КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет радіофізики, електроніки та комп'ютерних систем

Кафедра комп'ютерної інженерії

Системи та методи візуалізації фізичних розрахунків на прикладі дисперсії прямих об'ємних магнітостатичних хвиль в ізольованому феритовому шарі

Дипломна робота бакалавра

студента 4 року навчання

Спеціальність: 123 «Комп'ютерна інженерія»

Ігора РАКИТІНА

Науковий керівник

канд. фіз.-мат. наук Олексій НЕЧИПОРУК,

доцент кафедри квантової радіофізики

та наноелектроніки

Рецензент

Канд. фіз.-мат. наук, Владислав МОЙСЕЄНКО

зав. навч. лаб. радіоелектроніки

До захисту допускаю: Завідувач кафедрою

Юрій БОЙКО

Ухвалено на засіданні кафедри "____" _____2022 р., протокол № ____

Київ 2022

РЕФЕРАТ

Дипломна робота бакалавра: 36 с., 9 рис., 1 додаток, 26 джерел.

Досліджується дисперсія прямих об’ємних магнітостатичних хвиль (ПОМСХ) в ізольованому феритовому шарі. Для аналізу використовується метод комп’ютерного моделювання. Отримано та проаналізовано відповідне дисперсійне співвідношення. Запропоновано програму для моделювання дисперсії прямих об’ємних магнітостатичних хвиль при перпендикулярному куті поширення в ізольованому шарі фериту. Наводяться приклади моделювання дисперсії хвиль в тривимірному просторі хвильових чисел та частот.

КЛЮЧОВІ СЛОВА: хвилі магнітостатичні, дисперсія, комп’ютерне моделювання.

Зміст

Вступ

Сучасний етап технологічного прогресу вимагає виготовлення складних електронних пристроїв, які потребують надійності, економічності та розумної вартості. Збільшення функціональної та структурної складності пристрою пов’язане з підвищенням інтеграції мікросхем. Тим часом принцип створення переходів і блоків у вигляді схем із стандартних радіодеталей (транзисторів, діодів, конденсаторів тощо) існує досі. Однак збільшення інтеграції та пов’язане з цим зменшення розміру компонента мають певні фізичні обмеження. Інтеграція сотень тисяч елементів на кристалі є технічно складним процесом.

Функціональна електроніка пропонує абсолютно новий підхід, що дозволяє обробляти електромагнітні сигнали без використання стандартних електронних компонентів, безпосередньо на основі фізичних явищ у твердих тілах. У цьому випадку сама сутність має властивості, необхідні для виконання певної функції.

У зв’язку з цим дослідники зосередилися на магнітостатичних хвилях (МСХ) у монокристалах фериту. Серед усіх основних типів збудження твердих тіл, МСХ мають унікальну різноманітність лінійних і нелінійних властивостей, особливо дисперсійних властивостей, якими можна керувати різними методами. Крім того, МСК мають найнижчі втрати на перетворення та поширення в мікрохвильовому діапазоні, особливо в феритових середовищах, таких як залізо-ітрієвий гранат (ZIG) Y3Fe5O12. Все це стимулювало формування нового напряму функціональної електроніки - спін-хвильової електроніки, що дає можливість в реальному часі реалізувати планарні пристрої та системи обробки сигналів безпосередньо на несучій частоті (на відміну від схем і цифрових рішень). Властивості магнітостатичних хвиль дозволяють створювати на їх основі різноманітні НВЧ електронні пристрої (фільтри, лінії затримки, обмежувачі потужності) з широким діапазоном параметрів.

Сучасний рівень розвитку функціональної електроніки потребує дослідження характеристик МСХ в феритових структурах. Виходячи з цього в даній роботі поставлена мета дослідити дисперсію ПОМСХ у феритовому прошарку, а також провести її моделювання в тривимірному просторі частот та хвильових чисел.

1. ПРЯМІ ОБ'ЄМНІ МСХ

Перше дослідження магнітостатичного наближення прямих об’ємних магнітостатичних хвиль у нескінченних феромагнітних пластинах було проведено Баряхтаром і Кагановим у 1961 р. [3], опубліковано одночасно з роботою Деймона та Ашбаха для дотично намагніченого феритового шару.

Об'ємними називаються хвилі, для яких залежність змінної намагніченості та складових електромагнітного поля від координати напрямку, перпендикулярному поверхні плівки, є тригонометричної. Для поверхневих хвиль ця залежність є експоненційної – амплітуди змінної намагніченості та складових поля зменшуються при віддаленні однієї з поверхонь плівки. Від якої саме, залежить від напрямку намагнічування та напрямку поширення хвилі. Прямими називаються хвилі, для яких напрямок груповий швидкості (тобто швидкості розповсюдження сигналу та перенесення енергії) $\partial\omega/\partial k$ збігається з напрямком фазової швидкості ω/k.

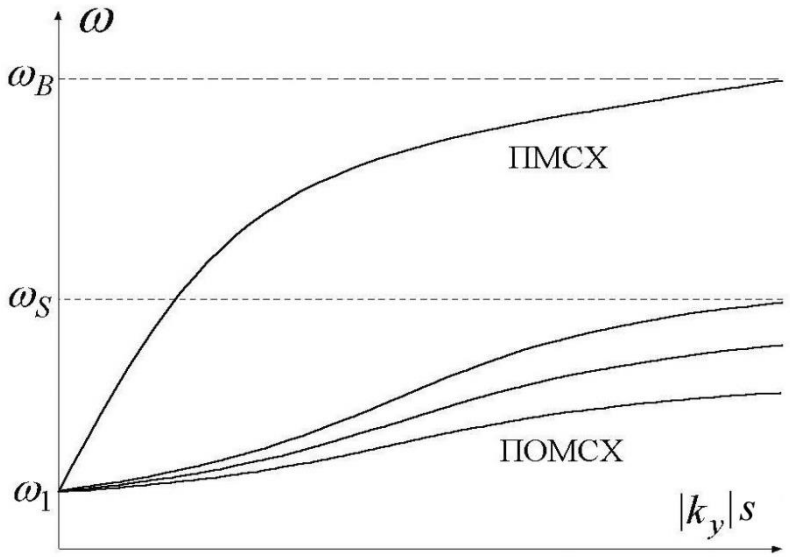

Рис.1.1. Дисперсія МСХ, які поширюються в феритовому шарі

Прямі об'ємні магнітостатичні хвилі (ПОМСХ). На відміну від ПМСХ (рис. 1.1) що поширюються у дотично намагнічений ЕФП, ПОМСХ збуджуються й поширюються в плівці при її намагнічуванні близькому до нормального.

В геометрії, яка має обертальну симетрію відносно вісі Z, зручно без обмеження загальності розглядати хвилі, що поширюються вздовж вісі Y(див. рис.1.2). Такі хвилі називають прямими об'ємними МСХ (ПОМСХ).

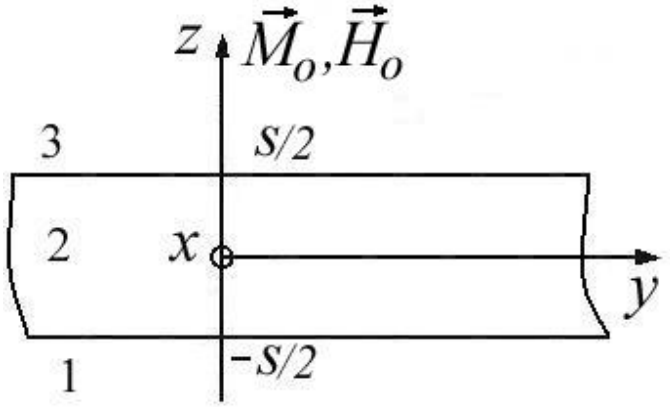

Рис.1.2 Нормально намагнічений феритовий шар.

Вважаємо, що магнітні властивості діелектричних шарів описуються тензором:

$$\mu_{ij} = \delta_{ij}\text{ , (}\delta_{ij}\text{ - символ Кронеккера).}$$

Система рівнянь, що описує МСХ в нашій структурі:

$$\begin{cases} \dfrac{\partial^2\Psi_1}{\partial y^2} + \dfrac{\partial^2\Psi_1}{\partial z^2} = 0, & z < -\dfrac{s}{2} \\ \mu\dfrac{\partial^2\Psi_2}{\partial y^2} + \dfrac{\partial^2\Psi_2}{\partial z^2} = 0, & |z| < \dfrac{s}{2} \\ \dfrac{\partial^2\Psi_3}{\partial y^2} + \dfrac{\partial^2\Psi_3}{\partial z^{92}} = 0, & z > \dfrac{s}{2} \end{cases} \qquad (1.1)$$

Вважаючи, що розв'язок має вигляд плоскої хвилі, тобто $\Psi_i = \mathrm{Z}_i(z)e^{i(\omega t - ky)}$, з системи (1.1) отримуємо наступні рівняння:

$$\begin{cases} \dfrac{\partial^2 \mathrm{Z}_1}{\partial z^2} - k^2 \mathrm{Z}_1 = 0, & z < -\dfrac{s}{2} \quad (1.2) \\ \dfrac{\partial^2 \mathrm{Z}_2}{\partial z^2} - \mu k^2 \mathrm{Z}_2 = 0, & |z| < \dfrac{s}{2} \quad (1.3) \\ \dfrac{\partial^2 \mathrm{Z}_3}{\partial z^2} - k^2 \mathrm{Z}_3 = 0, & z > \dfrac{s}{2} \quad (1.4) \end{cases}$$

Зрозуміло, що характер розв'язку цих рівнянь буде визначатися знаком μ. Докладніше в роботі [4]

В нормально намагніченому шарі відповідні магнітостатичні потенціали записуються, як:

$$\Psi_1 = Ae^{|k|z+i(\omega t-ky)} \quad (1.5)$$

$$\Psi_2 = \left(B\cos(-\mu)^{1/2}|k|z + C\sin(-\mu)^{1/2}|k|z\right)e^{i(\omega t-ky)} \quad (1.6)$$

$$\Psi_3 = De^{-|k|z+i(\omega t-ky)} \quad (1.7)$$

Для отримання коефіцієнта дисперсії магнітостатичної хвилі використовуємо граничні умови: безперервність тангенціальної складової напруженості магнітного поля і нормальної складової магнітної індукції на межі середовища, і отримуємо відповідні вирази:

$$\Psi_1|_{z=0} = \Psi_2|_{z=0}, \qquad \left.\frac{\partial \Psi_1}{\partial x}\right|_{z=0} = \left.\frac{\partial \Psi_2}{\partial x}\right|_{z=0}$$

$$\Psi_2|_{z=s} = \Psi_3|_{z=s}, \qquad \left.\frac{\partial \Psi_2}{\partial x}\right|_{z=s} = \left.\frac{\partial \Psi_3}{\partial x}\right|_{z=s}$$

Граничні умови приводять до наступного дисперсійного співвідношення для ПОМСХ:

$$\mathrm{tg}|k|\,s(-\mu)^{1/2} = -\frac{2\sqrt{-\mu}}{1-\mu} \tag{1.8}$$

Використовуючи формулу котангенсу подвійного аргументу, запишемо дисперсійне рівняння у вигляді:

$$\left(\mathrm{tg}\frac{|k|s\sqrt{-\mu}}{2} - \frac{1}{\sqrt{-\mu}}\right)\left(\mathrm{tg}\frac{|k|s\sqrt{-\mu}}{2} + \sqrt{-\mu}\right) = 0 \tag{1.9}$$

Воно розбивається на два незалежних дисперсійних співвідношення:

$$\mathrm{tg}\frac{|k|s\sqrt{-\mu}}{2} = \frac{1}{\sqrt{-\mu}}, \tag{1.10}$$

$$\mathrm{tg}\frac{|k|s\sqrt{-\mu}}{2} = -\sqrt{-\mu}, \tag{1.11}$$

Корені рівнянь можна визначити наступним чином:

$$|k_n|s = \frac{2}{\sqrt{-\mu}}\left(n\pi + \mathrm{arctg}\frac{1}{\sqrt{-\mu}}\right), \quad n = 0,1,2\ldots\ , \tag{1.12}$$

$$|k_m|s = \frac{2}{\sqrt{-\mu}}\left(m\pi + \mathrm{arctg}\sqrt{-\mu}\right), \quad m = 0,1,2\ldots\ , \tag{1.13}$$

Спектр ПОМСХ, що визначається дисперсійними співвідношеннями (1.12), (1.13) показано на рис.1.3. Хвилі, дисперсія яких описується виразами (1.10),(1.12), є симетричними, а хвилі з дисперсією (1.11), (1.13) – антисиметричними.

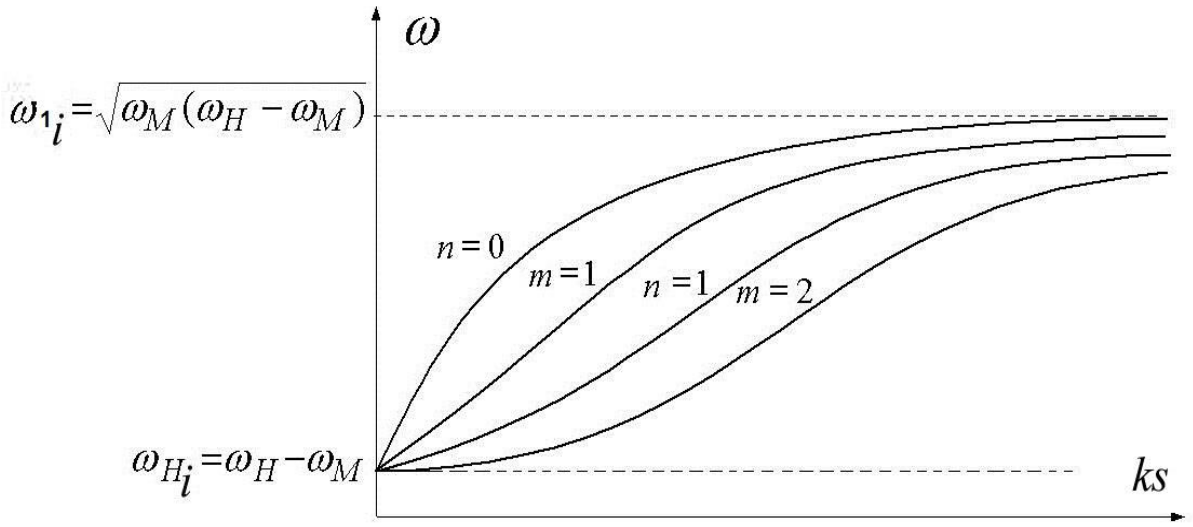

Рис. 1.3. Дисперсія прямих об’ємних МСХ в нормально намагніченому феритовому шарі

Основною (фундаментальною) модою ПОМСХ є симетрична мода з $n = 0$, яка є найвищою за частотою.

Зробимо тут важливе зауваження. Оскільки вектор статичної намагніченості $\vec{M}$ спрямовано перпендикулярно до площини шару, необхідно зробити заміну, яка б враховувала поле розмагнічування, а саме $\overrightarrow{H_0} > 4\pi\vec{M}$

Підкреслимо також, що необхідною є умова $\omega_H - \omega_M > 0$, інакше зразок буде знаходитись в ненасиченому стані і в ньому буде існувати доменна структура.

Діапазон існування хвиль $\Delta\omega$ прямує до 2400 МГц при Т=300 К для ЗІГ:

$$\Delta\omega = \omega_1 - \omega_H = \left.\frac{\omega_M^2}{\sqrt{1 + \frac{\omega_M}{\omega_H} + 1}}\right|_{H_0 \to \infty} \to \frac{\omega_M}{2} \qquad (1.14)$$

ПОМСХ – хвилі з прямою дисперсією (оскільки $d\omega/dk > 0$).

Оскільки МСХ (або ж дипольні СХ) швидко змінюються в просторі, то похідними за часом при записі рівнянь Максвела нехтують в порівнянні з похідними за координатами - нехтуємо ефектами запізнення (поширення) – запізнення реакції магнітної індукції на зміну магнітного поля (вважаємо, що швидкість поширення МСХ прямує до нескінченності).

Тобто критерій справедливості магнітостатичного наближення

$$\lambda_{MCX},\ s << \frac{\lambda_0}{\sqrt{\varepsilon}} \quad \text{чи} \quad k_{MCX} >> \frac{\omega}{\text{с}}\sqrt{\varepsilon} \tag{1.17}$$

- вимога малості характерного розміру зразка в порівнянні з довжиною електромагнітної хвилі (в протилежному випадку врахування впливу граничних умов може суттєво змінить спектр хвиль);

$\lambda_0 = 2\pi c/\omega$ - довжина електромагнітної хвилі в вакуумі.

Для МСХ в плівках ЗІГ магнітостатичне наближення є справедливим на частотах НВЧ діапазону в межах:

- хвильових чисел: $10\ \text{см}^{-1} \le k_{MCX} \le 10^4\ \text{см}^{-1}$,

- довжин хвиль: $6\cdot 10^{-4}\text{см} << \lambda_{MCX},\ s << 0.01\ \text{см}$.

При порушенні магнітостатичного наближення слід враховувати

- обмінні ефекти (робити заміну $\omega \to \omega + \omega_{обм} a^2 k^2$, a - стала гратки фериту, $\omega_{обм} = \gamma H_{обм}$) або ж
- ефекти запізнювання (використовувати записи повних рівнянь Максвела).

Приклад: дисперсія хвиль в дотично намагніченому необмеженому фериті(див. рис. 1.4):

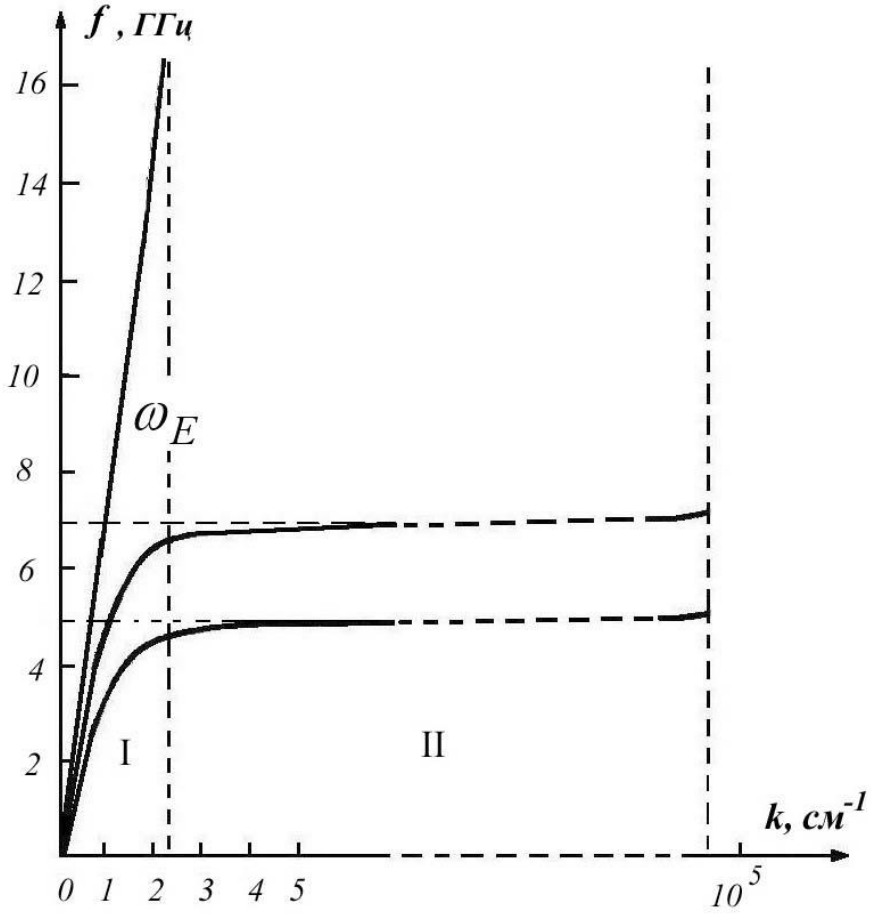

Рис.1. 4 дисперсія хвиль в дотично намагніченому необмеженому фериті

2 умовні ділянки:

I - $k < 10^{-1}$ см$^{-1}$ (або $k/k_0 \approx 1$) – область **електромагнітного поширення**; використовують повні рівняння Максвела; магнітна та електрична складові поля мало відрізняються від величин в звичайному діелектрику.

II – $k/k_0 >> 1$ - область **магнітостатичного поширення**; для опису хвиль використовують рівняння Максвела в наближенні магнітостатики; електричною

2. КОМП’ЮТЕРНЕ МОДЕЛЮВАННЯ ДИСПЕРСІЇ ПОМСХ В ІЗОЛЬОВАНОМУ ФЕРИТОВОМУ ШАРІ

Як згадувалось вище, мета роботи - розробка програми ком’ютерного моделювання дисперсії прямих об’ємних МСХ (ПОМСХ) в тривимірному просторі хвильових чисел та частот, яка б забезпечила візуальний аналіз. З методичної точки зору візуалізація дисперсійних залежностей у тривимірному просторі сприяє розумінню принципів роботи ПОМСХ та приладів спін-хвильової електроніки.

Використовуючи *Microsoft Visual Studio 2022* як інтегроване середовище програмування, та використовуючи бібліотеку WPF розроблено програму для знаходження розв’язків дисперсії ПОМСХ та побудови 3d графіку.

Інтерфейс програми має наступні елементи: область для введення товщини фериту, область для введення величини зовнішнього магнітного поля область для введення значень намагнічуючого поля та область для відображення графіку (див. рис. 2.1). Опишемо ці поля окремо.

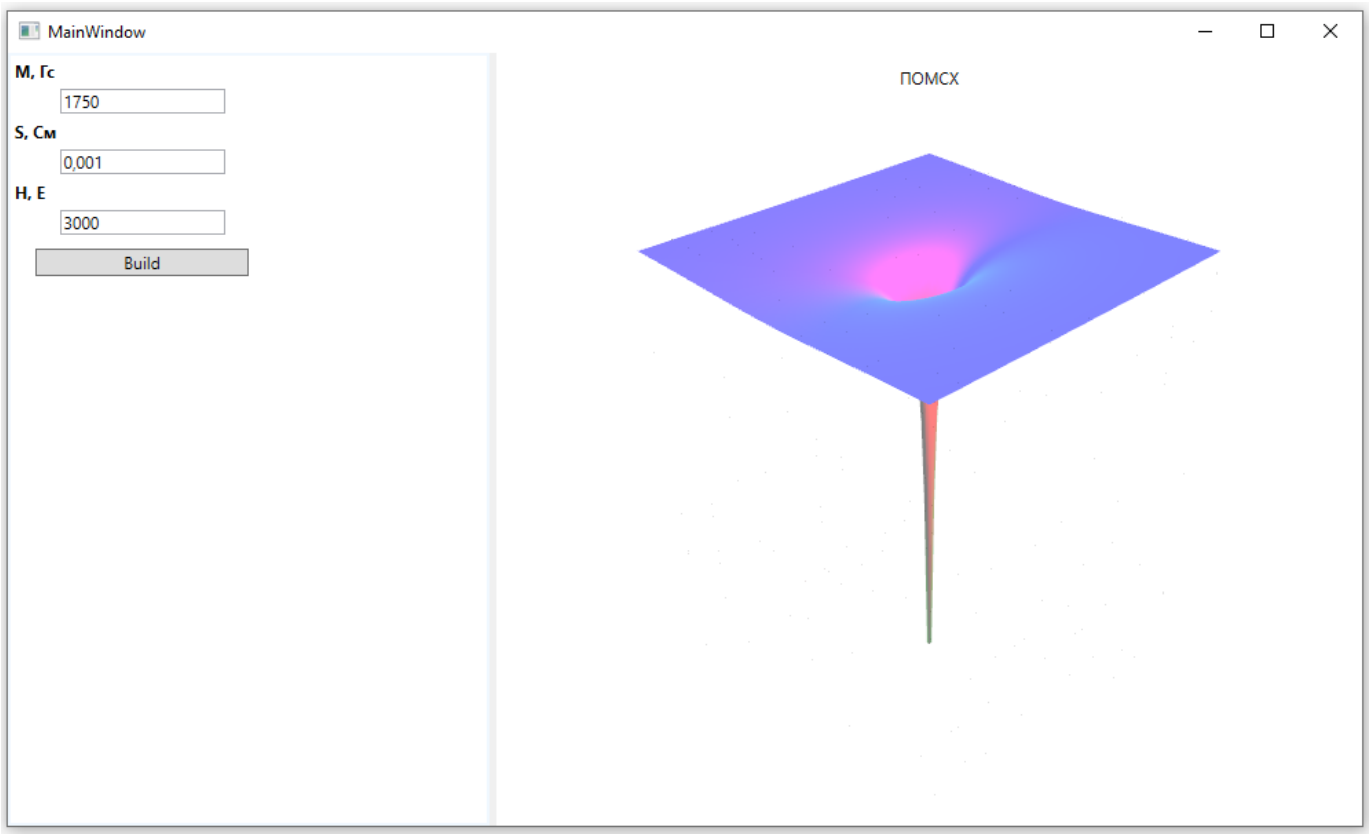

Рис. 2.1 Інтерфейс програми

2.1 Керування областю вводу

Область вводу складаеться з трьох полів (див. рис. 2.2) в які можна вводити:

- величину намагніченості фериту
- величину зовнішнього магнітного поля
- товщину феритової плівки

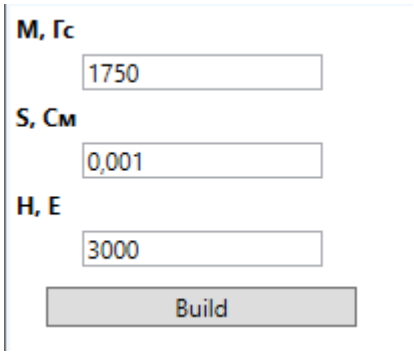

Рис.2.2 область вводу параметрів

Слід зауважити, що вводити слід допустимі значення. Введення некоректних(сторонні символи або крапка на місці коми), дуже великих значень або від'ємних значень - недопустимі. Також слід пам'ятати, що значення **H** повинно бути більше **M**. Величини по замовчуванню зображені на Рис.2.2.

2.2 Область виводу

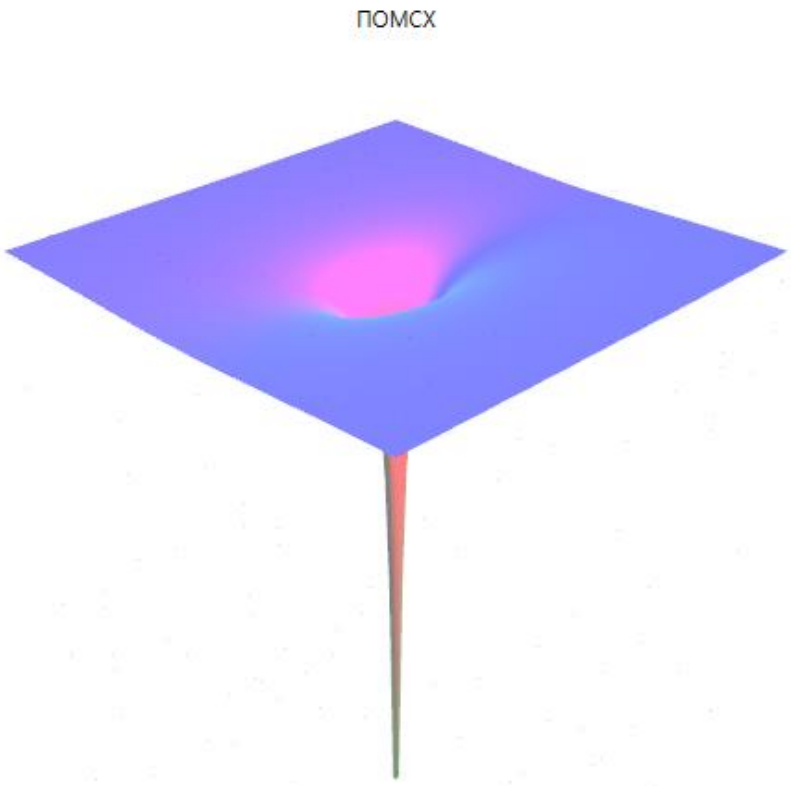

Рис. 2.3 область виводу

В області виводу будується графік дисперсійної залежності для ПОМСХ. Підбираючи параметри товщини фериту можна відображати дисперсійні залежності. Область відображення тривимірного представлення. В цій області знаходиться площина (k_y, k_z); значення k_y та k_z, що обмежені площиною, підбираються в залежності від товщини феритового шару. Перпендикулярно до центру площини проведено вісь для відображення частот (ω).

2.3 Керування областю виводу

Керування областю виводу здійснюється за допомогою миші та клавіатури. Обертання графіку здійснюється заживанням лівої кнопки миші та переміщення миші вздовж поверхні. Якщо при цьому зажати клавішу «Shift», то можна пересувати графік поступально. Клавішами «S» і «W» можна приблизити або віддалити зображення.

2.4 Приклад роботи

Як приклад застосування подивимось як виглядатиме функція для різних параметрів, зазначимо що вісі k_y та k_z мають значення πM та $-\pi M$ відповідно:

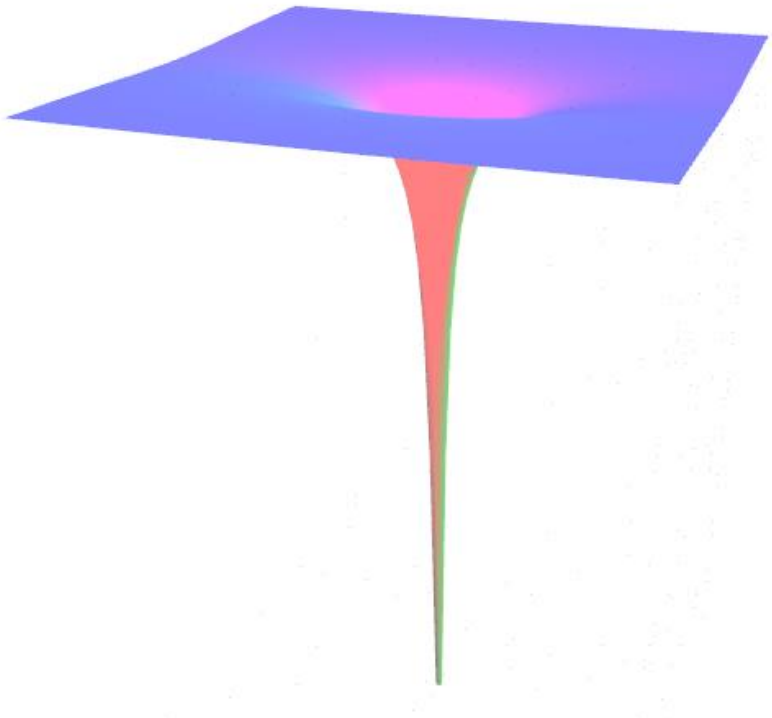

Рис. 2.4 дисперсія ПОМСХ : $4\pi M = 1750$ Гс, $S = 0.001$ см, $H = 3000$ Е

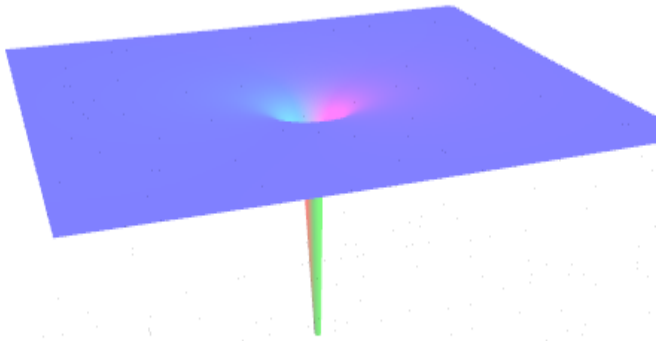

Рис. 2.5 дисперсія ПОМСХ : $4\pi M = 1750$ Гс, $S = 0.00075$ см, $H = 2000$ Е

Висновки

1. Отримано дисперсійне співвідношення для прямих об’ємних магнітостатичних хвиль в феритовому прошарку із врахуванням обмінної взаємодії.
2. Створено програму для знаходження та візуалізації розв’язків отриманого дисперсійного рівняння за різних значень товщин прошарків, намагнічуючих полів та зовнішнього магнітного поля.

Перелік посилань

1. Damon R.W., Eshbach J.R. Magnetostatic modes of ferromagnetic slab // Journ. Phys. Chem. Solids. 1961. **19**, №3/4. P.308-320.
2. Miller N.D.J. Non-reciprocal magnetostatic volume waves // IEEE Trans. 1978. **MAG-14**, №5. P. 829-831.
3. Sechardi S.R. Ray-optics of magnetostatic wave guided by planar film // Journ. Appl. Phys. 1981. **52**, №1. P.25-34.
4. Берегов А.С. Управление спектром и групповой скоростью магнитостатических волн // Радиотехника и электроника. 1983. **28**, №1. С.127-131.
5. Зависляк И.В., Сугаков В.И., Данилов В.В. Магнитостатические волны в длинных прямоугольных стержнях при наличии доменной структуры // Украинский физический журнал. 1977. **22**, №6. С.1042-1044.
6. O'Keeffe T.W., Patterson R.W. Magnetostatic surface-wave propagation in finite samples // Journ. Appl. Phys. 1978. **49**, №9. P.4886-4895.
7. Storey B.E., Cracknell A.O., Przystawa J.A.. The determination of the frequencies of magnetostatic modes in rectangular thin films of ferrimagnetic yttrium iron garnet // Journ. Phys. C.: Solid State Phys. 1977. **10**, №3. P.875-888.
8. Sharon T.M., Maradudim A.A. Magnetostatic modes of gyrotropyc cylinder and bar // Journ. Phys. Chem. Solids. 1977. **38**, №9. P.977-981.
9. Darby M.D., Isaak E.D. Magnetocrystalline anisotropy of ferro- and ferrimagnetics // IEEE Trans. 1974. **MAG-10**, №2. P.259-304.
10. Vittoria C., Wisley N.D. Magnetostatic wave propagation losses in an anisotropic insulator // Journ. Appl. Phys. 1974. **45**, №1. P.414-420.
11. Schneider B. Effect of crystalline anisotropy on the magnetostatic spin modes in ferromagnetic plates (I) // Phys. Stat. Sol. (B). 1972. **51**, №1. P.325-338.
12. Schneider B. Effect of crystalline anisotropy on the magnetostatic spin modes in ferromagnetic plates (I) // Phys. Stat. Sol. (B). 1974. **66**, №1. P.99-106.
13. Данилов В.В., Зависляк И.В., Фирсова Т.П. Магнитостатические волны в нормально намагниченном ферритовом слое // Вестник Киевского университета. Физика. 1982. Вып. 23, С.79-82.
14. Malosemoff A.P., De Luca J.C. Effect of misorientation on growth anisotropy in (III) – oriented garnet films // Journ. Appl. Phys. 1974. **45**, №10. P.4586-4589.
15. Bobkov V.B., Zavislyak I.V. Equilibrium state and permeability tensor of epitaxial ferrite films // Phys. Stat. Sol. (a), 1997. **164**, № 2, P.791-804.
16. Szymczak H., Tsuya N. Phenomenological theory of magnetostriction and growth-induced anisotropy in garnet films // Phys. Stat. Sol. (a), 1979. **54**, № 1. P.117-120.
17. Берегов А.С., Кудинов Е.В. Магнитостатические волны в произвольно ориентированной пленке кубического ферромагнетика. с наведенной анизотропией. Ч. 1. Получение тензора магнитной проницаемости // Электронная техника. Электроника. СВЧ. 1986. Вып. 6 (390), С.41-47.
18. Берегов А.С., Кудинов Е.В. Магнитостатические волны в произвольно ориентированной пленке кубического ферромагнетика с наведенной анизотропией. Ч. 2. Дисперсионные характеристики магнитостатических волн // Электронная техника. Электроника. СВЧ. 1987. Вып. 6 (400), С. 8-12.
19. Gieniusz R., Smoczynski L. Magnetostatic spin waves in (III) – oriented thin garnet films with combined cubic and uniaxial anisotropies // Journ. Magn. Mater. 1987. **66**, №2. P.366-372.

20. Бобков В.Б., Зависляк И.В., Романюк В.Ф. Магнитостатические волны в ферритовых пленках с тригональной анизотропией// Физика твердого тела. 1993. **35**, №2. С.431-435.
21. Бобков В.Б., Зависляк И.В., Романюк В.Ф. Радиоспектроскопия магнитостатических волн в эпитаксиальных ферритовых пленках // Радиотехника и электроника. 2003. **48**, №2. С.222-232.
22. Зависляк И.В., Романюк В.Ф. Опеделение магнитных параметров (111)-ЖИГ пленок // Украинский физический журнал. 1989. **34**, №10. С.1534-1536.
23. Панкрац А.И., Петраковский Г.А., Смык А.Ф. Закон дисперсии магнтостатических колебаний и магнитодинамический резонанс в пластинке слабого ферромагнетика $FeBO_3$// Доклады АН СССР. 1987. **294**, №5. С.1097-1101.
24. Boyle J., Booth J.D., Bordman A.D., Zavislyak I.V., Bobkov V.B., Romanyuk V.F. Investigation of epitaxial Ga:YIG (111) films by Brillouin ligth scattering and microwave spectroscopy // Journal de Physique, 1997. **7**. № 3, C1-497-498.
25. Pugh P.R., Booth J.D., Boyle J., Cowen J.A., Bordman A.D., Zavislyak I.V., Bobkov V.B, Romanyuk V.F. Investigation of anneal epitaxial Ga:YIG (111) films by Brillouin ligth scattering and microwave spectroscopy // Journal of Magnetism and Magnetic Materials. 1999. **196-197**, P . 342-351.
26. Зависляк И.В., Талаевский В.М., Чевнюк Л.В. Особенности спектров магнитостатических волн, обусловленные анизотропией // Физика твердого тела. 1989. **31**, №5. С.319-321.

Додаток 1

Текст програми:

1. MainWindow.xaml.cs:

```
using System;
using System.Windows;
namespace WPFSurfacePlot3D {
public partial class MainWindow : Window {
private SurfacePlotModel viewModel;
private double YY { get; set; }
private double PM4 { get; set; }
private double H0 { get; set; }
private double S0 { get; set; }
public MainWindow() {
InitializeComponent();
viewModel = new SurfacePlotModel();
YY = 2800000;
PM4 = 1750;
H0 = 3000;
S0 = 0.001;
surfacePlotView.DataContext = viewModel;
}
private void M_TextChanged(object sender,
System.Windows.Controls.TextChangedEventArgs e) {
string mm = M.Text;
double returnedValue;
if (double.TryParse((string)mm, out returnedValue)) {
PM4 = returnedValue;
} }
private void S_TextChanged(object sender,
System.Windows.Controls.TextChangedEventArgs e) {
string mm = S.Text; double returnedValue;
if (double.TryParse((string)mm, out returnedValue)) {
H0 = returnedValue;
} }
private void H_TextChanged(object sender,
System.Windows.Controls.TextChangedEventArgs e) {
string mm = H.Text;
double returnedValue;
if (double.TryParse((string)mm, out returnedValue)) {
H0 = returnedValue;
} }
private void Button_Click(object sender, RoutedEventArgs e) {
double wh = YY * H0;
double wm = YY * PM4;
double w_min = wh - wm;
double w_max = Math.Sqrt(wh * (wh + wm));
double w_1 = Math.Sqrt(wh * (wh + wm));
double w_step = (w_max - w_min) / 2;
double w = w_min + w_step;
double kz = -PM4 / 4, ky = PM4 / 4;
double u = (w * w - w_1 * w_1) / (w * w - wh * wh);
Func<double, double, double> sampleFunction = (x, y) => u + 1 - 2 * Math.Sqrt(u) * 1
/ Math.Tan(Math.Sqrt(y * y + x * x / u) * S0 / Math.Sqrt(u));
viewModel.PlotFunction(sampleFunction, kz, ky);
} } }
```

2. SurfacePlotModel.cs:

```
using HelixToolkit.Wpf;
using System;
using System.ComponentModel;
using System.Windows.Media;
using System.Windows.Media.Media3D;

namespace WPFSurfacePlot3D
{
    public enum ColorCoding
    {
        ByLights,
        ByGradientY
    }

    class SurfacePlotModel : INotifyPropertyChanged
    {
        private int defaultFunctionSampleSize = 100;

        public SurfacePlotModel()
        {
            Title = "ПОМСХ";
            XAxisLabel = "x-Axis";
            YAxisLabel = "y-Axis";
            ZAxisLabel = "z-Axis";

            ColorCoding = ColorCoding.ByLights;

            double Y = 2800000;
            double PM4 = 1750;
            double H = 3000;
            double S = 0.001;

            double wh = Y * H;
            double wm = Y * PM4;

            double w_min = wh - wm;
            double w_max = Math.Sqrt(wh * (wh + wm));
            double w_1 = Math.Sqrt(wh * (wh + wm));
            double w_step = (w_max - w_min) / 2;
            double w = w_min + w_step;
            double kz = -PM4 / 4, ky = PM4 / 4;
            double u = (w * w - w_1 * w_1) / (w * w - wh * wh);

            Func<double, double, double> sampleFunction = (x, y) => u + 1 - 2 *
Math.Sqrt(u) * 1 / Math.Tan(Math.Sqrt(y * y + x * x / u) * S / Math.Sqrt(u));
            PlotFunction(sampleFunction, kz , ky);
        }

        private double xMin, xMax, yMin, yMax, zMin, zMax;
        public void PlotFunction(Func<double, double, double> function)
        {
            PlotFunction(function, -1, 1, -1, 1, defaultFunctionSampleSize,
defaultFunctionSampleSize);
        }
```

```
        public void PlotFunction(Func<double, double, double> function, double
minimumXY, double maximumXY)
        {
            PlotFunction(function, minimumXY, maximumXY, minimumXY, maximumXY,
defaultFunctionSampleSize, defaultFunctionSampleSize);
        }

        public void PlotFunction(Func<double, double, double> function, double
minimumXY, double maximumXY, int sampleSize)
        {
            PlotFunction(function, minimumXY, maximumXY, minimumXY, maximumXY,
sampleSize, sampleSize);
        }

        public void PlotFunction(Func<double, double, double> function, double
xMinimum, double xMaximum, double yMinimum, double yMaximum)
        {
            PlotFunction(function, xMinimum, xMaximum, yMinimum, yMaximum,
defaultFunctionSampleSize, defaultFunctionSampleSize);
        }

        public void PlotFunction(Func<double, double, double> function, double
xMinimum, double xMaximum, double yMinimum, double yMaximum, int sampleSize)
        {
            PlotFunction(function, xMinimum, xMaximum, yMinimum, yMaximum,
sampleSize, sampleSize);
        }

        public void PlotFunction(Func<double, double, double> function, double
xMinimum, double xMaximum, double yMinimum, double yMaximum, int xSampleSize, int
ySampleSize)
        {
            xMin = xMinimum;
            xMax = xMaximum;
            yMin = yMinimum;
            yMax = yMaximum;

            double[] xArray = CreateLinearlySpacedArray(xMinimum, xMaximum,
xSampleSize);
            double[] yArray = CreateLinearlySpacedArray(yMinimum, yMaximum,
ySampleSize);

            DataPoints = CreateDataArrayFromFunction(function, xArray, yArray);
            switch (ColorCoding)
            {
                case ColorCoding.ByGradientY:
                    ColorValues = FindGradientY(DataPoints);
                    break;
                case ColorCoding.ByLights:
                    ColorValues = null;
                    break;
            }
            RaisePropertyChanged("DataPoints");
            RaisePropertyChanged("ColorValues");
            RaisePropertyChanged("SurfaceBrush");
        }
```

```
        private Point3D[,] CreateDataArrayFromFunction(Func<double, double, double>
f, double[] xArray, double[] yArray)
        {
            Point3D[,] newDataArray = new Point3D[xArray.Length, yArray.Length];
            for (int i = 0; i < xArray.Length; i++)
            {
                double x = xArray[i];
                for (int j = 0; j < yArray.Length; j++)
                {
                    double y = yArray[j];
                    newDataArray[i, j] = new Point3D(x, y, f(x, y));
                }
            }
            return newDataArray;
        }

        private double[] CreateLinearlySpacedArray(double minValue, double maxValue,
int numberOfPoints)
        {
            double[] array = new double[numberOfPoints];
            double intervalSize = (xMax - xMin) / (numberOfPoints - 1);
            for (int i = 0; i < numberOfPoints; i++)
            {
                array[i] = minValue + i * intervalSize;
            }
            return array;
        }

        public event PropertyChangedEventHandler PropertyChanged;

        protected void RaisePropertyChanged(string property)
        {
            var handler = PropertyChanged;
            if (handler != null)
            {
                handler(this, new PropertyChangedEventArgs(property));
            }
        }

        private Point3D[,] dataPoints;
        public Point3D[,] DataPoints
        {
            get { return dataPoints; }
            set
            {
                dataPoints = value;
            }
        }

        private double[] xAxisTicks;
        public double[] XAxisTicks
        {
            get { return xAxisTicks; }
            set
            {
                xAxisTicks = value;
            }
```

```
}

private double[] yAxisTicks;
public double[] YAxisTicks
{
    get { return yAxisTicks; }
    set
    {
        yAxisTicks = value;
    }
}

private double[] zAxisTicks;
public double[] ZAxisTicks
{
    get { return zAxisTicks; }
    set
    {
        zAxisTicks = value;
    }
}

private string title;
public string Title
{
    get { return title; }
    set
    {
        title = value;
        RaisePropertyChanged("Title");
    }
}

private string xAxisLabel;
public string XAxisLabel
{
    get { return xAxisLabel; }
    set
    {
        xAxisLabel = value;
        RaisePropertyChanged("XAxisLabel");
    }
}

private string yAxisLabel;
public string YAxisLabel
{
    get { return yAxisLabel; }
    set
    {
        yAxisLabel = value;
        RaisePropertyChanged("YAxisLabel");
    }
}

private string zAxisLabel;
public string ZAxisLabel
{
```

```
    get { return zAxisLabel; }
    set
    {
        zAxisLabel = value;
        RaisePropertyChanged("ZAxisLabel");
    }
}

private bool showSurfaceMesh;
public bool ShowSurfaceMesh
{
    get { return showSurfaceMesh; }
    set
    {
        showSurfaceMesh = value;
        RaisePropertyChanged("ShowSurfaceMesh");
    }
}

private bool showContourLines;
public bool ShowContourLines
{
    get { return showContourLines; }
    set
    {
        showContourLines = value;
        RaisePropertyChanged("ShowContourLines");
    }
}

private bool showMiniCoordinates;
public bool ShowMiniCoordinates
{
    get { return showMiniCoordinates; }
    set
    {
        showMiniCoordinates = value;
        RaisePropertyChanged("ShowMiniCoordinates");
    }
}




public double[,] ColorValues { get; set; }

public ColorCoding ColorCoding { get; set; }

public Model3DGroup Lights
{
    get
    {
        var group = new Model3DGroup();
        switch (ColorCoding)
        {
            case ColorCoding.ByGradientY:
                group.Children.Add(new AmbientLight(Colors.White));
                break;
```

```
                    case ColorCoding.ByLights:
                        group.Children.Add(new AmbientLight(Colors.Gray));
                        group.Children.Add(new PointLight(Colors.Red, new Point3D(0,
-1000, 0)));
                        group.Children.Add(new PointLight(Colors.Blue, new Point3D(0,
0, 1000)));
                        group.Children.Add(new PointLight(Colors.Green, new
Point3D(1000, 1000, 0)));
                        break;
                }
                return group;
            }
        }
        public Brush SurfaceBrush
        {
            get
            {
                switch (ColorCoding)
                {
                    case ColorCoding.ByGradientY:
                        return BrushHelper.CreateGradientBrush(Colors.Red,
Colors.White, Colors.Blue);
                    case ColorCoding.ByLights:
                        return Brushes.White;
                }
                return null;
            }
        }
        public double[,] FindGradientY(Point3D[,] data)
        {
            int n = data.GetUpperBound(0) + 1;
            int m = data.GetUpperBound(0) + 1;
            var K = new double[n, m];
            for (int i = 0; i < n; i++)
                for (int j = 0; j < m; j++)
                {
                    var p10 = data[i + 1 < n ? i + 1 : i, j - 1 > 0 ? j - 1 : j];
                    var p00 = data[i - 1 > 0 ? i - 1 : i, j - 1 > 0 ? j - 1 : j];
                    var p11 = data[i + 1 < n ? i + 1 : i, j + 1 < m ? j + 1 : j];
                    var p01 = data[i - 1 > 0 ? i - 1 : i, j + 1 < m ? j + 1 : j];

                    double dy = p10.Y - p00.Y;
                    double dz = p10.Z - p00.Z;

                    K[i, j] = dz / dy;
                }
            return K;
        }
    }
}
```

3. SurfacePlotView.xaml.cs:

```
using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;
using System.Windows.Media.Media3D;

namespace WPFSurfacePlot3D
```

```
{

    public partial class SurfacePlotView : UserControl
    {
        public SurfacePlotView()
        {
            InitializeComponent();
            DataContext = LayoutRoot.DataContext;
            hViewport.ZoomExtentsGesture = new KeyGesture(Key.Space);
        }

        public Point3D[,] DataPoints
        {
            get { return (Point3D[,])GetValue(DataPointsProperty); }
            set { SetValue(DataPointsProperty, value); }
        }

        public static readonly DependencyProperty DataPointsProperty =
DependencyProperty.Register("DataPoints", typeof(Point3D[,]),
typeof(SurfacePlotView), new
FrameworkPropertyMetadata(SurfacePlotVisual3D.SamplePoints));

        public string Title
        {
            get { return (string)GetValue(TitleProperty); }
            set { SetValue(TitleProperty, value); }
        }

        public static readonly DependencyProperty TitleProperty =
DependencyProperty.Register("Title", typeof(string), typeof(SurfacePlotView), new
FrameworkPropertyMetadata("Surface Plot Title"));

        public string XAxisLabel
        {
            get { return (string)GetValue(XAxisLabelProperty); }
            set { SetValue(XAxisLabelProperty, value); }
        }

        public static readonly DependencyProperty XAxisLabelProperty =
DependencyProperty.Register("XAxisLabel", typeof(string), typeof(SurfacePlotView),
new FrameworkPropertyMetadata("X Axis Label"));

        public string YAxisLabel
        {
            get { return (string)GetValue(YAxisLabelProperty); }
            set { SetValue(YAxisLabelProperty, value); }
        }

        public static readonly DependencyProperty YAxisLabelProperty =
DependencyProperty.Register("YAxisLabel", typeof(string), typeof(SurfacePlotView),
new FrameworkPropertyMetadata("Y Axis Label"));

        public string ZAxisLabel
        {
            get { return (string)GetValue(ZAxisLabelProperty); }
            set { SetValue(ZAxisLabelProperty, value); }
```

```
        }

        public static readonly DependencyProperty ZAxisLabelProperty =
DependencyProperty.Register("ZAxisLabel", typeof(string), typeof(SurfacePlotView),
new FrameworkPropertyMetadata("Z Axis Label"));

        public bool ShowSurfaceMesh
        {
            get { return (bool)GetValue(ShowSurfaceMeshProperty); }
            set { SetValue(ShowSurfaceMeshProperty, value); }
        }

        public static readonly DependencyProperty ShowSurfaceMeshProperty =
DependencyProperty.Register("ShowSurfaceMesh", typeof(bool), typeof(SurfacePlotView),
new FrameworkPropertyMetadata(true));

        public bool ShowContourLines
        {
            get { return (bool)GetValue(ShowContourLinesProperty); }
            set { SetValue(ShowContourLinesProperty, value); }
        }

        public static readonly DependencyProperty ShowContourLinesProperty =
DependencyProperty.Register("ShowContourLines", typeof(bool),
typeof(SurfacePlotView), new FrameworkPropertyMetadata(true));

        public bool ShowMiniCoordinates
        {
            get { return (bool)GetValue(ShowMiniCoordinatesProperty); }
            set { SetValue(ShowMiniCoordinatesProperty, value); }
        }

        public static readonly DependencyProperty ShowMiniCoordinatesProperty =
DependencyProperty.Register("ShowMiniCoordinates", typeof(bool),
typeof(SurfacePlotView), new FrameworkPropertyMetadata(true));

    }
}
```

4. SurfacePlotVisual3D.cs:

```
using HelixToolkit.Wpf;
using System;
using System.Collections.Generic;
using System.Windows;
using System.Windows.Media;
using System.Windows.Media.Media3D;


namespace WPFSurfacePlot3D
{

    public class SurfacePlotVisual3D : ModelVisual3D
    {
        private readonly ModelVisual3D modelContainer;

        public SurfacePlotVisual3D()
```

```
        {
            IntervalX = 1;
            IntervalY = 1;
            IntervalZ = 0.25;
            FontSize = 0.06;
            LineThickness = 0.01;

            modelContainer = new ModelVisual3D();
            Children.Add(modelContainer);
        }
        public Point3D[,] DataPoints
        {
            get { return (Point3D[,])GetValue(DataPointsProperty); }
            set { SetValue(DataPointsProperty, value); }
        }

        public static readonly DependencyProperty DataPointsProperty =
DependencyProperty.Register("DataPoints", typeof(Point3D[,]),
typeof(SurfacePlotVisual3D), new UIPropertyMetadata(SamplePoints, ModelWasChanged));

        public double[,] ColorValues
        {
            get { return (double[,])GetValue(ColorValuesProperty); }
            set { SetValue(ColorValuesProperty, value); }
        }

        public static readonly DependencyProperty ColorValuesProperty =
DependencyProperty.Register("ColorValues", typeof(double[,]),
typeof(SurfacePlotVisual3D), new UIPropertyMetadata(null, ModelWasChanged));

        public Brush SurfaceBrush
        {
            get { return (Brush)GetValue(SurfaceBrushProperty); }
            set { SetValue(SurfaceBrushProperty, value); }
        }

        public static readonly DependencyProperty SurfaceBrushProperty =
DependencyProperty.Register("SurfaceBrush", typeof(Brush),
typeof(SurfacePlotVisual3D), new UIPropertyMetadata(null, ModelWasChanged));

        public double IntervalX { get; set; }
        public double IntervalY { get; set; }
        public double IntervalZ { get; set; }
        public double FontSize { get; set; }
        public double LineThickness { get; set; }

        private static void ModelWasChanged(DependencyObject d,
DependencyPropertyChangedEventArgs e)
        {
            ((SurfacePlotVisual3D)d).UpdateModel();

        }

        private void UpdateModel()
        {
            this.Children.Clear();
```

```
            Children.Add(modelContainer);

            this.Content = CreateModel();
        }
        private Model3DGroup CreateModel()
        {
            var newModelGroup = new Model3DGroup();
            double lineThickness = 0.01;
            double axesOffset = 0.05;

            int numberOfRows = DataPoints.GetUpperBound(0) + 1;
            int numberOfColumns = DataPoints.GetUpperBound(1) + 1;

            double minX = double.MaxValue;
            double maxX = double.MinValue;
            double minY = double.MaxValue;
            double maxY = double.MinValue;
            double minZ = double.MaxValue;
            double maxZ = double.MinValue;

            double minColorValue = double.MaxValue;
            double maxColorValue = double.MinValue;

            for (int i = 0; i < numberOfRows; i++)
            {
                for (int j = 0; j < numberOfColumns; j++)
                {
                    double x = DataPoints[i, j].X;
                    double y = DataPoints[i, j].Y;
                    double z = DataPoints[i, j].Z;
                    maxX = Math.Max(maxX, x);
                    maxY = Math.Max(maxY, y);
                    maxZ = Math.Max(maxZ, z);
                    minX = Math.Min(minX, x);
                    minY = Math.Min(minY, y);
                    minZ = Math.Min(minZ, z);
                    if (ColorValues != null)
                    {
                        maxColorValue = Math.Max(maxColorValue, ColorValues[i, j]);
                        minColorValue = Math.Min(minColorValue, ColorValues[i, j]);
                    }
                }
            }

            int numberOfXAxisTicks = 10;
            int numberOfYAxisTicks = 10;
            int numberOfZAxisTicks = 5;
            double XAxisInterval = (maxX - minX) / numberOfXAxisTicks;
            double YAxisInterval = (maxY - minY) / numberOfYAxisTicks;
            double ZAxisInterval = (maxZ - minZ) / numberOfZAxisTicks;

            if (Math.Abs(minColorValue) < Math.Abs(maxColorValue)) { minColorValue =
-maxColorValue; }
            else { maxColorValue = -minColorValue; }

            var textureCoordinates = new Point[numberOfRows, numberOfColumns];
```

```
            for (int i = 0; i < numberOfRows; i++)
            {
                for (int j = 0; j < numberOfColumns; j++)
                {
                    double tc;
                    if (ColorValues != null) { tc = (ColorValues[i, j] -
minColorValue) / (maxColorValue - minColorValue); }
                    else { tc = (DataPoints[i, j].Z - minZ) / (maxZ - minZ); }
                    textureCoordinates[i, j] = new Point(tc, tc);
                }
            }

            MeshBuilder surfaceModelBuilder = new MeshBuilder();
            surfaceModelBuilder.AddRectangularMesh(DataPoints, textureCoordinates);

            GeometryModel3D surfaceModel = new
GeometryModel3D(surfaceModelBuilder.ToMesh(),
MaterialHelper.CreateMaterial(SurfaceBrush, null, null, 1, 0));
            surfaceModel.BackMaterial = surfaceModel.Material;

            MeshBuilder surfaceMeshLinesBuilder = new MeshBuilder();
            MeshBuilder surfaceContourLinesBuilder = new MeshBuilder();
            MeshBuilder gridBuilder = new MeshBuilder();

            ModelVisual3D axesLabelsModel = new ModelVisual3D();

            for (double x = minX; x <= maxX + 0.0001; x += XAxisInterval)
            {
                var surfacePath = new List<Point3D>();
                double i = (x - minX) / (maxX - minX) * (numberOfColumns - 1);
                for (int j = 0; j < numberOfColumns; j++)
                {
                    surfacePath.Add(DoBilinearInterpolation(DataPoints, i, j));
                }
                surfaceMeshLinesBuilder.AddTube(surfacePath, lineThickness, 9,
false);

                BillboardTextVisual3D label = new BillboardTextVisual3D();
                label.Text = string.Format("{0:F2}", x);
                label.Position = new Point3D(x, minY - axesOffset, minZ -
axesOffset);
                axesLabelsModel.Children.Add(label);

                var gridPath = new List<Point3D>();
                gridPath.Add(new Point3D(x, minY, minZ));
                gridPath.Add(new Point3D(x, maxY, minZ));
                gridPath.Add(new Point3D(x, maxY, maxZ));
                gridBuilder.AddTube(gridPath, lineThickness, 9, false);

            }

            for (double y = minY; y <= maxY + 0.0001; y += YAxisInterval)
            {
                var path = new List<Point3D>();
                double j = (y - minY) / (maxY - minY) * (numberOfRows - 1);
                for (int i = 0; i < numberOfRows; i++)
```

```
                {
                    path.Add(DoBilinearInterpolation(DataPoints, i, j));
                }
                surfaceMeshLinesBuilder.AddTube(path, lineThickness, 9, false);

                BillboardTextVisual3D label = new BillboardTextVisual3D();
                label.Text = string.Format("{0:F2}", y);
                label.Position = new Point3D(minX - axesOffset, y, minZ -
axesOffset);
                axesLabelsModel.Children.Add(label);

                var gridPath = new List<Point3D>();
                gridPath.Add(new Point3D(minX, y, minZ));
                gridPath.Add(new Point3D(maxX, y, minZ));
                gridPath.Add(new Point3D(maxX, y, maxZ));
                gridBuilder.AddTube(gridPath, lineThickness, 9, false);
            }

            for (double z = minZ; z <= maxZ + 0.0001; z += ZAxisInterval)
            {

                var path = new List<Point3D>();
                path.Add(new Point3D(minX, maxY, z));
                path.Add(new Point3D(maxX, maxY, z));
                path.Add(new Point3D(maxX, minY, z));
                gridBuilder.AddTube(path, lineThickness, 9, false);

                BillboardTextVisual3D label = new BillboardTextVisual3D();
                label.Text = string.Format("{0:F2}", z);
                label.Position = new Point3D(minX - axesOffset, maxY + axesOffset,
z);
                axesLabelsModel.Children.Add(label);
            }

            BillboardTextVisual3D xLabel = new BillboardTextVisual3D();
            xLabel.Text = "X Axis";
            xLabel.Position = new Point3D((maxX - minX) / 2, minY - 3 * axesOffset,
minZ - 5 * axesOffset);
            axesLabelsModel.Children.Add(xLabel);
            BillboardTextVisual3D yLabel = new BillboardTextVisual3D();
            yLabel.Text = "Y Axis";
            yLabel.Position = new Point3D(minX - 3 * axesOffset, (maxY - minY) / 2,
minZ - 5 * axesOffset);
            axesLabelsModel.Children.Add(yLabel);
            BillboardTextVisual3D zLabel = new BillboardTextVisual3D();
            zLabel.Text = "Z Axis";
            zLabel.Position = new Point3D(minX - 5 * axesOffset, maxY + 5 *
axesOffset, 0); // Note: trying to find the midpoint of minZ, maxZ doesn't work when
minZ = -0.5 and maxZ = 0.5...
            axesLabelsModel.Children.Add(zLabel);

            GeometryModel3D surfaceMeshLinesModel = new
GeometryModel3D(surfaceMeshLinesBuilder.ToMesh(), Materials.Black);
            GeometryModel3D gridModel = new GeometryModel3D(gridBuilder.ToMesh(),
Materials.Black);
```

```
            newModelGroup.Children.Add(surfaceModel);
            newModelGroup.Children.Add(surfaceMeshLinesModel);
            newModelGroup.Children.Add(gridModel);

            return newModelGroup;
        }

        private static Point3D DoBilinearInterpolation(Point3D[,] points, double i,
double j)
        {
            int n = points.GetUpperBound(0);
            int m = points.GetUpperBound(1);
            var i0 = (int)i;
            var j0 = (int)j;
            if (i0 + 1 >= n) i0 = n - 2;
            if (j0 + 1 >= m) j0 = m - 2;

            if (i < 0) i = 0;
            if (j < 0) j = 0;
            double u = i - i0;
            double v = j - j0;
            Vector3D v00 = points[i0, j0].ToVector3D();
            Vector3D v01 = points[i0, j0 + 1].ToVector3D();
            Vector3D v10 = points[i0 + 1, j0].ToVector3D();
            Vector3D v11 = points[i0 + 1, j0 + 1].ToVector3D();
            Vector3D v0 = v00 * (1 - u) + v10 * u;
            Vector3D v1 = v01 * (1 - u) + v11 * u;
            return (v0 * (1 - v) + v1 * v).ToPoint3D();
        }
        public static Point3D[,] SamplePoints
        {
            get
            {
                int n = 50;
                Point3D[,] points = new Point3D[n, n];
                for (int i = 0; i < n; i++)
                {
                    for (int j = 0; j < n; j++)
                    {
                        double x = i * (Math.PI / n);
                        double y = j * (Math.PI / n);
                        double z = 0.5 * Math.Sin(x * y);
                        points[i, j] = new Point3D(x, y, z);
                    }
                }
                return points;
            }
        }
    }
}
```

5. MainWindow.xaml:

```
<Window x:Class="WPFSurfacePlot3D.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
```

```
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:ExtendedToolkit="http://schemas.xceed.com/wpf/xaml/toolkit"
        xmlns:SurfacePlot="clr-namespace:WPFSurfacePlot3D"
        mc:Ignorable="d"
        Title="MainWindow" Height="600" Width="1000">

    <DockPanel x:Name="Parent">
        <Grid>
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="1*" MinWidth="350" />
                <ColumnDefinition Width="5" />
                <ColumnDefinition Width="3*" MinWidth="200" />
            </Grid.ColumnDefinitions>

            <Border Grid.Column="0" Margin="0" BorderBrush="AliceBlue"
BorderThickness="2" />
            <Border Grid.Column="1" Margin="0" BorderBrush="AliceBlue"
BorderThickness="2" />

            <ScrollViewer Grid.Column="0" VerticalScrollBarVisibility="Auto"
Margin="0,0,175,0">
                <StackPanel Margin="0" Width="175">
                    <Label Content="M, Гс" FontWeight="Bold"/>
                    <TextBox x:Name="M"  TextWrapping="Wrap" Text="1750" Width="120"
TextChanged="M_TextChanged" Margin="20,0,0,0" />
                    <Label Content="S, См" FontWeight="Bold"/>
                    <TextBox x:Name="S" TextWrapping="Wrap" Text="0,001" Width="120"
TextChanged="S_TextChanged" Margin="20,0,0,0"/>
                    <Label Content="H, E" FontWeight="Bold"/>
                    <TextBox x:Name="H"  TextWrapping="Wrap" Text="3000" Width="120"
TextChanged="H_TextChanged" Margin="20,0,0,0"/>

                    <Button Content="Build" Click="Button_Click" Margin="20,10,0,0"/>



                </StackPanel>
            </ScrollViewer>

            <GridSplitter Grid.Column="1" Width="5" HorizontalAlignment="Stretch" />

            <SurfacePlot:SurfacePlotView
                x:Name="surfacePlotView"
                Title="Sample Surface Plot" Grid.Column="2"
HorizontalAlignment="Center" Width="645"
                />
        </Grid>
    </DockPanel>
</Window>
```

6. SurfacePlotView.xaml:

```
<UserControl x:Class="WPFSurfacePlot3D.SurfacePlotView"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:h="clr-namespace:HelixToolkit.Wpf;assembly=HelixToolkit.Wpf"
```

```
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             xmlns:local="clr-namespace:WPFSurfacePlot3D"
             mc:Ignorable="d"
             d:DesignHeight="300" d:DesignWidth="300">

    <Grid x:Name="LayoutRoot">
        <Grid.RowDefinitions>
            <RowDefinition Height="1*"/>
            <RowDefinition Height="11*"/>
        </Grid.RowDefinitions>

        <TextBlock
            Text="{Binding Title, UpdateSourceTrigger=PropertyChanged}"
            Grid.Row="0"
            Margin="10"
            HorizontalAlignment="Center"
            />

        <h:HelixViewport3D
            x:Name="hViewport"
            Grid.Row="1"
            ZoomExtentsWhenLoaded="True"
            PanGesture="LeftClick"
            ShowViewCube="False"
            ShowCoordinateSystem="{Binding ShowMiniCoordinates,
UpdateSourceTrigger=PropertyChanged}"
            CoordinateSystemLabelX="{Binding XAxisLabel,
UpdateSourceTrigger=PropertyChanged}"
            CoordinateSystemLabelY="{Binding YAxisLabel,
UpdateSourceTrigger=PropertyChanged}"
            CoordinateSystemLabelZ="{Binding ZAxisLabel,
UpdateSourceTrigger=PropertyChanged}"
            >

            <h:HelixViewport3D.Camera>
                <PerspectiveCamera LookDirection="1,1,-1" UpDirection="0,0,1"/>
            </h:HelixViewport3D.Camera>

            <!-- Lights -->
            <ModelVisual3D Content="{Binding Lights,
UpdateSourceTrigger=PropertyChanged}"/>

            <!-- The plot visual (surface, axes and labels) -->
            <local:SurfacePlotVisual3D
                DataPoints="{Binding DataPoints,
UpdateSourceTrigger=PropertyChanged}"
                ColorValues="{Binding ColorValues,
UpdateSourceTrigger=PropertyChanged}"
                SurfaceBrush="{Binding SurfaceBrush,
UpdateSourceTrigger=PropertyChanged}"
                />

        </h:HelixViewport3D>
    </Grid>
</UserControl>
```