макетів проекту. Розробник повинен вибрати інструменти та бібліотеки, які вони хочуть використовувати. Спільнота пропонує багато розширень, які полегшують додавання нових функціональних можливостей.

Для подальшого використання та уніфікації функцій хвилеводів був використаний такий тип даних як об’єкт. Також були задані параметри об’єкта для зберігання та обробки отриманих від користувача. За допомогою метода __init__ (рис. 2.3) задали необхідні параметри.

```
class Waveguide:
        root_dir: str
        dir_out: str
        colormap: str
        sim_data: dict

        def __init__(self, waveguide_type: str, preview: bool = False):
                if preview:
                        waveguide_type = "preview/"+waveguide_type

                self.root_dir = root_dir
                self.dir_out = 'mobile-meep-out/' + waveguide_type
                self.colormap = path.join(self.root_dir, 'app', 'meep_api', 'static', 'colormaps', 'dkbluered')
                self.sim_data = {}
```

Рисунок 2.3 – Оголошення класу “Хвилевод” та необхідних змінних [12]

Метод __init__ схожий на конструктори в C ++ та Java. Конструктори використовуються для ініціалізації стану об'єкта. Завдання конструкторів – ініціалізувати (призначити значення) членам даних класу, коли створюється об’єкт класу.

Наступним кроком були реалізовані функції задання параметрів комірки та їх отримання (рис. 2.4).

```
def set_cell(self, args: dict):
        self.sim_data['cell'] = mp.Vector3(args['x'], args['y'], args['z'])

def get_cell(self) -> object:
        cell = self.sim_data['cell']

        return cell
```

Рисунок 2.4 – Функції для задання та отримання розмірів комірки [12]

Об'єкт Vector3 зберігає розмір комірки в кожному з трьох координатних напрямків. Це 2*d* комірка *z*, *x* та *y*, де напрямок *z* має розмір 0.

Найчастіше структура пристрою задається набором GeometricObjects, що зберігається в об’єкті геометрії. Найчастіше структура пристрою задається набором GeometricObjects, що зберігається в об’єкті геометрії (рис. 2.5).

В наведеній функції pml_layers – це набір об'єктів PML. Може бути більше одного об'єкта PML, щоб шари PML були лише на певних сторонах комірки, наприклад. mp.PML (товщина = 1,0, напрям = mp.X, сторона = mp.high) задає шар PML лише на стороні + *x* (рис. 2.6). Важливий момент: шар PML знаходиться всередині комірки, перекриваючи всі об’єкти, які є.

```
def set_geometry(self, args: dict):
        if args['coordinates']['z'] == 0:
                args['coordinates']['z'] = mp.inf

        self.sim_data['geometry'] = [
                mp.Block(
                        mp.Vector3(
                                args['coordinates']['x'],
                                args['coordinates']['y'],
                                args['coordinates']['z']
                        ),
                        center=mp.Vector3(
                                args['center']['x'],
                                args['center']['y']
                        ),
                        material=mp.Medium(
                                epsilon=args['material']
                        )
                )
        ]
```

Рисунок 2.5 – Функція для задання хвилеводу та його параметрів [12]

```
def set_layers(self, args: dict):
        if not args:
                args = 1.0

        self.sim_data['pml_layers'] = [mp.PML(args)]
```

Рисунок 2.6 – Функція для задання ідеально підібраних шарів (PML) [12]

Приймаємо частоту джерела 0,15 і вказали ContinuousSource, який є просто синусоїдом з фіксованою частотою ($-i\omega t$), який за замовчуванням включається при $t = 0$ (рис. 2.7).

```
def set_sources(self, args: dict):
        if not args:
                args['frequency'] = 0.15
                args['center'] = {
                        "x": -7,
                        "y": 0
                }

        self.sim_data['sources'] = [mp.Source(
                mp.ContinuousSource(frequency=args['frequency']),
                component=mp.Ez,
                center=mp.Vector3(args['center']['x'], args['center']['y']))
        ]
```

Рисунок 2.7 – Функція для задання джерела випромінювання [12]

Кінцевим об'єктом, який потрібно вказати, є Simulation (Симулювання), яке базується на всіх раніше визначених об'єктах (рис. 2.8).

```
def simulate(self, data: dict) -> mp.Simulation:
        simulation = mp.Simulation(
                cell_size=data['cell'],
                boundary_layers=data['pml_layers'],
                geometry=data['geometry'],
                sources=data['sources'],
                resolution=data['resolution'])

        return simulation
```

Рисунок 2.8 – Функція для симуляції [12]

MEEP дискретизує цю структуру в просторі та часі, і це визначається однією змінною, роздільною здатністю, яка дає кількість пікселів на одиницю відстані (рис. 2.9). Приймаємо цю роздільну здатність в 10 пікселів/мкм, що відповідає приблизно 67 пікселів/довжина хвилі, або приблизно

20 пікселів/довжина хвилі у матеріалі з високим індексом. Загалом, принаймні 8 пікселів/довжина хвилі у найвищому діелектрику – хороша ідея. Це відповідає розміру 160 × 80 комірок.

```
def set_resolution(self, args: dict):
        if not args:
                args = 10

        self.sim_data['resolution'] = args
```

Рисунок 2.9 – Функція для задання роздільної здатності [12]

Також була реалізована функція генерації попереднього зображення комірки та хвилевода (рис. 2.10).

```
def preview_figure(self, simulation: mp.Simulation, waveguide: Waveguide, ) -> figure.Figure:
        eps_data = simulation.get_array(center=mp.Vector3(), size=waveguide.get_cell(), component=mp.Dielectric)
        pyplot.figure()
        pyplot.imshow(eps_data.transpose(), interpolation='spline36', cmap='binary')
        pyplot.axis('off')
        fig_electric = pyplot.gcf()

        return fig_electric

def preview_output(self, fig_electric: figure.Figure):
        file = open(self.dir_out+'/preview.png', 'wb+')
        buffer = BytesIO()
        fig_electric.savefig(buffer, format='png')
        buffer.seek(0)
        file.write(buffer.read())
        buffer.close()
        file.close()

        return self.dir_out+'/preview/preview.png'
```

Рисунок 2.10 – Функції для створення попереднього зображення комірки та хвилеводу [12]

Замість запуску output_efield_z лише в кінці моделювання, однак, вона запускається кожні 0,6 одиниці часу (приблизно в 10 разів за період) через mp.at_every (0,6, mp.output_efield_z). Сам по собі це виводить окремий файл на кожен різний час виводу, але замість цього буде використовуватися інша функція для виведення в один 3*d*-файл HDF5, де третій вимір – час. "ez" визначає ім'я вихідного файлу, який буде називатися ez.h5, якщо програма працює в інтерактивному режимі або буде встановлено префікс з ім'ям імені файлу Python (наприклад, tutorial-ez.h5 для tutorial.py).

Imageio – це бібліотека Python, яка забезпечує простий інтерфейс для читання та запису широкого спектру даних про зображення, включаючи анімовані зображення, об'ємні дані та наукові формати. Це крос платформна платформа, працює на Python 3.5+ і легко встановлюється. Вона була використана для отримання кінцевого зображення у GIF форматі. Програма генерує сотні зображень які перетворюються у GIF зображення завдяки Imageio (рис. 2.11).

```
def output(self, simulation: mp.Simulation, each: Union[int, float], until: Union[int, float]):
        simulation.use_output_directory(self.dir_out)

        simulation.run(mp.at_every(each, mp.output_png(mp.Ez, "-Zc" + self.colormap)), until=until)

def image_transform(self, duration: Union[int, float]):
        image_transformer = ImageTransformer(self.dir_out)
        image_transformer.png_to_gif(duration)
```

Рисунок 2.11 – Функції для отримання кінцевого зображення у GIF форматі [12]

Об’єкт StraightWaveguide (Прямий Хвилевод) використовує усі вище наведені функції та генерує зображення у форматі GIF (рис. 2.12).

```python
@waveguide_api.route('/straight-waveguide')
class StraightWaveguide(Resource):
        @waveguide_api.doc('Returns test text and computes of e/m wave propagation in straight waveguide')
        @waveguide_api.expect(waveguide_model)
        @waveguide_api.param('waveguide_type', 'Describing waveguide type selected at the beginning')
        def post(self):
                parser = reqparse.RequestParser()
                parser.add_argument('data', type=dict)
                parser.add_argument('waveguide_type', type=str)
                args = parser.parse_args()

                waveguide_type = args['waveguide_type']
                del args['waveguide_type']

                data = args['data']

                waveguide = Waveguide(waveguide_type, False)

                waveguide.set_cell(data['cell'])
                waveguide.set_geometry(data['geometry'])
                waveguide.set_sources(data['sources'])
                waveguide.set_layers(data['pml_layers'])
                waveguide.set_resolution(data['resolution'])

                sim = waveguide.simulate(waveguide.sim_data)
                waveguide.output(sim, 0.7, 500)
                waveguide.image_transform(0.7)

                folder = 'mobile-meep-out/'+waveguide_type
                for filename in os.listdir(folder):
                        file_path = os.path.join(folder, filename)
                        try:
                                if os.path.isfile(file_path) or os.path.islink(file_path):
                                        os.unlink(file_path)
                                elif os.path.isdir(file_path):
                                        shutil.rmtree(file_path)
                        except Exception as e:
                                print('Failed to delete %s. Reason: %s' % (file_path, e))

                return jsonify(
                        electric='mobile-meep-out/'+waveguide_type+'-movie.gif'
                )
```

Рисунок 2.12 – Застосування усіх вище приведених функцій для генерування [12]

Також була реалізована можливість згенерувати попереднє зображення хвилеводу для перевірки ідеї щодо розмірів комірки та хвилеводу (рис. 2.13).

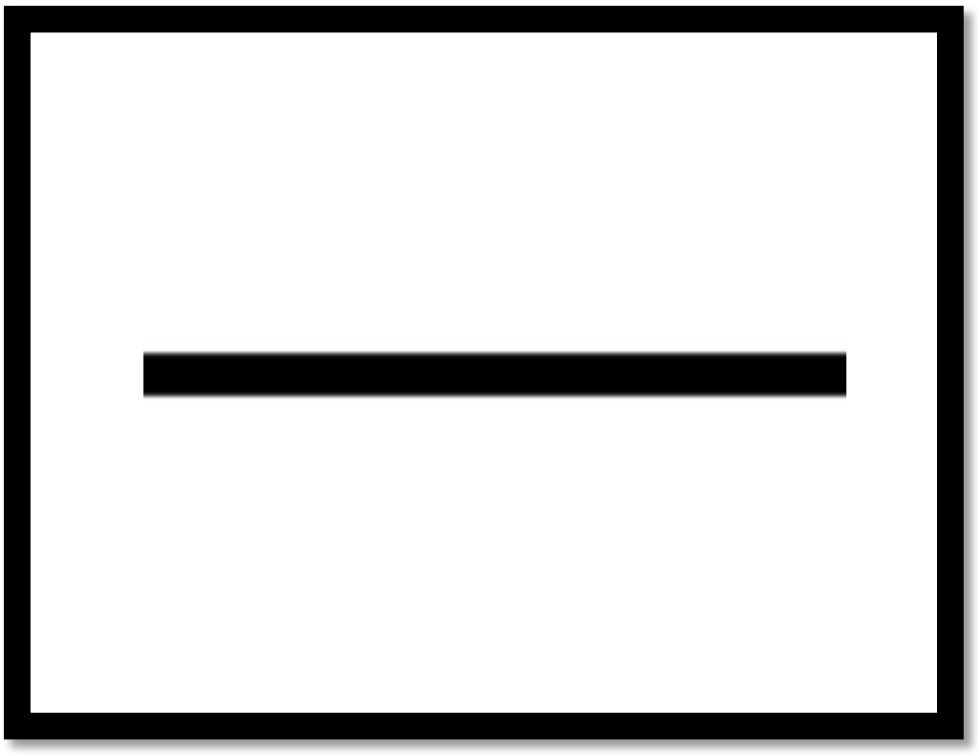


Рисунок 2.13 – Приклад попереднього зображення комірки та хвилеводу [12]

Серверний додаток приймає дані у форматі JSON, який має певну структуру (рис. 2.14).


```
{
  "coordinates": {
    "x": 17,
    "y": 1,
    "z": 0
  },
  "center": {
    "x": 0,
    "y": 0
  },
  "material": 12
}
```


Рисунок 2.14 – Приклад інформації для генерації попереднього зображення комірки та хвилеводу [12]

Дані для генерації кінцевого зображення у форматі GIF також приймаються у форматі JSON, проте розмір більший ніж у даних для генерації попереднього зображення адже задається джерело та роздільна здатність (рис. 2.15).

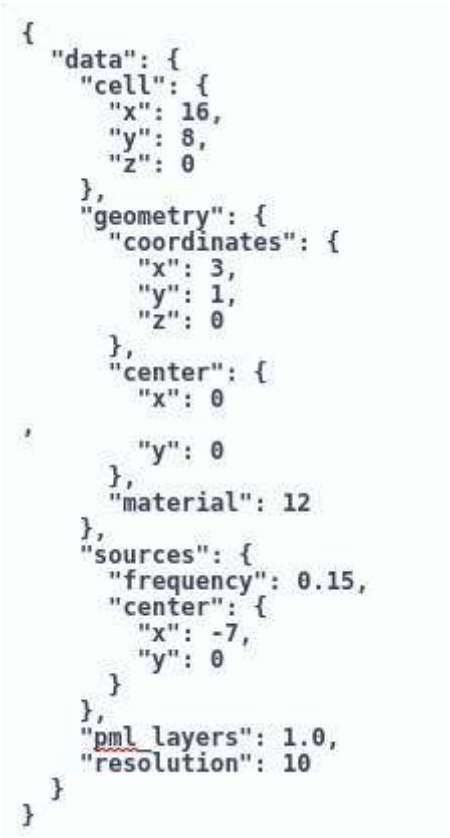


Рисунок 2.15 – Приклад інформації для генерації кінцевого зображення симуляції [12]

2.3 Графічний інтерфейс

Серверний додаток є частиною яка оброблює дані які передає йому користувач. Проте для полегшення користування програмним забезпеченням потрібно розробити графічний інтерфейс. Він слугуватиме для передачі даних до серверного додатку. Його задача – бути максимально простим, адже важкі інтерфейси часто заважають помітити важливі елементи вводу (рис. 2.16).

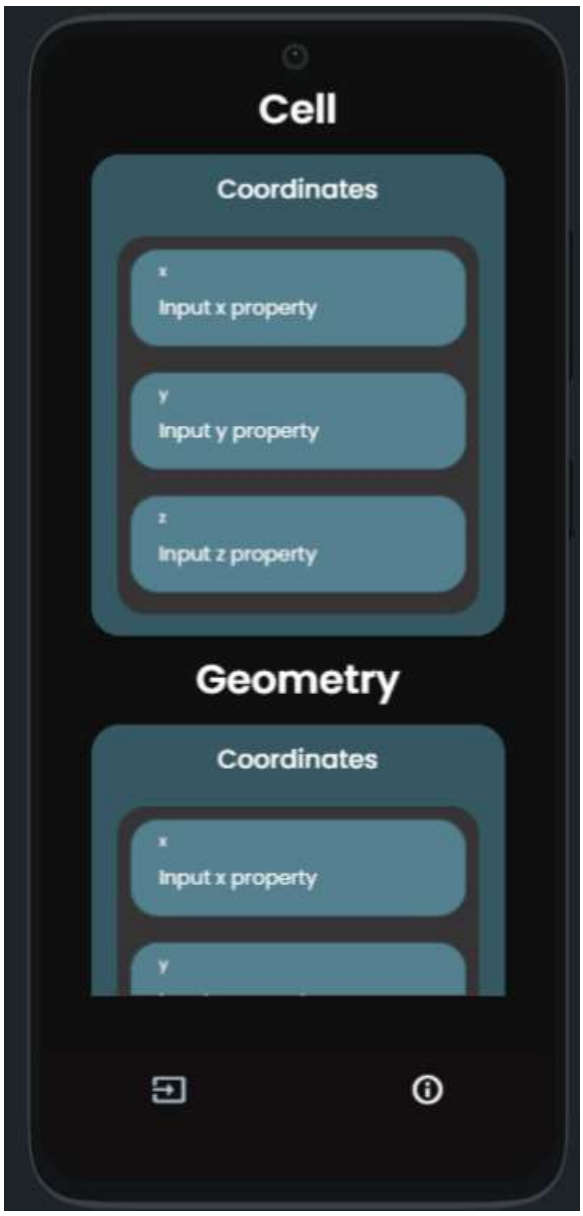


Рисунок 2.16 – Графічний додаток для роботі з пакетом MEEP [13]

Графічний інтерфейс у вигляді додатку для сучасних смартфонів був розроблений за допомогою мови програмування Dart та SDK Flutter. Це один із найсучасніших та прогресивних інструментів для розробки графічних інтерфейсів для будь якої платформи.

Dart — це мова програмування, розроблена для клієнтської розробки [14], наприклад для веб-додатків і мобільних додатків. Він розроблений Google і може також використовуватися для створення серверних і настільних програм. Це об’єктно-орієнтована, заснована на класах мова зі збиранням сміття з

синтаксисом C [15]. Він може компілювати або машинний код, або JavaScript, а також підтримує інтерфейси, міксини, абстрактні класи, генерики.

Flutter — це набір програмного забезпечення для розробки інтерфейсу користувача з відкритим кодом, створений Google. Він використовується для розробки крос-платформних додатків для Android, iOS, Linux, macOS, Windows, Google Fuchsia та браузеру з однієї кодової бази [16]. Вперше описаний у 2015 році Flutter був випущений у травні 2017 року [17].

Програми Flutter написані мовою Dart і використовують багато розширених функцій мови. Під час написання та налагодження програми Flutter працює у віртуальній машині Dart, яка має механізм виконання JIT. Це забезпечує швидкий час компіляції, а також «гаряче перезавантаження», за допомогою якого модифікації вихідних файлів можна вводити у запущену програму. Flutter розширює це завдяки підтримці гарячого перезавантаження з урахуванням стану, коли в більшості випадків зміни у вихідному коді негайно відображаються у запущеній програмі без необхідності перезапуску чи втрати стану [18]. Для кращої продуктивності версії програм Flutter на всіх платформах використовують компіляцію наперед (AOT), за винятком веб-версій, де код транспільується в JavaScrip [18]. Flutter успадковує менеджер пакунків Dart Pub і репозиторій програмного забезпечення, що дозволяє користувачам публікувати та використовувати спеціальні пакети, а також спеціальні плагіни Flutter [19].

Файл для графічного додатку також стартує з імпортів необхідних бібліотек (рис. 2.17). Вони включають декілька компонентів та базові шрифти із темою.

Усі основні елементи для побудування графічного інтерфейсу у SDK Flutter називаються Widget (рис. 2.18). Будь що з того що ви бачите на екрані являться віджетомопере. Їх відносини між собою вибудовуються у дерево, де кожен елемент пов'язаний із пднім через відносини “батько-дитина”. Таким чином уся сторінка або компонент зрозумілі та легко модифікуються.

```dart
import '../components/three_axis_input_widget.dart';
import '../components/two_axis_input_widget.dart';
import '../flutter_flow/flutter_flow_theme.dart';
import '../flutter_flow/flutter_flow_util.dart';
import '../flutter_flow/flutter_flow_widgets.dart';
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
```

Рисунок 2.17 – Імпорти бібліотек для графічного додатку [13]

```dart
class ThreeAxisInputWidget extends StatefulWidget {
  const ThreeAxisInputWidget({Key? key}) : super(key: key);

  @override
  _ThreeAxisInputWidgetState createState() => _ThreeAxisInputWidgetState();
}

class _ThreeAxisInputWidgetState extends State<ThreeAxisInputWidget> {
```

Рисунок 2.18 – Декларування нового віджету [13]

Така побудова додатку дає гнучкість і легкість розробки. Це в свою чергу дозволяє вносити необхідні зміни або добудовувати додаток згідно з утверджених нових ідей. Такі дерева та відносини “батько-дитина” сприяють полегшенню розуміння роботи та поріг входу для починаючих (рис. 2.19).

Віджети Flutter створено з використанням сучасної структури, яка черпає натхнення з React. Основна ідея полягає в тому, що ви створюєте свій інтерфейс користувача з віджетів. Віджети описують, як має виглядати їхнє подання з огляду на поточну конфігурацію та стан. Коли стан віджета змінюється, віджет перебудовує свій опис, який фреймворк відрізняє від попереднього опису, щоб визначити мінімальні зміни, необхідні в базовому дереві візуалізації для переходу з одного стану в інший.

```
Widget build(BuildContext context) {
  return Column(
    mainAxisSize: MainAxisSize.max,
    children: [
      Padding(
        padding: EdgeInsetsDirectional.fromSTEB(5, 5, 5, 5),
        child: Card(
          clipBehavior: Clip.antiAliasWithSaveLayer,
          color: Color(0xFF54808F),
          shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(20),
          ),
          child: Padding(
            padding: EdgeInsetsDirectional.fromSTEB(10, 10, 10, 10),
            child: TextFormField(
              controller: textController1,
              obscureText: false,
              decoration: InputDecoration(
                labelText: 'x',
                hintText: 'Input x property',
                enabledBorder: UnderlineInputBorder(
                  borderSide: BorderSide(
                    color: Color(0x00000000),
                    width: 1,
                  ),
                  borderRadius: const BorderRadius.only(
                    topLeft: Radius.circular(4.0),
                    topRight: Radius.circular(4.0),
                  ),
                ),
                focusedBorder: UnderlineInputBorder(
                  borderSide: BorderSide(
                    color: Color(0x00000000),
                    width: 1,
                  ),
```

Рисунок 2.19 – Дерево із відносинами “батько-дитина” [13]

Віджети які відносяться до графічних мають багато налаштувань і дозволяють задавати колір, радіус граней, текст підказок і подібні (рис. 2.20). Це дозволяє гнучко налаштовувати зовнішній вигляд віджету.

```
child: Container(
  width: MediaQuery.of(context).size.width * 0.8,
  height: MediaQuery.of(context).size.height * 0.42,
  decoration: BoxDecoration(
    color: Color(0xFF365862),
    borderRadius: BorderRadius.circular(20),
  ),
```

Рисунок 2.20 – Приклад графічного віджету і його налаштувань [13]

Компоненти – це винесені окремо частинки коду які використовуються там де ці частинки коду краще винести для покращеного повторного використання. Це приводить до кращої чистоти та зрозумілості коду. Компоненти у даному графічному додатку використовуються для полей вхідних даних. У проекті присутні два компоненти: один для вхідних даних трьох осей та інший для вхідних даних двох осей. Оскільки є декілька параметрів, які потребують трьох осей та двох осей створення цих компонентів було цілком виправдане та доцільне. Створення компоненту для одного поля вхідних даних не має сенсу.

Також усі віджети мають дуже гнучкі налаштування поведінки і обробки взаємодії користувача із ними (рис. 2.21). Основний віджет як для компоненту для вхідних даних трьох осей так і для компоненту для вхідних даних двох осей є TextFormField [20]. Цей віджет відповідає за введення даних від користувача, які потім будуть відіслані до серверного додатку. Цей віджет має особливі параметри як для графічного відображення так і для його логіки. Всі поля для введення даних користувачем у цьому проекті є віджетами TextFormField. Значення цього поля вводу не треба зберігати окремою кнопкою, введене значення автоматично зберігається в контроллері для цього віджету.

Фінальним елементом графічного інтерфейсу є віджет який є кнопкою за натиском якої перевіряється наявність усіх необхідних даних і відправка даних до серверного додатку (рис. 2.22).

```
child: TextFormField(
  controller: textController1,
  obscureText: false,
  decoration: InputDecoration(
    labelText: 'x',
    hintText: 'Input x property',
    enabledBorder: UnderlineInputBorder(
      borderSide: BorderSide(
        color: Color(0x00000000),
        width: 1,
      ),
      borderRadius: const BorderRadius.only(
        topLeft: Radius.circular(4.0),
        topRight: Radius.circular(4.0),
      ),
    ),
    focusedBorder: UnderlineInputBorder(
      borderSide: BorderSide(
        color: Color(0x00000000),
        width: 1,
      ),
      borderRadius: const BorderRadius.only(
        topLeft: Radius.circular(4.0),
        topRight: Radius.circular(4.0),
      ),
    ),
    errorBorder: UnderlineInputBorder(
      borderSide: BorderSide(
        color: Color(0x00000000),
        width: 1,
      ),
      borderRadius: const BorderRadius.only(
        topLeft: Radius.circular(4.0),
        topRight: Radius.circular(4.0),
      ),
    ),
```

Рисунок 2.21 – Приклад віджету і його налаштувань [13]

```
FFButtonWidget(
  onPressed: () {
    print('Button pressed ...');
  },
  text: 'Submit',
  options: FFButtonOptions(
    width: 130,
    height: 40,
    color: Color(0xFF54808F),
    textStyle:
        FlutterFlowTheme.of(context).subtitle2.override(
              fontFamily: 'Poppins',
              color: Colors.white,
            ),
    borderSide: BorderSide(
      color: Colors.transparent,
      width: 1,
    ),
    borderRadius: BorderRadius.circular(20),
  ),
),
,
```

Рисунок 2.22 – Приклад віджету кнопки і його налаштувань [13]

2.4 Попередні результати розрахунку

Наведені вище серверний та графічний додаток можуть бути використані для наглядного вирішення безліч задач. Прикладом однієї з таких задач є розрахунок гаусового пучка. В роботі розроблені додатки за допомогою мови Python та графічний за допомогою мови програмування Dart та SDK Flutter. У зв’язці ці додатки дозволяють візуалізувати багато складних структур, таким чином покращуються роботи та розуміння теорії у людей які будуть використовувати ці додатки.

Також ці системи можуть використовуватися для систем контролю характеристик лазерного випромінювання. Це може використовуватися у багатьох виробництвах де застосовуються лазери і буде покращувати і контролювати їх властивості [21, 22].

У пакеті MEEP є можливість моделювання випромінювання гаусового пучка. Для цього використовується вбудований в пакет MEEP підклас класу Source під назвою GaussianBeamSource (рис. 2.23).

```
mp.GaussianBeamSource(
    src=mp.ContinuousSource(fcen),
    center=mp.Vector3(0, -0.5 * s + dpml + 1.0),
    size=mp.Vector3(s),
    beam_x0=beam_x0,
    beam_kdir=beam_kdir,
    beam_w0=beam_w0,
    beam_E0=beam_E0,
)
```

Рисунок 2.23 – Клас GaussianBeamSource та його параметри [23]

Із додаткових параметрів тут наявні параметри які відповідають за характеристики пучка на додачу до параметрів звичайного джерела випромінювання. Однак компонентний параметр об’єкта Source ігнорується.

Гаусів пучок – це поперечна електромагнітна мода, для якої область джерела має бути лінією (у $2d$) або площиною (у $3d$). Для пучка, поляризованого в напрямку x з поширенням уздовж $+z$, електричне поле визначається як:

$$E(r,z) = E_0\hat{x}\frac{\omega_0}{\omega(z)}\exp\left(\frac{-r^2}{\omega(z)^2}\right)\exp\left(-i\left(kz + k\frac{r^2}{2R(z)}\right)\right) \#(2.1)$$

де r – радіальна відстань від центральної осі променя;

z – осьова відстань від фокуса променя (або «талії»);

$k = 2\pi n/\lambda$ – хвильове число (для довжини хвилі вільного простору λ і показника заломлення n однорідного середовища без втрат, в якому поширюється промінь);

E_0 – амплітуда електричного поля в початку координат;

(z) – радіус, при якому амплітуда поля спадає на $1/e$ його осьового значення;

ω_0 – ширина горловини пучка;

$R(z)$ – радіус кривизни хвильового фронту променя при z.

Єдиними незалежними параметрами, які необхідно вказати, є ω_0, E_0, k і розташування фокусу променя (тобто початок: $r = z = 0$).

Але краще розуміння, що і є метою цього проекту, відобразиться у візуальному поданні. Отже для обчислення параметрів гаусівського пучку основними параметрами які треба задати є ширина горловини пучка та кут поширення пучка який потім конвертується в напрямок випромінювання пучка. Гарним прикладом базового гаусівського пучка є симуляція із параметром ширини горловини що дорівнює 0,8 і кутом поширення що дорівнює 0° (рис. 2.24).

Однак якщо змінити параметри то наочність згенерованих пучків допоможе краще зрозуміти те як ці параметри впливають на реальні пучки. Як от, наприклад, зміна ширини пучка до значення в 0,4 вже істотно змінює пучок що і відображається у результаті (рис. 2.25).

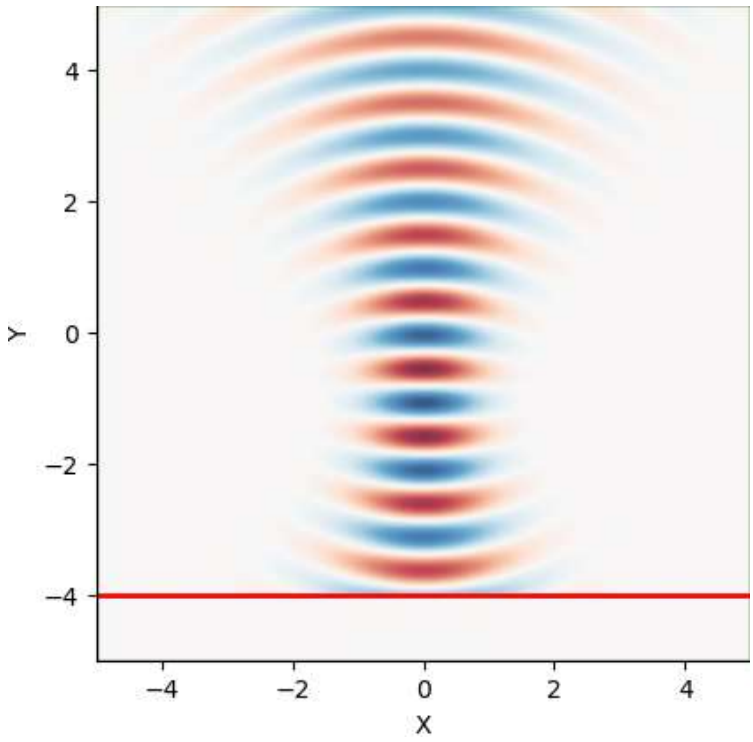


Рисунок 2.24 – Гаусовий пучок за параметрів ширини пучка 0.8 і кутом в 0°

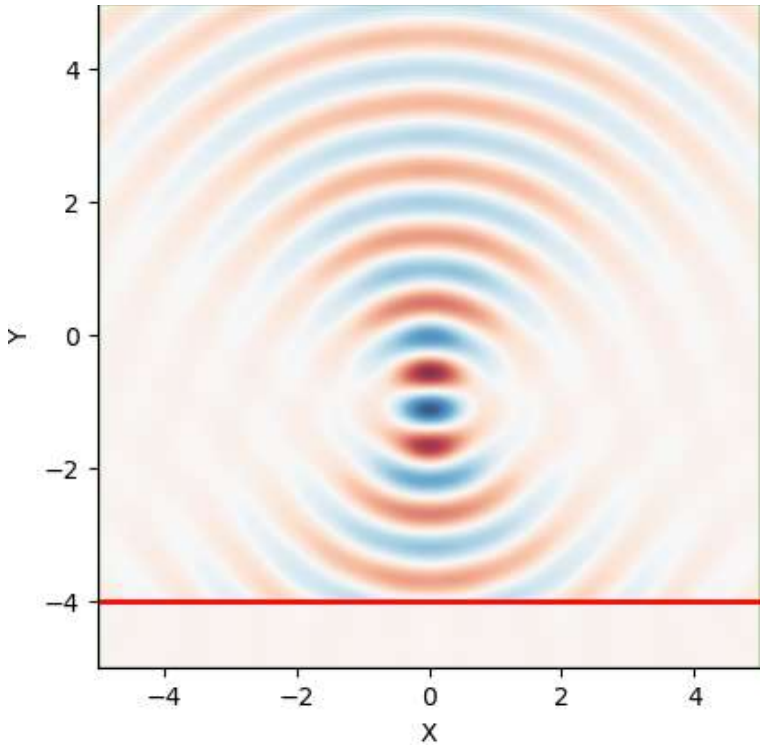


Рисунок 2.25 – Гаусовий пучок за параметрів ширини пучка 0,4 і кутом в 0°

Звісно ж збільшення параметру ширини пучка також відобразиться (рис. 2.26).

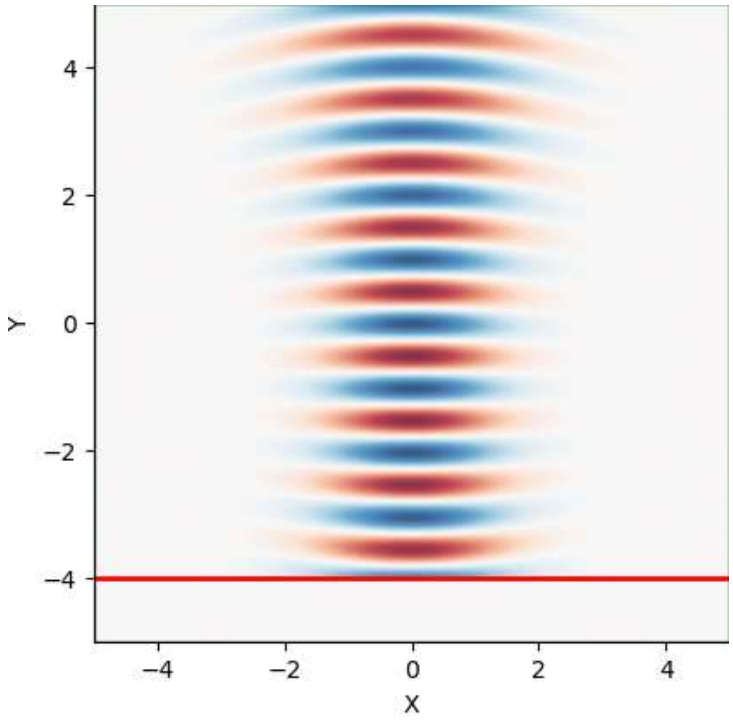


Рисунок 2.26 – Гаусовий пучок за параметрів ширини пучка 1,2 і кутом в 0°

При комбінуванні параметру кута поширення пучка та його ширини картина також змінюється (рис. 2.27). Таким чином можна констатувати що цей інструмент дозволить покращити розуміння та додати до списку навичок наукового співробітника або студента корисний інструмент або дати досвід для використання інших програм.

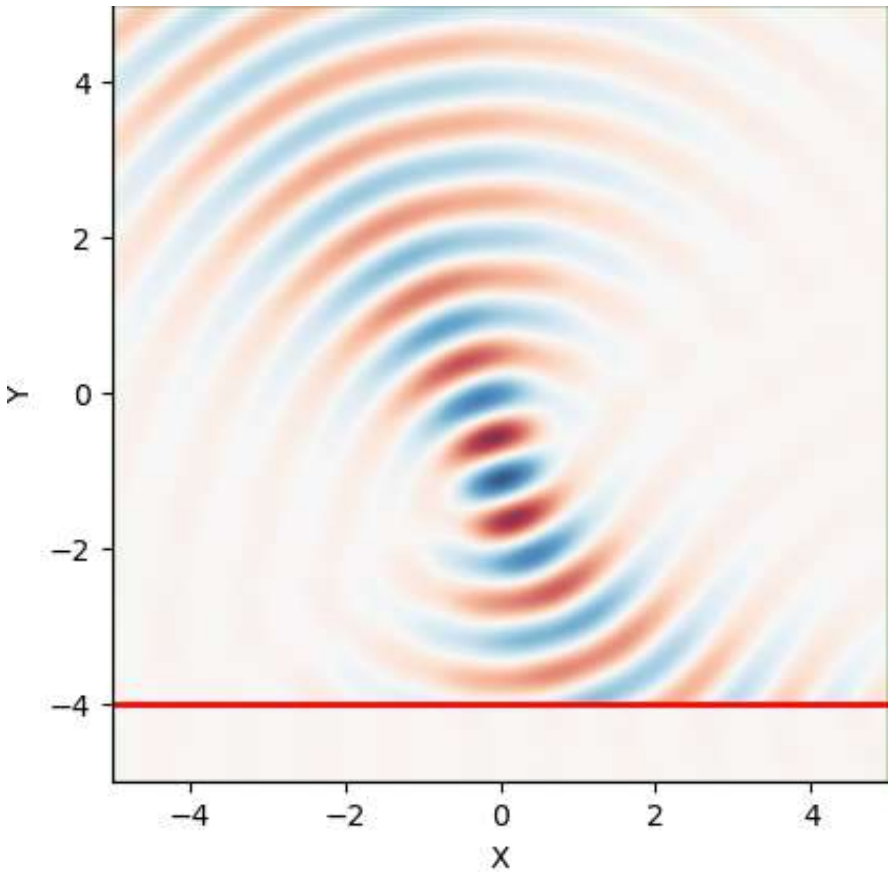


Рисунок 2.27 – Гаусовий пучок за параметрів ширини пучка 1,2 і кутом в 20°

ВИСНОВКИ

Проведений аналітичний огляд наявних додатків для використання пакету MEEP та графічних імплементацій для використання

В кваліфікаційній роботі розглянуті існуючі засоби для симуляції електромагнітних явищ, зокрема метод скінченних різниць в часовій області. Також розглянуто пакет MEEP.

Розроблено серверний додаток для моделювання оптичних хвилеводів в пакеті MEEP за допомогою мови програмування Python та бібліотек MEEP, Imageio, Flask та Flask-restplus. Також розроблено графічний інтерфейс за допомогою мови програмування Dart та Flutter SDK. Наведені приклади розроблених функцій та об’єктів та результати обчислень. Також наведені приклади інформації потрібної для використання програми та отримання результату.

Розроблена тестова модель градієнтної лінзи типу “риб’яче око“ Максвела та простого діелектричного хвилеводу.

На основі тестової моделі проведені розрахунки просторового випромінювання гаусового пучка. Отримані результати добре узгоджуються з відомими, а отже таким чином створені додатки дають змогу отримувати точні результати.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИ ЛАННЯ

1. Sidney A. Focusing of Spherical Gaussian Beams // Applied Optics. 1983. Vol. 22. Issue 5. P. 658–661.

2. Anupam Garg. Classical Electromagnetism in a Nutshell. Princeton, N.J.: Princeton University Press. 1st Edition. 2012, 712 p.

3. Amnon Y. Quantum Electronics. 3rd ed. Wiley, 1991. 676 p.

4. Oskooi A. F. and others. MEEP: A flexible free-software package for electromagnetic simulations by the FDTD method // Computer Physics Communications. 2010. Vol. 181. P. 687–702.

5. Емблема пакету MEEP URL: https://meep.readthedocs.io/en/latest/images/Meep-banner.png (дата звернення 02.06.2021).

6. Приклад розповсюдження змодельованого за допомогою пакету MEEP. URL:URL:https:// encryptedtbn0.gstatic.com/images?q=tbn:ANd9GcTIDolXPUpBB0uGRqSBj9FJZwyVd9Xt8Z_ian9JgYBw8deqvxV5bjDfcx8WVXNZwozRV1g&usqp=CAU (дата звернення 02.06.2021).

7. PML Layers. URL:https://meep.readthedocs.io/en/latest/images/pml_glancing_field.png (дата звернення 03.06.2021).

8. Решітка Йі. URL: https://meep.readthedocs.io/en/latest/images/Yee-cube.png (дата звернення 03.06.2021).

9. Моделювання фотонного кристалу за допомогою FDTD. URL: https://www.google.com/url?sa=i&url=http%3A%2F%2Fab-initio.mit.edu%2F~oskooi%2Fpapers%2FOskooi10.pdf&psig=AOvVaw38wS4EkMx2ybyvAj1eGAfS&ust=1621547612475000&source=images&cd=vfe&ved=0CAIQjRxqFwoTCKCc9uHd1vACFQAAAAAdAAAAABAK (дата звернення 02.06.2021)

10. A coupled electromagnetic-mechanical-acoustic model of a DC electric motor by Furlan, Martin; Cernigoj, Andrej and Boltežar, Miha. URL: http://www.cad-cam-cae.com/blog/Furlan03.jpg (дата звернення 04.06.2021).

11. Зображення інтерфейсу пакета MEEP з сайту Nanohub. URL: https://nanohub.org/tools/meep/session (дата звернення 05.06.2021).

12. Mobile-meep-api. URL: https://github.com/subwayaddicted/mobile-meep-api (дата звернення 04.06.2021).

13. Mobile MEEP application. URL:https://app.flutterflow.io/preview/mobile-m-e-e-p-kydlnr (дата звернення 07.09.2022).

14. Dart overview. URL: dart.dev (дата звернення 06.10.2022).

15. A Tour of the Dart Language. URL: dart.dev (дата звернення 06.10.2022).

16. Amadeo R. Google starts a push for cross-platform app development with Flutter SDK // Ars Technica. Retrieved 2021-06-11. URL:https://app.flutterflow.io/preview/mobile-m-e-e-p-kydlnr (дата звернення 15.10.2022).

17. Flutter W. Google Aims Dart to Mobile App Cross-Development. URL: https://www.infoq.com/news/2015/12/flutter-dart-cross-platform/ (дата звернення 06.10.2022).

18. Building a web application with Flutter. URL: https: // docs.flutter.dev (дата звернення 06.10.2022).

19. Using packages. URL: https: // docs.flutter.dev (дата звернення 06.10.2022).

20. TextFormField class. URL: https://api.flutter.dev/flutter/material/TextFormField-class.html (дата звернення 06.11.2022).

21. Odarenko E. N., Svich V. A.; Smat'ko A. A. Transformation of Gaussian beam polarization by metamaterial layer: 19th International Crimean Conference “Microwave & Telecommunication Technology. 14–18 September 2009. Sevastopol. Ukraine.

22. Odarenko E. N., Svich V. A., Smat'ko A. A. Wave Beam Scattering by Thin Lossy Dielectric Cylinder:// 17th International Crimean Conference “Microwave & Telecommunication Technology”. 10–14 September 2007. Sevastopol. Ukraine.

23. GaussianBeamSource class. URL: https://github.com/NanoComp/meep/blob/master/python/examples/gaussian-beam.py (дата звернення 06.11.2022).