Одеський національний університет імені І. І. Мечникова
Факультет математики, фізики та інформаційних технологій
Кафедра методів математичної фізики

Кваліфікаційна робота

на здобуття ступеня вищої освіти «магістр»

«Задача повздовжнього зсуву для зовнішнього складеного сектору кола з міжфазним дефектом»

«The antiplane problem of the elasticity theory for the external composed sector of a circle with an interface defect»

Виконав: здобувач денної форми навчання
спеціальності 113 Прикладна математика
Освітня програма «Прикладна математика»

Метасов Андрій Сергійович

Керівник: канд. фіз.-мат. наук, доц. Процеров Ю.С. _____
Рецензент: канд. фіз.-мат. наук, доц. Журавльова З. Ю.

Рекомендовано до захисту:
Протокол засідання кафедри
№ ___ від _____ р.

Завідувач кафедри

_____ _____
(підпис) (прізвище, ім'я)

Захищено на засіданні ЕК №_____
протокол № ____ від _____ р.
Оцінка_____/___ ___/_____
(за національною шкалою, шкалою ECTS, бали)
Голова ЕК

_____ _____
(підпис) (прізвище, ім'я)

Одеса – 2024

ЗМІСТ

ВСТУП

В роботі досліджується задача поздовжнього зсуву тіла, що є зовнішнім сектором кола, складається з двох матеріалів із різними пружними характеристиками та має дефект у вигляді тріщини на лінії сполучення матеріалів.

Дослідження поздовжньої деформації в складених тілах з міжфазними дефектами є важливим етапом забезпечення надійності та довговічності відповідних інженерних конструкцій. Особливий інтерес викликає наявність тріщини між матеріалами, яка може призвести до концентрації напружень і, відповідно, до розвитку дефекту, що може значно вплинути на міцність конструкції.

Розглянемо пружне тіло, яке знаходиться в циліндричній системі координат (r, θ, z) та обмежено наступною областю: $a < r < \infty,\ -\beta < \theta < \beta,$ $-\infty < z < \infty.$ Нехай одна з прямолінійних граней, відповідна до $\theta = \beta,\ a < r < \infty,$ $|z| < \infty$ – нерухомо закріплена, а протилежна грань, відповідна до $\theta = -\beta,$ $a < r < \infty,\ |z| < \infty$ – вільна від напружень. Припустимо, що до криволінійної грані, яка відповідає $r = a,\ -\beta < \theta < \beta,\ |z| < \infty$ – прикладено навантаження з відомою інтенсивністю, яка описується функцією $f(\theta)$ і діє вздовж осі OZ.

За такої постановки задачі тіло перебуває в умовах антиплоскої деформації, що означає відсутність залежності всіх величин від координати z. Ненульовими за таких умов залишаться тільки три механічні характеристики: переміщення вздовж осі OZ $W(r, \theta)$, тангенціальне дотичне напруження $\mathrm{T}_{\theta z}(r, \theta)$ і радіальне дотичне напруження $\mathrm{T}_{rz}(r, \theta)$. Тут

$$\mathrm{T}_{\theta z} = G\frac{1}{r}\frac{\partial W}{\partial \theta},\ \mathrm{T}_{rz} = G\frac{\partial W}{\partial r}.$$

Також, нехай розглянуте тіло складається з двох матеріалів, які мають різні пружні характеристики і примикають один до одного по лінії $r=c$. Частина тіла, яка відповідає області $a<r<c,\ -\beta<\theta<\beta,\ |z|<\infty$, матиме модуль зсуву G_1, а інша частина, яка відповідає $c<r<\infty,\ -\beta<\theta<\beta,\ |z|<\infty$, матиме модуль зсуву G_2. Припустимо, що на лінії сполучення матеріалів $(r=c)$ виконуються умови ідеального механічного контакту всюди, за винятком ділянки, де знаходиться дефект. Тобто, переміщення $W(r,\theta)$ та радіальні напруження $T_{rz}(r,\theta)$ неперервні при переході через лінію $r=c$ всюди, окрім проміжка $r=c$, $\omega_1<\theta<\omega_2$, де розташована тріщина, берега якої вільні від напружень (рис. 1).

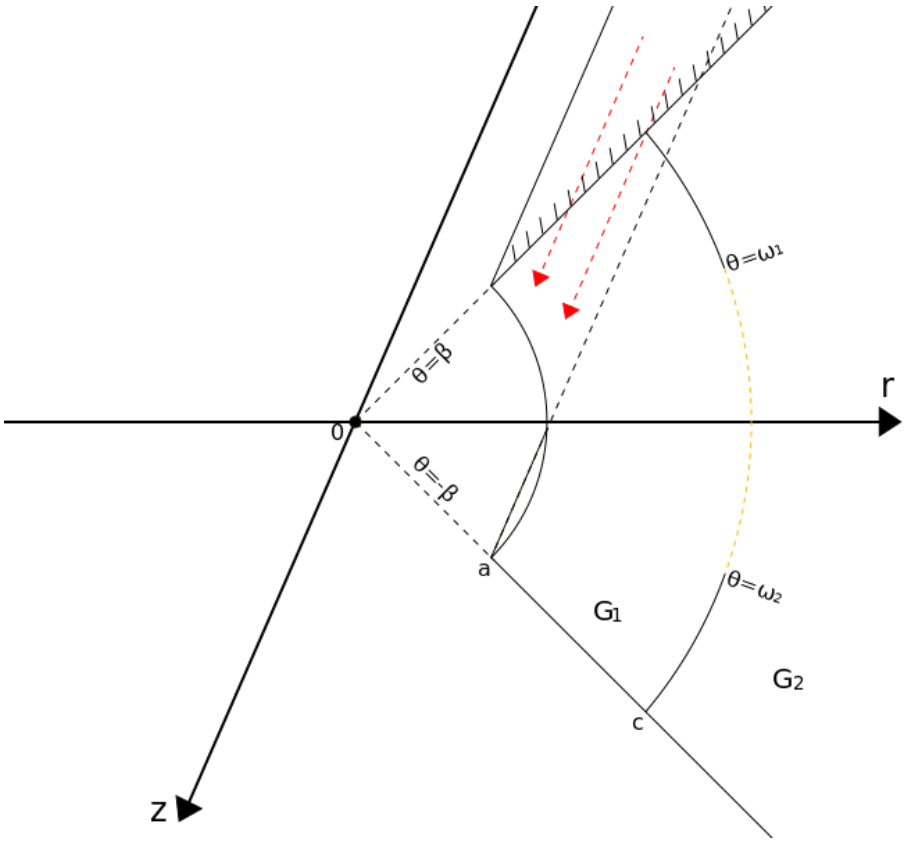

Рис. 1. Схема розглянутого тіла

Мета роботи полягає в знаходженні функцій, які описують переміщення і напруження точок тіла, знаходженні коефіцієнтів інтенсивності напружень в околі вершин дефекту та ефективній імплементації чисельних розрахунків за знайденими формулами на мові Python для візуалізації результатів.

РОЗДІЛ 1

Постановка задачі

Розглянута у вступі задача може бути математично представлена у вигляді наступної крайової.

Функція переміщення точок тіла має задовольняти рівнянню Ламе:

$$r \cdot \frac{\partial}{\partial r}\left(r \cdot \frac{\partial W}{\partial r} \right) + \frac{\partial^2 W}{\partial \theta^2} = 0, \quad a < r < \infty, \quad r \neq c, \quad -\beta < \theta < \beta. \tag{1.1}$$

Також, переміщення та дотичні напруження мають задовольняти наступним крайовим умовам:

$$W\big|_{\theta=\beta} = 0; \quad \mathrm{T}_{\theta z}\big|_{\theta=-\beta} = 0 \Rightarrow \left.\frac{\partial W}{\partial \theta}\right|_{\theta=-\beta} = 0; \tag{1.2}$$

$$\mathrm{T}_{rz}\big|_{r=a} = G_1 \left.\frac{\partial W}{\partial r}\right|_{r=a} = -f(\theta); \quad W, r\frac{\partial W}{\partial r} \xrightarrow[r\to\infty]{} 0; \tag{1.3}$$

та умовам сполучення частин тіла:

$$\langle W \rangle = W\big|_{r=c-0} - W\big|_{r=c+0} = \begin{cases} \varphi(\theta), & \omega_1 < \theta < \omega_2, \\ 0, & \theta \in (-\beta;\ \omega_1) \cup (\omega_2;\ \beta); \end{cases} \tag{1.4}$$

$$\begin{gathered} \langle \mathrm{T}_{rz} \rangle = G_1 \left.\frac{\partial W}{\partial r}\right|_{r=c-0} - G_2 \left.\frac{\partial W}{\partial r}\right|_{r=c+0} = 0; \\ \mathrm{T}_{rz}\big|_{r=c\pm 0} = 0 \ \ (\omega_1 < \theta < \omega_2). \end{gathered} \tag{1.5}$$

Тут $W(r,\theta), \mathrm{T}_{rz}(r,\theta), \mathrm{T}_{\theta z}(r,\theta)$ – функція переміщення точок тіла вздовж осі OZ, радіальне та тангенціальне дотичні напруження відповідно; a, c, β, ω_1, ω_2, G_1, G_2 – відомі константи; $\langle W \rangle$, $\langle \mathrm{T}_{rz} \rangle$ – стрибки відповідних функцій на лінії сполучення частин тіла $(r = c)$; $\varphi(\theta)$ – невідома функція переміщення одного берега тріщини відносно іншого.

РОЗДІЛ 2

Розв’язок

2.1 Зведення до одновимірної задачі

Зведемо спочатку цю задачу до одновимірної за допомогою кінцевого інтегрального перетворення Фур’є за змінною θ:

$$W_n(r)=\int_{-\beta}^{\beta} W(r,\theta)\cos\left(\alpha_n(\theta+\beta)\right)d\theta;\quad \alpha_n=\frac{\pi}{4\beta}(2n-1),\quad n=\overline{1,2\ldots\infty}. \tag{2.1}$$

Формула обернення для цього перетворення:

$$W(r,\theta)=\frac{1}{\beta}\sum_{n=1}^{\infty}W_n(r)\cos\left(\alpha_n(\theta+\beta)\right). \tag{2.2}$$

Застосуємо перетворення (2.1) до задачі (1.1) – (1.5) і отримаємо:

$$r\left(rW_n'(r)\right)'-\alpha_n^2W_n(r)=0,\quad a<r<\infty,\quad r\neq c; \tag{2.3}$$

$$\begin{gathered} W_n'(a)=\frac{-f_n}{G_1},\quad де\ \ f_n=\int_{-\beta}^{\beta} f(\theta)\cos\left(\alpha_n(\theta+\beta)\right)d\theta; \\ W_n(r),\ rW_n'(r)\xrightarrow[r\to\infty]{}0; \end{gathered} \tag{2.4}$$

$$\begin{gathered} \langle W_n\rangle=W_n(c-0)-W_n(c+0)=\varphi_n,\quad де\ \ \varphi_n=\int_{\omega_1}^{\omega_2}\varphi(\theta)\cos\left(\alpha_n(\theta+\beta)\right)d\theta; \\ \left\langle \mathrm{T}_{rz,n}\right\rangle=G_1W_n'(c-0)-G_2W_n'(c+0)=0. \end{gathered} \tag{2.5}$$

(2.3) – (2.5) є розривною одновимірною задачею відносно трансформанти $W_n(r)$.

Будемо шукати її розв’язок у вигляді суми неперервного та розривного розв’язків:

$$W_n(r)=W_n^{нер.}(r)+W_n^{розр.}(r). \tag{2.6}$$

2.2 Побудова розв’язку одновимірної задачі

Неперервний розв’язок є розв’язком рівняння Ейлера (2.3) з крайовими умовами (2.4). Процес розв’язання саме такого рівняння викладено в моїй попередній роботі [1], тому візьмемо розв’язок звідти:

$$W_n^{нер.}(r) = \frac{a}{\alpha_n G_1} f_n \left(\frac{a}{r}\right)^{\alpha_n}. \tag{2.7}$$

Розривний розв’язок будуватимемо наступним чином:

$$W_n^{розр.}(r) = c\left[A_0 \frac{\partial G_n(r,\rho)}{\partial \rho}\bigg|_{\rho=c} - A_1 G_n(r,\rho)\big|_{\rho=c}\right]. \tag{2.8}$$

Тут A_0 – стрибок функції переміщення на лінії сполучення частин тіла, A_1 – стрибок похідної функції переміщення на цій лінії, $G_n(r,\rho)$ – функція Гріна крайової задачі (2.3) – (2.4).

Знов звернемось до моєї попередньої роботи [1], так як процес побудови саме такої функції Гріна там викладено. Звідти:

$$G_n(r,\rho) = \frac{-1}{2\alpha_n}\left(e^{-\alpha_n\left|\ln\frac{r}{\rho}\right|} - \left(\frac{a}{r}\right)^{\alpha_n} e^{-\alpha_n\left|\ln\frac{a}{\rho}\right|} \cdot sign\left(\ln\frac{a}{\rho}\right)\right). \tag{2.9}$$

Тепер знайдемо похідну цієї функції за змінною ρ:

$$\frac{\partial G_n(r,\rho)}{\partial \rho} = \frac{-1}{2\rho}\left(e^{-\alpha_n\left|\ln\frac{r}{\rho}\right|} \cdot sign\left(\ln\frac{r}{\rho}\right) - \left(\frac{a}{r}\right)^{\alpha_n} \cdot e^{-\alpha_n\left|\ln\frac{a}{\rho}\right|}\right). \tag{2.10}$$

Тоді, при $\rho = c$ отримаємо:

$$G_n(r,\rho)\Big|_{\rho=c} = \frac{-1}{2\alpha_n}\left(e^{-\alpha_n\left|\ln\frac{r}{c}\right|} + \left(\frac{a}{r}\right)^{\alpha_n} e^{\alpha_n \ln\frac{a}{c}} \right),$$
$$\frac{\partial G_n(r,\rho)}{\partial\rho}\Big|_{\rho=c} = \frac{-1}{2c}\left(e^{-\alpha_n\left|\ln\frac{r}{c}\right|} \cdot sign\left(\ln\frac{r}{c}\right) - \left(\frac{a}{r}\right)^{\alpha_n} \cdot e^{\alpha_n \ln\frac{a}{c}} \right). \qquad (2.11)$$

Далі, оскільки A_0 це стрибок $W_n(r)$ на лінії $r=c$, з першої умови в (2.5) нам відомо, що $A_0 = \langle W_n \rangle = \varphi_n$. З другої умови (2.5) знайдемо A_1, процес пошуку аналогічний викладеному в [1]. Отримаємо:

$$A_1 = G^* W_n'(c+0), \quad \text{де } G^* = \frac{G_2 - G_1}{G_1}. \qquad (2.12)$$

Поки залишимо цей коефіцієнт у вигляді A_1 і підставимо все інше у розв'язок (2.6) одновимірної задачі. Врахуємо, що за умовою $\frac{a}{c} > 0$, тому $e^{\alpha_n \ln\frac{a}{c}} = \left(\frac{a}{c}\right)^{\alpha_n}$.

Отримаємо:

$$W_n(r) = \frac{a}{\alpha_n G_1} f_n \left(\frac{a}{r}\right)^{\alpha_n} + \frac{-\varphi_n}{2}\left(e^{-\alpha_n\left|\ln\frac{r}{c}\right|} \cdot sign\left(\ln\frac{r}{c}\right) - \left(\frac{a}{r}\right)^{\alpha_n}\left(\frac{a}{c}\right)^{\alpha_n} \right) +$$
$$+ \frac{A_1 c}{2\alpha_n}\left(e^{-\alpha_n\left|\ln\frac{r}{c}\right|} + \left(\frac{a}{r}\right)^{\alpha_n}\left(\frac{a}{c}\right)^{\alpha_n} \right). \qquad (2.13)$$

Знайдемо тепер похідну цього розв'язку:

$$W_n'(r) = \frac{-a}{G_1 r} f_n \left(\frac{a}{r}\right)^{\alpha_n} + \frac{a_n \varphi_n}{2r}\left(e^{-\alpha_n\left|\ln\frac{r}{c}\right|} - \left(\frac{a}{c}\right)^{\alpha_n}\left(\frac{a}{r}\right)^{\alpha_n} \right) -$$
$$- \frac{A_1 c}{2r}\left(e^{-\alpha_n\left|\ln\frac{r}{c}\right|} \cdot sign\left(\ln\frac{r}{c}\right) + \left(\frac{a}{c}\right)^{\alpha_n}\left(\frac{a}{r}\right)^{\alpha_n} \right). \qquad (2.14)$$

Положимо в (2.14) $r = c + 0$. Тоді $\ln\left(\frac{r}{c}\right) \to +0$, $sign\left(\ln\frac{r}{c}\right) = 1$. Отримаємо:

$$W_n'(c+0)=\frac{-a}{G_1 c}\,f_n\left(\frac{a}{c}\right)^{\alpha_n}+\frac{\alpha_n\varphi_n}{2c}\left(1-\left(\frac{a}{c}\right)^{2\alpha_n}\right)+\frac{-A_1}{2}\left(1+\left(\frac{a}{c}\right)^{2\alpha_n}\right). \qquad (2.15)$$

Підставивши в (2.15) вираз для A_1 з (2.12) отримаємо рівняння відносно $W_n'(c+0)$. Виразимо цю похідну з отриманого рівняння:

$$W_n'(c+0)\left[1+\frac{G^*}{2}\left(1+\left(\frac{a}{c}\right)^{2\alpha_n}\right)\right]=\frac{-a}{G_1 c}\,f_n\left(\frac{a}{c}\right)^{\alpha_n}+\frac{\alpha_n\varphi_n}{2c}\left(1-\left(\frac{a}{c}\right)^{2\alpha_n}\right). \qquad (2.16)$$

Позначимо в (2.16) $\Delta_n=\left[1+\frac{G^*}{2}\left(1+\left(\frac{a}{c}\right)^{2\alpha_n}\right)\right]$ і розділимо на нього рівняння:

$$W_n'(c+0)=\frac{-a}{G_1 c\Delta_n}\,f_n\left(\frac{a}{c}\right)^{\alpha_n}+\frac{\alpha_n\varphi_n}{2c\Delta_n}\left(1-\left(\frac{a}{c}\right)^{2\alpha_n}\right). \qquad (2.17)$$

Нарешті, знаючи вираз для $W_n'(c+0)$, знайдемо вираз для A_1:

$$A_1=G^*W_n'(c+0)=G^*\left(\frac{-a}{G_1 c\Delta_n}\,f_n\left(\frac{a}{c}\right)^{\alpha_n}+\frac{\alpha_n\varphi_n}{2c\Delta_n}\left(1-\left(\frac{a}{c}\right)^{2\alpha_n}\right)\right). \qquad (2.18)$$

Підставивши цей вираз в (2.13) отримаємо остаточний вигляд розв’язку одновимірного рівняння:

$$\begin{aligned}W_n(r)=&\frac{a}{\alpha_n G_1}\,f_n\left(\frac{a}{r}\right)^{\alpha_n}+\frac{-\varphi_n}{2}\left(e^{-\alpha_n\left|\ln\frac{r}{c}\right|}\cdot sign\left(\ln\frac{r}{c}\right)-\left(\frac{a}{r}\right)^{\alpha_n}\left(\frac{a}{c}\right)^{\alpha_n}\right)+\\&+\frac{G^*}{2\alpha_n}\left(\frac{-a}{G_1\Delta_n}\,f_n\left(\frac{a}{c}\right)^{\alpha_n}+\frac{\alpha_n\varphi_n}{2\Delta_n}\left(1-\left(\frac{a}{c}\right)^{2\alpha_n}\right)\right)\left(e^{-\alpha_n\left|\ln\frac{r}{c}\right|}+\left(\frac{a}{r}\right)^{\alpha_n}\left(\frac{a}{c}\right)^{\alpha_n}\right).\end{aligned} \qquad (2.19)$$

Невідомими у цьому виразі залишаються тільки числа φ_n. Побудуємо рівняння для їх знаходження.

2.2 Побудова рівняння для знаходження $\varphi(\theta)$

Рівняння для знаходження $\varphi(\theta)$ буде отримано з наступної умови:

$$\mathrm{T}_{rz}\big|_{r=c\pm 0} = 0 \Rightarrow \left.\frac{\partial W}{\partial r}\right|_{r=c\pm 0} = \left[\frac{1}{\beta}\sum_{n=1}^{\infty} W_n'(r)\cos\left(\alpha_n(\theta+\beta)\right)\right]\Bigg|_{r=c\pm 0} = 0. \qquad (2.20)$$

Почнемо з підстановки виразу (2.18) для A_1 в похідну розв'язку одновимірної задачі (2.14). Трохи перетворивши, отримаємо:

$$\begin{aligned} W_n'(r) = {} & \frac{-a}{G_1 r} f_n \left(\frac{a}{r}\right)^{\alpha_n} + \frac{a_n \varphi_n}{2r}\left(e^{-\alpha_n\left|\ln\frac{r}{c}\right|} - \left(\frac{a}{c}\right)^{\alpha_n}\left(\frac{a}{r}\right)^{\alpha_n}\right) + \\ & + \frac{G^*}{2r\Delta_n}\cdot\frac{a}{G_1} f_n \left(\frac{a}{c}\right)^{\alpha_n}\cdot\left(e^{-\alpha_n\left|\ln\frac{r}{c}\right|}\cdot sign\left(\ln\frac{r}{c}\right) + \left(\frac{a}{c}\right)^{\alpha_n}\left(\frac{a}{r}\right)^{\alpha_n}\right) + \\ & + \frac{G^*}{2r\Delta_n}\cdot\frac{\alpha_n\varphi_n}{2}\left(\left(\frac{a}{c}\right)^{2\alpha_n} - 1\right)\left(e^{-\alpha_n\left|\ln\frac{r}{c}\right|}\cdot sign\left(\ln\frac{r}{c}\right) + \left(\frac{a}{c}\right)^{\alpha_n}\left(\frac{a}{r}\right)^{\alpha_n}\right). \end{aligned} \qquad (2.21)$$

Підставимо це в вираз із (2.20), розділимо на дві частини та спростимо:

$$\begin{aligned} \frac{\partial W}{\partial r} = {} & \frac{1}{\beta}\sum_{n=1}^{\infty} W_n'(r)\cos\left(\alpha_n(\theta+\beta)\right) = \\ = {} & \frac{-a}{\beta G_1 r}\sum_{n=1}^{\infty}\Bigg[f_n\left\{\left(\frac{a}{r}\right)^{\alpha_n} - \frac{G^*}{2\Delta_n}\left(\frac{a}{c}\right)^{\alpha_n}\left(e^{-\alpha_n\left|\ln\frac{r}{c}\right|}\cdot sign\left(\ln\frac{r}{c}\right) + \left(\frac{a}{c}\right)^{\alpha_n}\left(\frac{a}{r}\right)^{\alpha_n}\right)\right\}\cdot \\ & \cdot\cos\left(\alpha_n(\theta+\beta)\right)\Bigg] + \\ & + \frac{1}{2\beta r}\sum_{n=1}^{\infty}\Bigg[\alpha_n\varphi_n\left\{e^{-\alpha_n\left|\ln\frac{r}{c}\right|} - \left(\frac{a}{c}\right)^{\alpha_n}\left(\frac{a}{r}\right)^{\alpha_n} + \frac{G^*}{2\Delta_n}\left(\left(\frac{a}{c}\right)^{2\alpha_n} - 1\right)\cdot\right. \\ & \left.\cdot\left(e^{-\alpha_n\left|\ln\frac{r}{c}\right|}\cdot sign\left(\ln\frac{r}{c}\right) + \left(\frac{a}{c}\right)^{\alpha_n}\left(\frac{a}{r}\right)^{\alpha_n}\right)\right\}\cdot\cos\left(\alpha_n(\theta+\beta)\right)\Bigg]. \end{aligned} \qquad (2.22)$$

Завдяки такому розділенню, перший ряд з (2.22) сходиться абсолютно й рівномірно, так як $\frac{a}{r}<1,\ \frac{a}{c}<1,\ \Delta_n \xrightarrow[n\to\infty]{} 1+\frac{G^*}{2}$. Тому перейдемо в цьому ряду к границі при $r \to c+0$. Отриманий результат позначимо $H(\theta)$:

$$H(\theta)=\frac{-a}{\beta G_1 c}\sum_{n=1}^{\infty}\left[f_n\left(\frac{a}{c}\right)^{\alpha_n}\left\{1-\frac{G^*}{2\Delta_n}\left(1+\left(\frac{a}{c}\right)^{2\alpha_n}\right)\right\}\cos\left(\alpha_n(\theta+\beta)\right)\right]. \qquad (2.23)$$

В другому ряду з (2.22) підставимо

$$\varphi_n=\int_{\omega_1}^{\omega_2}\varphi(\eta)\cos\left(\alpha_n(\eta+\beta)\right)d\eta \qquad (2.24)$$

та змінимо порядок підсумовування та інтегрування, вважаючи, що $r \neq c$, щоб ряд сходився рівномірно. Позначимо результат як P_2:

$$P_2=\frac{1}{2\beta r}\int_{\omega_1}^{\omega_2}\left[\varphi(\eta)\left\{\sum_{n=1}^{\infty}\left[e^{-\alpha_n\left|\ln\frac{r}{c}\right|}-\left(\frac{a^2}{rc}\right)^{\alpha_n}+\frac{G^*}{2\Delta_n}\left(\left(\frac{a}{c}\right)^{2\alpha_n}-1\right)\cdot\right.\right.$$
$$\left.\left.\cdot\left(e^{-\alpha_n\left|\ln\frac{r}{c}\right|}\cdot sign\left(\ln\frac{r}{c}\right)+\left(\frac{a^2}{rc}\right)^{\alpha_n}\right)\alpha_n\cos\left(\alpha_n(\theta+\beta)\right)\cos\left(\alpha_n(\eta+\beta)\right)\right]\right\}d\eta\right]. \qquad (2.25)$$

Розіб’ємо цей ряд на дві частини, виділивши частину, що слабко сходиться:

$$P_2=\frac{1}{2\beta r}\int_{\omega_1}^{\omega_2}\left\{\varphi(\eta)\sum_{n=1}^{\infty}\left[e^{-\alpha_n\left|\ln\frac{r}{c}\right|}\left(1-\frac{G^*}{2\Delta_n}sign\left(\ln\frac{r}{c}\right)\right)\alpha_n\cdot\right.\right.$$
$$\left.\left.\cdot\cos\left(\alpha_n(\theta+\beta)\right)\cos\left(\alpha_n(\eta+\beta)\right)\right]d\eta\right\}-$$
$$-\frac{1}{2\beta r}\int_{\omega_1}^{\omega_2}\left\{\varphi(\eta)\sum_{n=1}^{\infty}\left[\left(\frac{a^2}{rc}\right)^{\alpha_n}-\frac{G^*}{2\Delta_n}\left(e^{-\alpha_n\left|\ln\frac{r}{c}\right|}sign\left(\ln\frac{r}{c}\right)\left(\frac{a}{c}\right)^{2\alpha_n}+\left(\frac{a^2}{rc}\right)^{\alpha_n}\cdot\right.\right.\right. \qquad (2.26)$$
$$\left.\left.\left.\cdot\left[\left(\frac{a}{c}\right)^{2\alpha_n}-1\right]\right)\alpha_n\cos\left(\alpha_n(\theta+\beta)\right)\cos\left(\alpha_n(\eta+\beta)\right)\right]d\eta\right\}.$$

В другому інтегралі з (2.26) ряд сходиться рівномірно. Перейдемо там до границі при $r \to c+0$ і позначимо результат $P_2 I_2$, щоб не загубитися:

$$P_2 I_2(c+0) = \frac{-1}{2\beta c}\int_{\omega_1}^{\omega_2}\left\{\varphi(\eta)\sum_{n=1}^{\infty}\left[\left(\frac{a}{c}\right)^{2\alpha_n}\left[1-\frac{G^*}{2\Delta_n}\left(\frac{a}{c}\right)^{2\alpha_n}\right]\alpha_n \cdot \right.\right. \\ \left.\left. \cdot\cos\left(\alpha_n(\theta+\beta)\right)\cos\left(\alpha_n(\eta+\beta)\right)\right]d\eta\right\}. \tag{2.27}$$

Позначимо тут

$$K(\theta,\eta) = \sum_{n=1}^{\infty}\left(\frac{a}{c}\right)^{2\alpha_n}\left[1-\frac{G^*}{2\Delta_n}\left(\frac{a}{c}\right)^{2\alpha_n}\right]\alpha_n\cos\left(\alpha_n(\theta+\beta)\right)\cos\left(\alpha_n(\eta+\beta)\right). \tag{2.28}$$

Повернемось тепер до першого інтегралу з (2.26). Ряд в ньому розходитиметься при $r \to c+0$, тому спробуємо виділити з нього частину, яка сходитиметься рівномірно. Для цього спочатку помітимо, що

$$\frac{G^*}{2\Delta_n} = \frac{G^*}{2+G^*\left(1+\left(\frac{a}{c}\right)^{2\alpha_n}\right)} \xrightarrow[\alpha_n\to\infty]{} \frac{G^*}{G^*+2}. \tag{2.29}$$

Якщо відняти та додати праву частину (2.29) до лівої, можна показати, що

$$\frac{G^*}{2\Delta_n} = \frac{G^*}{G^*+2} + \left(\frac{G^*}{2\Delta_n} - \frac{G^*}{G^*+2}\right) = \cdots = \frac{G^*}{G^*+2} - \frac{\left(G^*\right)^2\left(\frac{a}{c}\right)^{2\alpha_n}}{2\Delta_n\left(G^*+2\right)}. \tag{2.30}$$

Підставимо тепер результат (2.30) до ряду з першого інтегралу (2.26) і перетворимо, вважаючи, що $r \neq c$. Позначимо цей вираз $P_2 P_1$:

$$P_2P_1=\left[1-\frac{G^*}{G^*+2}sign\left(\ln\frac{r}{c}\right)\right]\cdot$$
$$\cdot\sum_{n=1}^{\infty}\left[e^{-\alpha_n\left|\ln\frac{r}{c}\right|}\cdot\alpha_n\cos\left(\alpha_n(\theta+\beta)\right)\cos\left(\alpha_n(\eta+\beta)\right)\right]+$$
$$+\frac{\left(G^*\right)^2}{2G^*+4}\cdot$$
$$\cdot\sum_{n=1}^{\infty}\left[\frac{1}{\Delta_n}\left(\frac{a}{c}\right)^{2\alpha_n}sign\left(\ln\frac{r}{c}\right)e^{-\alpha_n\left|\ln\frac{r}{c}\right|}\cdot\alpha_n\cos\left(\alpha_n(\theta+\beta)\right)\cos\left(\alpha_n(\eta+\beta)\right)\right].\quad(2.31)$$

Завдяки множнику $\left(\frac{a}{c}\right)^{2\alpha_n}$, другий ряд з (2.31) сходиться рівномірно при всіх r.

Перейдемо в ньому до границі при $r\to c+0$. Для зручності, продовжимо вводити ієрархічні позначення.

$$P_2P_1P_2(c+0)=\frac{\left(G^*\right)^2}{2G^*+4}\sum_{n=1}^{\infty}\frac{1}{\Delta_n}\left(\frac{a}{c}\right)^{2\alpha_n}\alpha_n\cos\left(\alpha_n(\theta+\beta)\right)\cos\left(\alpha_n(\eta+\beta)\right).\quad(2.32)$$

В першому ряду з (2.31), досі вважаючи, що $r\neq c$, скористаємося наступною тотожністю

$$\cos\left(\alpha_n(\theta+\beta)\right)=\frac{-1}{\alpha_n^2}\frac{d^2}{d\theta^2}\cos\left(\alpha_n(\theta+\beta)\right),\quad(2.33)$$

а також змінимо порядок інтегрування та підсумовування. Отримаємо:

$$P_2P_1P_1=\left[\frac{G^*}{G^*+2}sign\left(\ln\frac{r}{c}\right)-1\right]\cdot$$
$$\cdot\frac{\partial^2}{\partial\theta^2}\sum_{n=1}^{\infty}e^{-\alpha_n\left|\ln\frac{r}{c}\right|}\frac{\cos\left(\alpha_n(\theta+\beta)\right)\cos\left(\alpha_n(\eta+\beta)\right)}{\alpha_n}.\quad(2.34)$$

Розіб’ємо ряд з (2.34) на дві частини перетворивши добуток косинусів на суму. Також підставимо $\alpha_n=\frac{\pi}{4\beta}(2n-1)$. Отримаємо:

$$P_2P_1P_1=\left[\frac{G^*}{G^*+2}sign\left(\ln\frac{r}{c}\right)-1\right]\frac{2\beta}{\pi}\cdot$$
$$\cdot\frac{\partial^2}{\partial\theta^2}\left[\sum_{n=1}^{\infty}\frac{\cos\left[\frac{\pi}{4\beta}(2n-1)(\theta-\eta)\right]}{2n-1}e^{\frac{-\pi}{4\beta}\left|\ln\frac{r}{c}\right|(2n-1)}+\right. \tag{2.35}$$
$$\left.+\sum_{n=1}^{\infty}\frac{\cos\left[\frac{\pi}{4\beta}(2n-1)(2\beta+\theta+\eta)\right]}{2n-1}e^{\frac{-\pi}{4\beta}\left|\ln\frac{r}{c}\right|(2n-1)}\right].$$

Для отриманих рядів доцільно скористатися наступною формулою з довідника А.П. Прудніков а «Інтеграли та ряди»:

$$\sum_{n=1}^{\infty}\frac{\cos\left(x(2n-1)\right)}{(2n-1)}e^{-a(2n-1)}=\frac{1}{4}\ln\left(\mathrm{ch}(a)+\cos(x)\right)-\frac{1}{4}\ln\left(\mathrm{ch}(a)-\cos(x)\right). \tag{2.36}$$

Перший ряд з (2.35) відповідає цій формулі при $x=\frac{\pi}{4\beta}(\theta-\eta),\ a=\frac{\pi}{4\beta}\left|\ln\frac{r}{c}\right|$. Для другого ряду з (2.35) це $x=\frac{\pi}{4\beta}(2\beta+\theta+\eta),\ a=\frac{\pi}{4\beta}\left|\ln\frac{r}{c}\right|$. Використаємо цю формулу для обох рядів та перейдемо до границі при $r\to c+0$.

$$P_2P_1P_1(c+0)=\left[\frac{G^*}{G^*+2}-1\right]\frac{\beta}{2\pi}\cdot$$
$$\cdot\frac{\partial^2}{\partial\theta^2}\left\{\ln\left[1+\cos\left(\frac{\pi}{4\beta}(\theta-\eta)\right)\right]-\ln\left[1-\cos\left(\frac{\pi}{4\beta}(\theta-\eta)\right)\right]+\right. \tag{2.37}$$
$$\left.+\ln\left[1+\cos\left(\frac{\pi}{4\beta}(2\beta+\theta+\eta)\right)\right]-\ln\left[1-\cos\left(\frac{\pi}{4\beta}(2\beta+\theta+\eta)\right)\right]\right\}.$$

Таким чином, ми привели кожну частину (2.22) до границі при $r\to c+0$. За умовою (2.20) відомо, що $\left.\frac{\partial W}{\partial r}\right|_{r\to c+0}=0,$ тому, об'єднавши все отримане та трохи спростивши, ми приходимо до наступного інтегрального рівняння:

$$\frac{1}{2\pi c\left(G^*+2\right)}\cdot\frac{d^2}{d\theta^2}\int_{\omega_1}^{\omega_2}\varphi(\eta)\ln\left[2\sin^2\left(\frac{\pi}{8\beta}(\theta-\eta)\right)\right]d\eta\ +$$

$$+\frac{-1}{2\pi c\left(G^*+2\right)}\cdot\frac{d^2}{d\theta^2}\int_{\omega_1}^{\omega_2}\left\{\varphi(\eta)\left(\ln\left[2\cos^2\left(\frac{\pi}{8\beta}(\theta-\eta)\right)\right]+\right.\right.$$

$$\left.\left.+\ln\left[2\cos^2\left(\frac{\pi}{8\beta}(\theta+\eta+2\beta)\right)\right]-\ln\left[2\sin^2\left(\frac{\pi}{8\beta}(\theta+\eta+2\beta)\right)\right]\right)d\eta\right\}+ \quad (2.38)$$

$$+\frac{\left(G^*\right)^2}{2G^*+4}\cdot\frac{1}{2\beta c}\int_{\omega_1}^{\omega_2}\varphi(\eta)\sum_{n=1}^{\infty}\frac{\alpha_n}{\Delta_n}\left(\frac{a}{c}\right)^{2\alpha_n}\cos\left(\alpha_n(\theta+\beta)\right)\cos\left(\alpha_n(\eta+\beta)\right)d\eta+$$

$$+\frac{-1}{2\beta c}\int_{\omega_1}^{\omega_2}\varphi(\eta)K(\theta,\eta)\,d\eta=-H(\theta);\quad \omega_1<\theta<\omega_2.$$

Тут перший інтеграл має сингулярне ядро, що дає особливість при $\eta=\theta$, а регулярні ядра інших трьох об'єднаємо в одне, спочатку помноживши обидві частини на $2\pi c\left(G^*+2\right)$. Отримаємо:

$$\frac{d^2}{d\theta^2}\int_{\omega_1}^{\omega_2}\varphi(\eta)\ln\left[2\sin^2\left(\frac{\pi}{8\beta}(\theta-\eta)\right)\right]d\eta-\int_{\omega_1}^{\omega_2}\varphi(\eta)R(\theta,\eta)\,d\eta=$$
$$=-2\pi c\left(G^*+2\right)H(\theta), \quad (2.39)$$

де

$$R(\theta,\eta)=\frac{d^2}{d\theta^2}\left\{\ln\left[2\cos^2\left(\frac{\pi}{8\beta}(\theta-\eta)\right)\right]+\right.$$

$$\left.+\ln\left[2\cos^2\left(\frac{\pi}{8\beta}(\theta+\eta+2\beta)\right)\right]-\ln\left[2\sin^2\left(\frac{\pi}{8\beta}(\theta+\eta+2\beta)\right)\right]\right\}-$$

$$-\frac{\pi\left(G^*\right)^2}{2\beta}\sum_{n=1}^{\infty}\left\{\frac{\alpha_n}{\Delta_n}\left(\frac{a}{c}\right)^{2\alpha_n}\cos\left(\alpha_n(\theta+\beta)\right)\cos\left(\alpha_n(\eta+\beta)\right)\right\}+ \quad (2.40)$$

$$+\frac{\pi\left(G^*+2\right)}{\beta}K(\theta,\eta).$$

Зробимо тепер заміну змінних, яка зводить проміжок інтегрування $(\omega_1;\omega_2)$ до проміжку $(-1;1)$.

$$\theta=\frac{\omega_2-\omega_1}{2}t+\frac{\omega_2+\omega_1}{2};\quad \eta=\frac{\omega_2-\omega_1}{2}\xi+\frac{\omega_2+\omega_1}{2};$$
$$d\theta=\frac{\omega_2-\omega_1}{2}dt;\quad d\eta=\frac{\omega_2-\omega_1}{2}d\xi. \tag{2.41}$$

Зробивши цю заміну в (2.39), отримаємо:

$$\frac{d^2}{dt^2}\int_{-1}^{1}\varphi^*(\xi)\ln\left[2\sin^2\left(\frac{\pi}{16\beta}(\omega_2-\omega_1)(t-\xi)\right)\right]d\xi-$$
$$-\left(\frac{\omega_2-\omega_1}{2}\right)^2\int_{-1}^{1}\varphi^*(\xi)R^*(t,\xi)d\xi=-\pi c(\omega_2-\omega_1)\left(G^*+2\right)H^*(t), \tag{2.42}$$

де

$$\varphi^*(\xi)=\varphi\left(\frac{\omega_2-\omega_1}{2}\xi+\frac{\omega_2+\omega_1}{2}\right);$$
$$H^*(t)=H\left(\frac{\omega_2-\omega_1}{2}t+\frac{\omega_2+\omega_1}{2}\right); \tag{2.43}$$
$$R^*(t,\xi)=R\left(\frac{\omega_2-\omega_1}{2}t+\frac{\omega_2+\omega_1}{2},\frac{\omega_2-\omega_1}{2}\xi+\frac{\omega_2+\omega_1}{2}\right).$$

Далі, перетворимо сингулярне ядро з (2.42). Виділимо з нього $(t-\xi)$:

$$\ln\left[2\sin^2\left(\frac{\pi}{16\beta}(\omega_2-\omega_1)(t-\xi)\right)\right]=$$
$$=-\ln\left(\frac{1}{|t-\xi|^2}\right)+\left(\ln\left[2\sin^2\left(\frac{\pi}{16\beta}(\omega_2-\omega_1)(t-\xi)\right)\right]+\ln\left(\frac{1}{|t-\xi|^2}\right)\right)= \tag{2.44}$$
$$=-2\ln\left(\frac{1}{|t-\xi|}\right)+\ln\left[\frac{2\sin^2\left(\frac{\pi}{16\beta}(\omega_2-\omega_1)(t-\xi)\right)}{(t-\xi)^2}\right].$$

Тут другий доданок не має особливості, так як

$$\lim_{|t-\xi|\to 0}\left(\ln\left[\frac{2\sin^2\left(\frac{\pi}{16\beta}(\omega_2-\omega_1)(t-\xi)\right)}{(t-\xi)^2}\right]\right)=\ln\left[2\left(\frac{\pi}{16\beta}(\omega_2-\omega_1)\right)^2\right]. \quad (2.45)$$

Підставивши (2.44) в (2.42), помножимо обидві частини на $-\frac{1}{2}$ і занесемо другий доданок (2.45) до регулярного ядра, включно з константою перед другим інтегралом. Отримаємо:

$$\begin{aligned}&\frac{d^2}{dt^2}\int_{-1}^{1}\varphi^*(\xi)\ln\left(\frac{1}{|t-\xi|}\right)d\xi+\int_{-1}^{1}\varphi^*(\xi)M(t,\xi)d\xi=\\&=\frac{\pi}{2}c(\omega_2-\omega_1)(G^*+2)H^*(t);\quad -1<t<1.\end{aligned} \quad (2.46)$$

Тут нове регулярне ядро:

$$\begin{aligned}&M(t,\xi)=\\&=\frac{-1}{2}\frac{d^2}{dt^2}\left(\ln\left[\frac{2\sin^2\left(\frac{\pi}{16\beta}(\omega_2-\omega_1)(t-\xi)\right)}{(t-\xi)^2}\right]\right)+\frac{1}{2}\left(\frac{\omega_2-\omega_1}{2}\right)^2R^*(t,\xi).\end{aligned} \quad (2.47)$$

Отримали інтегро-диференціальне рівняння (2.46), до якого можна застосувати метод ортогональних многочленів.

2.3 Розв’язок рівняння методом ортогональних многочленів

Перейдемо до розв’язку (2.46). Існує відоме спектральне співвідношення:

$$\frac{d^2}{dt^2}\int_{-1}^{1}\ln\left(\frac{1}{|t-\xi|}\right)\sqrt{1-\xi^2}U_n(\xi)d\xi=-\pi(n+1)U_n(t), \quad (2.48)$$

де $U_n(t)$ – многочлени Чебишева другого роду. Це співвідношення дозволяє шукати розв'язок рівняння (2.46) у вигляді розвинення за многочленами Чебишева:

$$\varphi^*(\xi)=\sqrt{1-\xi^2}\sum_{n=0}^{\infty}\varphi_n^* U_n(\xi). \tag{2.49}$$

Підставимо це розвинення в (2.46) та змінимо порядок інтегрування та підсумовування:

$$\begin{gathered}\sum_{n=0}^{\infty}\left[\varphi_n^*\frac{d^2}{dt^2}\int_{-1}^{1}\ln\left(\frac{1}{|t-\xi|}\right)\sqrt{1-\xi^2}U_n(\xi)d\xi\right]+\\ +\sum_{n=0}^{\infty}\left[\varphi_n^*\int_{-1}^{1}M(t,\xi)\sqrt{1-\xi^2}U_n(\xi)d\xi\right]=\frac{\pi}{2}c(\omega_2-\omega_1)(G^*+2)H^*(t).\end{gathered} \tag{2.50}$$

Використавши спектральне співвідношення (2.48) в першій сумі з (2.50) отримаємо:

$$\begin{gathered}-\pi\sum_{n=0}^{\infty}\left[\varphi_n^*(n+1)U_n(t)\right]+\sum_{n=0}^{\infty}\left[\varphi_n^*\int_{-1}^{1}M(t,\xi)\sqrt{1-\xi^2}U_n(\xi)d\xi\right]=\\ =\frac{\pi}{2}c(\omega_2-\omega_1)(G^*+2)H^*(t).\end{gathered} \tag{2.51}$$

Помножимо обидві частини (2.51) на $\sqrt{1-t^2}U_m(t)$ та візьмемо інтеграл від обох частин за змінною t на проміжку $(-1;1)$:

$$\begin{gathered}-\pi\sum_{n=0}^{\infty}\varphi_n^*(n+1)\int_{-1}^{1}U_n(t)U_m(t)\sqrt{1-t^2}dt\ +\\ +\sum_{n=0}^{\infty}\varphi_n^*\int_{-1}^{1}\left[\sqrt{1-t^2}U_m(t)dt\int_{-1}^{1}M(t,\xi)\sqrt{1-\xi^2}U_n(\xi)d\xi\right]=\\ =\frac{\pi}{2}c(\omega_2-\omega_1)(G^*+2)\int_{-1}^{1}H^*(t)U_m(t)\sqrt{1-t^2}dt.\end{gathered} \tag{2.52}$$

Тепер ми можемо скористатися властивістю ортогональності многочленів Чебишева другого роду:

$$\int_{-1}^{1}\sqrt{1-t^2}U_n(t)U_m(t)dt=\frac{\pi}{2}\delta_{nm},\quad \partial e\ \ \delta_{nm}=\begin{cases}1,\ n=m;\\ 0,\ n\neq m.\end{cases} \tag{2.53}$$

Враховуючи це в (2.52), отримаємо:

$$\begin{gathered}\frac{-\pi^2}{2}(m+1)\varphi_m^*+\sum_{n=0}^{\infty}\varphi_n^*\int_{-1}^{1}\left\{\sqrt{1-t^2}U_m(t)dt\int_{-1}^{1}M(t,\xi)\sqrt{1-\xi^2}U_n(\xi)d\xi\right\}=\\ =\frac{\pi}{2}c(\omega_2-\omega_1)(G^*+2)\int_{-1}^{1}H^*(t)U_m(t)\sqrt{1-t^2}dt;\quad m=\overline{0,1,2\ldots\infty}.\end{gathered} \tag{2.54}$$

Помножимо тепер обидві частини рівняння на $\frac{-2}{\pi^2\sqrt{m+1}}$ та «створимо» під сумою множник $\sqrt{n+1}\cdot\frac{1}{\sqrt{n+1}}$:

$$\begin{gathered}\sqrt{m+1}\cdot\varphi_m^*+\frac{-2}{\pi^2}\sum_{n=0}^{\infty}\Bigg[\sqrt{n+1}\cdot\varphi_n^*\cdot\\ \cdot\frac{1}{\sqrt{(m+1)(n+1)}}\int_{-1}^{1}\sqrt{1-t^2}U_m(t)dt\int_{-1}^{1}M(t,\xi)\sqrt{1-\xi^2}U_n(\xi)d\xi\Bigg]=\\ =\frac{-c(\omega_2-\omega_1)(G^*+2)}{\pi\sqrt{m+1}}\int_{-1}^{1}H^*(t)\sqrt{1-t^2}U_m(t)dt.\end{gathered} \tag{2.55}$$

Введемо в (2.55) наступні позначення:

$$\begin{gathered}A_{mn}=\frac{-2}{\pi^2\sqrt{(m+1)(n+1)}}\int_{-1}^{1}\sqrt{1-t^2}U_m(t)dt\int_{-1}^{1}M(t,\xi)\sqrt{1-\xi^2}U_n(\xi)d\xi,\\ B_m=\frac{-c(\omega_2-\omega_1)(G^*+2)}{\pi\sqrt{m+1}}\int_{-1}^{1}H^*(t)\sqrt{1-t^2}U_m(t)dt,\\ X_m=\sqrt{m+1}\cdot\varphi_m^*,\quad X_n=\sqrt{n+1}\cdot\varphi_n^*.\end{gathered} \tag{2.56}$$

З ними (2.55) перетворюється на наступну нескінченну систему лінійних алгебраїчних рівнянь:

$$X_m + \sum_{n=0}^{\infty} A_{mn} \cdot X_n = B_m; \quad m = \overline{0,1,2\ldots\infty}. \tag{2.57}$$

Розв'язувати її будемо за допомогою метода редукції. Для обчислення інтегралів, що містяться у виразах для A_{mn} та B_m, скористаємось наступною квадратурною формулою Чебишева:

$$\int_{-1}^{1} F(x)\sqrt{1-x^2}\,dx = \sum_{i=1}^{N} L_i F(x_i),$$
$$\text{де} \quad L_i = \frac{\pi}{N+1}\sin^2\left(\frac{\pi i}{N+1}\right); \quad x_i = \cos\left(\frac{\pi i}{N+1}\right). \tag{2.58}$$

Враховуючи, що

$$U_k(x_i) = \frac{\sin\left((k+1)\arccos(x_i)\right)}{\sin\left(\arccos(x_i)\right)} = \frac{\sin\left((k+1)\dfrac{\pi i}{N+1}\right)}{\sin\left(\dfrac{\pi i}{N+1}\right)}, \tag{2.59}$$

отримаємо наступну квадратурну формулу:

$$\int_{-1}^{1} F(x)\sqrt{1-x^2}U_k(x)\,dx =$$
$$= \frac{\pi}{N+1}\sum_{i=1}^{N}\sin\left(\frac{\pi i}{N+1}\right)\sin\left((k+1)\frac{\pi i}{N+1}\right)F\left(\cos\left(\frac{\pi i}{N+1}\right)\right). \tag{2.60}$$

Тоді, застосувавши це до інтегралів в A_{mn}, B_m, отримаємо:

$$A_{mn} = \frac{-2}{(N+1)^2\sqrt{(m+1)(n+1)}}\sum_{i=1}^{N}\left\{\sin\left(\frac{\pi i}{N+1}\right)\sin\left((m+1)\frac{\pi i}{N+1}\right)\cdot\right.$$
$$\left.\cdot\sum_{j=1}^{N}\sin\left(\frac{\pi j}{N+1}\right)\sin\left((n+1)\frac{\pi j}{N+1}\right)\cdot M\left(\cos\left(\frac{\pi i}{N+1}\right),\cos\left(\frac{\pi j}{N+1}\right)\right)\right\},$$
$$B_m = \frac{-c(\omega_2-\omega_1)(G^*+2)}{(N+1)\sqrt{m+1}}\cdot \tag{2.61}$$
$$\cdot\sum_{i=1}^{N}\sin\left(\frac{\pi i}{N+1}\right)\sin\left((m+1)\frac{\pi i}{N+1}\right)\cdot H^*\left(\cos\left(\frac{\pi i}{N+1}\right)\right).$$

2.4 Знаходження коефіцієнтів інтенсивності напружень

В задачах з дефектом у вигляді тріщини особливу увагу приділяють коефіцієнту інтенсивності напружень в околі вершин тріщини, бо саму на цьому параметрі базуються деякі закони механіки руйнування, які дозволяють передбачити подальшу поведінку тріщини – чи буде вона розвиватися, чи залишиться стабільною.

У випадку цієї задачі КІН визначається як:

$$K_{\text{III}}^{+} = \lim_{\theta \to \omega_2 + 0} \left[\sqrt{2\pi(\theta - \omega_2)} \cdot \mathrm{T}_{rz}\big|_{r \to c+0} \right];$$
$$K_{\text{III}}^{-} = \lim_{\theta \to \omega_1 - 0} \left[\sqrt{2\pi(\omega_1 - \theta)} \cdot \mathrm{T}_{rz}\big|_{r \to c+0} \right]. \tag{2.62}$$

Оскільки $\mathrm{T}_{rz}\big|_{r \to c+0} = G_2 \dfrac{\partial W}{\partial r}\Big|_{r \to c+0}$, у раніше отриманому виразі для $\dfrac{\partial W}{\partial r}\Big|_{r \to c+0}$ (різниця лівої та правої частин (2.38)) відкинемо доданки, які мають скінчену границю при $\theta \to \omega_2 + 0$ або $\theta \to \omega_1 - 0$, бо відповідний множник під коренем ($\theta - \omega_2$ або $\omega_1 - \theta$), який обернеться в нуль, оберне в нуль і добуток з будь-яким доданком зі скінченою границею. Тоді, якщо врахувати всі зроблені перетворення, залишиться:

$$\frac{\partial W}{\partial r}\Big|_{r \to c+0} = \frac{2}{\pi c(\omega_2 - \omega_1)(G^* + 2)} \cdot \frac{d^2}{dt^2} \int_{-1}^{1} \varphi^*(\xi) \ln\left(\frac{1}{|t - \xi|} \right) d\xi. \tag{2.63}$$

Тоді для K_{III}^{+}:

$$K_{\text{III}}^{+} = \frac{2G_2}{\pi c(\omega_2 - \omega_1)(G^* + 2)} \cdot$$
$$\cdot \lim_{\substack{\theta \to \omega_2 + 0 \\ t \to 1+0}} \left[\sqrt{\pi(\omega_2 - \omega_1)(t-1)} \cdot \frac{d^2}{dt^2} \int_{-1}^{1} \varphi^*(\xi) \ln\left(\frac{1}{|t - \xi|} \right) d\xi \right]. \tag{2.64}$$

Підставивши сюди вираз для $\varphi^*(\xi)$ з (2.49), отримаємо:

$$K_{\text{III}}^{+} = \frac{2G_2}{c\left(G^* + 2\right)\sqrt{\pi\left(\omega_2 - \omega_1\right)}} \cdot$$
$$\cdot \sum_{n=0}^{\infty} \varphi_n^* \lim_{t \to 1+0} \left[\sqrt{t-1} \cdot \frac{d^2}{dt^2} \int_{-1}^{1} \sqrt{1-\xi^2} U_n(\xi) \ln\left(\frac{1}{|t-\xi|} \right) d\xi \right]. \tag{2.65}$$

Асимптотика останнього інтеграла наступна:

$$\frac{d^2}{dt^2} \int_{-1}^{1} \sqrt{1-\xi^2} U_n(\xi) \ln\left(\frac{1}{|t-\xi|} \right) d\xi \xrightarrow[t \to 1+0]{} \frac{\pi |t|}{\sqrt{t^2-1}} U_n(t). \tag{2.66}$$

Підставивши це в (2.65), отримаємо:

$$K_{\text{III}}^{+} = \frac{2G_2}{c\left(G^* + 2\right)\sqrt{\pi\left(\omega_2 - \omega_1\right)}} \cdot \sum_{n=0}^{\infty} \varphi_n^* \lim_{t \to 1+0} \left[\sqrt{t-1} \cdot \frac{\pi |t|}{\sqrt{t^2-1}} U_n(t) \right] =$$
$$= \frac{G_2 \sqrt{2\pi}}{c\left(G^* + 2\right)\sqrt{\omega_2 - \omega_1}} \sum_{n=0}^{\infty} \varphi_n^* \cdot U_n(1). \tag{2.67}$$

Враховуючи, що $U_n(1) = n+1$, $\varphi_n^* = \frac{X_n}{\sqrt{n+1}}$, нарешті отримаємо остаточну формулу для K_{III}^{+}:

$$K_{\text{III}}^{+} = \frac{G_2 \sqrt{2\pi}}{c\left(G^* + 2\right)\sqrt{\omega_2 - \omega_1}} \sum_{n=0}^{\infty} \sqrt{n+1} \cdot X_n. \tag{2.68}$$

Аналогічно для K_{III}^{-}:

$$K_{\text{III}}^{-} = \frac{G_2 \sqrt{2\pi}}{c\left(G^* + 2\right)\sqrt{\omega_2 - \omega_1}} \sum_{n=0}^{\infty} (-1)^n \sqrt{n+1} \cdot X_n. \tag{2.69}$$

РОЗДІЛ 3

Чисельні розрахунки та візуалізації

Перейдемо до розрахунків та візуалізацій напружень та переміщень. Всі вони реалізовані за допомогою програмного коду на мові Python 3.11, який приведено в Додатку А.

Почнемо з коефіцієнтів інтенсивності напружень. Зафіксуємо наступні константи: $a=5,\ \beta=\frac{\pi}{4}.$ Іншим початковим змінним спочатку виставимо наступні значення: $c=15,\ G_1=1,\ G_2=5,\ \omega_1=\frac{-\pi}{8},\ \omega_2=\frac{\pi}{8},\ f(\theta)=1,$ а далі будемо їм по черзі надавати інші значення і стежити за тим, як змінюватимуться коефіцієнти інтенсивності напружень. Результати представлені у таблиці 3.1.

Таблиця 3.1

Коефіцієнти інтенсивності напружень

Початкова змінна						КІН	
c	G_1	G_2	ω_1	ω_2	$f(\theta)$	K_{III}^{+}	K_{III}^{-}
15	1	5	$-\pi/8$	$\pi/8$	1	0.1298	0.1879
15	1	5	$-\pi/8$	$\pi/8$	2	0.2596	0.3757
15	1	1	$-\pi/8$	$\pi/8$	1	0.0825	0.1201
15	5	1	$-\pi/8$	$\pi/8$	1	0.0292	0.0429
8	1	5	$-\pi/8$	$\pi/8$	1	0.5189	0.6208
15	1	5	$-\pi/16$	$\pi/16$	1	0.1126	0.1333
15	1	5	$-\pi/8$	0	1	0.1331	0.1477

Як можемо помітити, коефіцієнти інтенсивності напружень поводяться передбачувано – зростають при наближені дефекту до криволінійної грані, де прикладено навантаження, або при збільшені цього навантаження, та зменшуються при зменшені тріщини. Також ці коефіцієнти доволі цікаво реагують на зміну співвідношення модулів зсуву двох матеріалів, на лінії сполучення яких розташований дефект.

Подивимось тепер на те, як в тілі поширюються переміщення та напруження при різних значеннях початкових змінних. Оберемо такий самий «стандартний» набір початкових значень: $a=5,\ \beta=\frac{\pi}{4},\ c=15,\ G_1=1,\ G_2=5,$ $\omega_1=\frac{-\pi}{8},\ \omega_2=\frac{\pi}{8},\ f(\theta)=1.$ Співвідношення модулів зсуву 1 до 5 може відповідати, наприклад, магнію та сталі.

Подивимось спочатку на переміщення точок тіла за таких умов (рис. 3.1). Можемо побачити, як виконуються умови задачі (1.2) – (1.4): переміщення дорівнюють нулю на закріпленій грані, прямують до нуля зі збільшенням радіусу та не мають стрибка на $r=c$ всюди, за виключенням тріщини, яку і можна побачити завдяки стрибку переміщень.

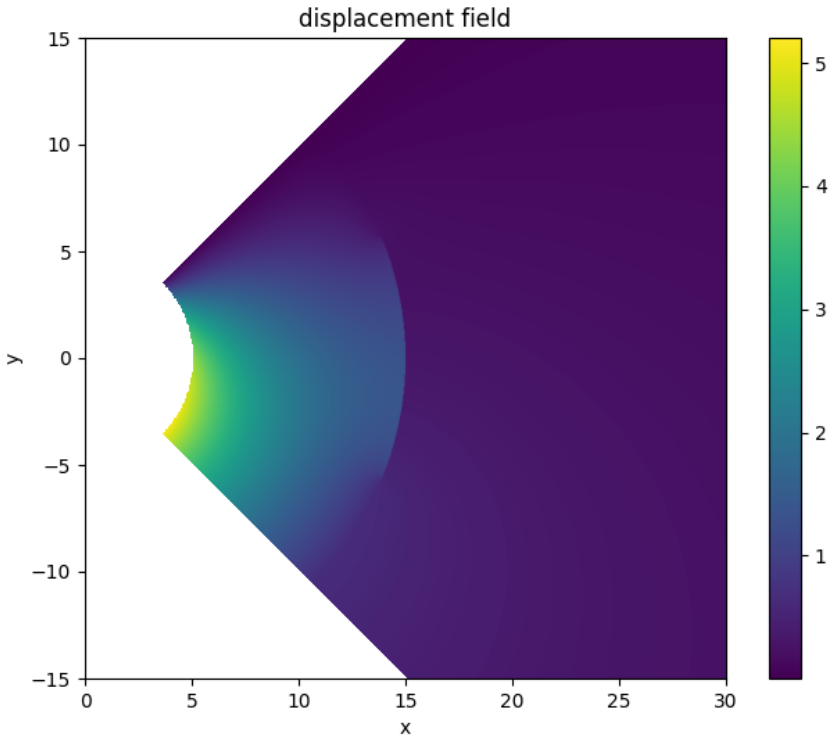

Рис. 3.1. Переміщення точок тіла при «стандартному» наборі констант

Подивимось тепер на радіальні дотичні напруження за тих самих умов (рис. 3.2). Тут також можемо побачити виконання умов задачі: напруження великі за модулем на криволінійній грані, неперервні при переході через лінію сполучення матеріалів $(r = c)$ та дорівнюють нулю на тріщині. Також радіальні напруження відносно великі за модулем на вершинах дефекту, що й очікувалось.

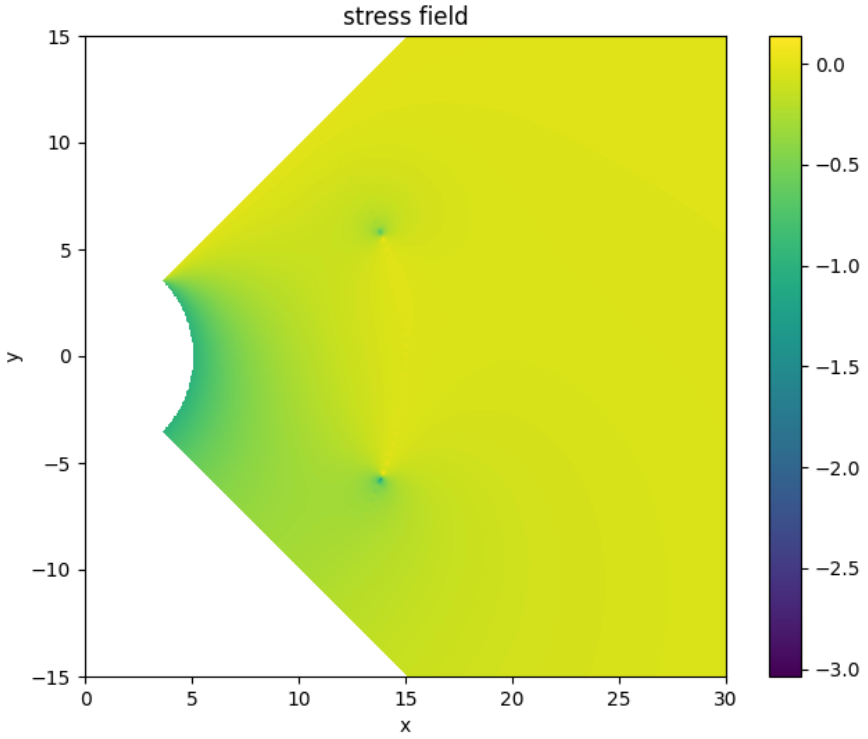

Рис. 3.2. Радіальні напруження точок тіла при «стандартному» наборі констант

Змінимо тепер модуль зсуву G_2 з 5 до 1. Це відповідає випадку, у якому обидва матеріали, з яких зроблено тіло, мають однаковий модуль зсуву. Помітимо, що на рис. 3.3 переміщення краще розповсюджуються на другий матеріал, особливо між дефектом та вільною гранню.

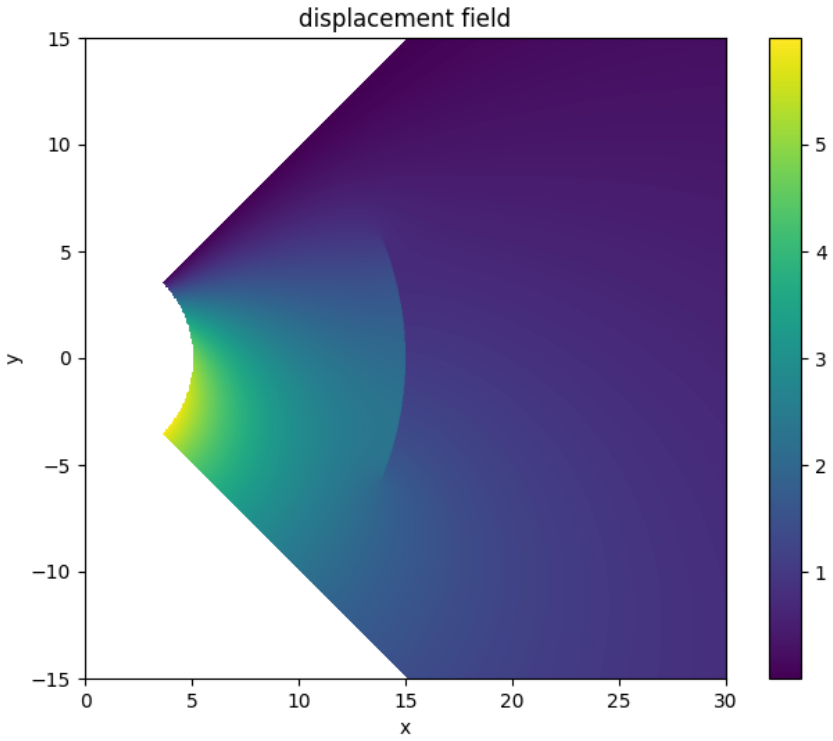

Рис. 3.3. Переміщення точок тіла при однакових модулях зсуву

Повернемось до «стандартного» набору значень та покладемо $c = 8$. Ця зміна перенесе лінію сполучення матеріалів разом з дефектом ближче до грані, до якої прикладено навантаження. На рис. 3.4 можна побачити те, як це змінює розповсюдження переміщень на другий матеріал, навіть попри те, що співвідношення модулів зсуву знову дорівнює один до п’яти.

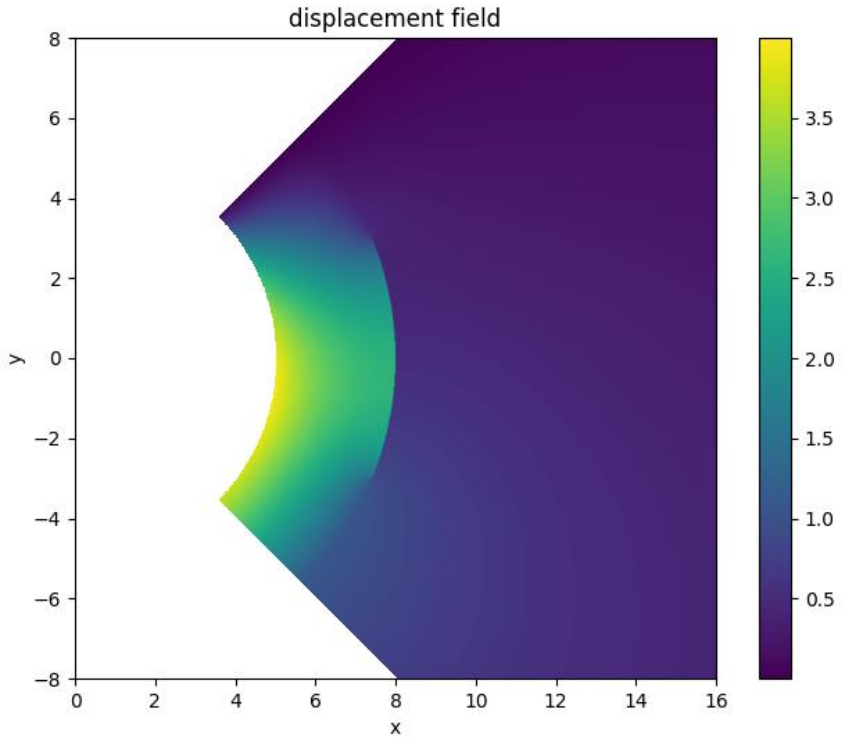

Рис. 3.4. Переміщення точок тіла при $c = 8$

Зрозуміло, існує ще безліч цікавих комбінацій параметрів. Для можливості швидко та зручно побудувати візуалізацію переміщень та тангенціальних напружень для будь-якої комбінації було створено код-обгортку, який приведено в Додатку Б. Інтерфейс, який він пропонує, приведено на рис. 3.5.

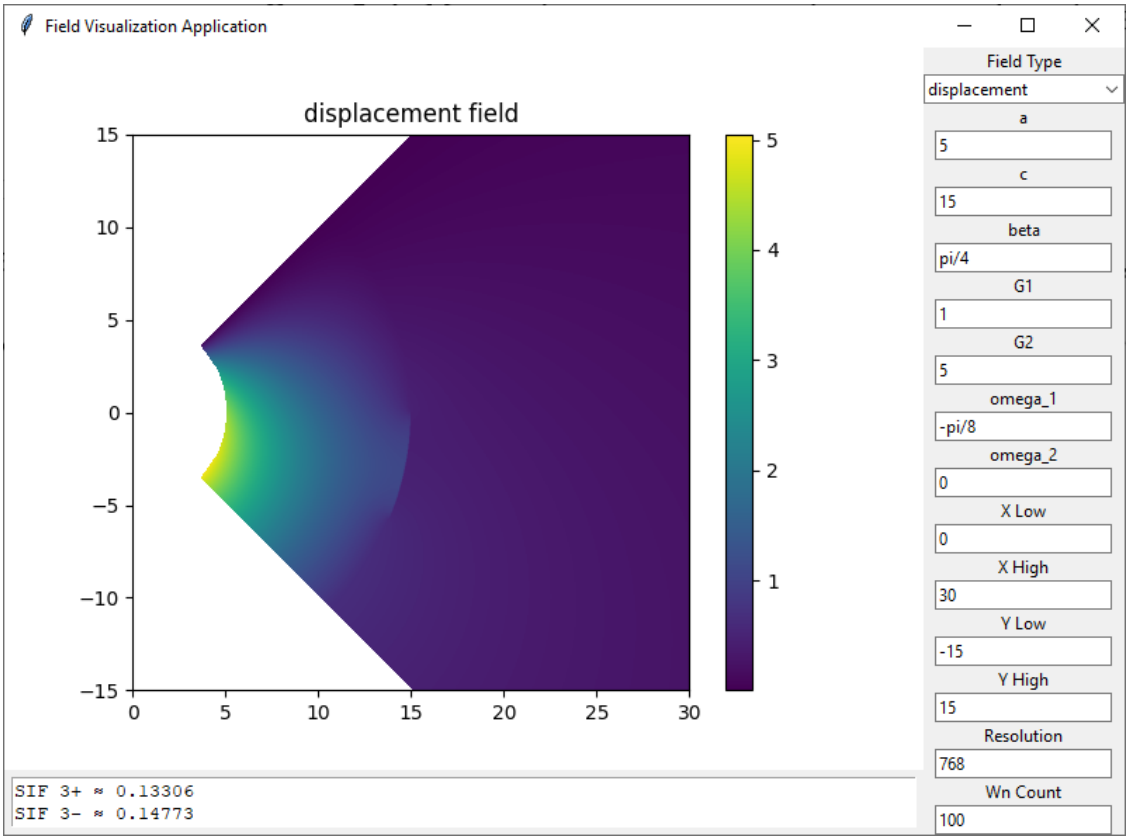

Рис. 3.5. Інтерфейс програми з додатку Б

ВИСНОВКИ

У межах цієї роботи була поставлена та успішно вирішена антиплоска задача теорії пружності для зовнішнього складеного сектору кола з міжфазним дефектом. До сформованої крайової задачі був застосований метод інтегральних перетворень, який дозволив звести її до одновимірної. За допомогою отриманого розв’язку одновимірної задачі було виведено рівняння для знаходження невідомої функції переміщення одного берега тріщини відносно іншого, яке було зведено до нескінченої системи рівнянь за допомогою методу ортогональних многочленів, а для розв’язання системи застосовано метод редукції.

Для аналітичних розв’язків було розроблено програмний код на мові Python 3.11, який забезпечив візуалізацію отриманих функцій переміщень та тангенціальних дотичних напружень, а також розрахунок коефіцієнтів інтенсивності напружень в околі вершин дефекту. Для зручних експериментів із початковими параметрами задачі було створено простий графічний інтерфейс.

Побудовані візуалізації та виконані числові розрахунки підтвердили адекватність отриманого розв’язку та допомогли зрозуміти характер розподілу переміщень та напружень у розглянутому тілі. Отримані результати можуть бути корисними при аналізі інженерних конструкцій, що складаються з різнорідних матеріалів та можуть мати міжфазний дефект. Таким чином, мету роботи повністю досягнуто.

СПИСОК ЛІТЕРАТУРИ

1. Метасов, А. С. Антиплоска задача теорії пружності для зовнішнього складеного сектору кола : кваліфікаційна робота бакалавра ; Antiplane problem of elasticity theory for the external compound sector of a circle / А. С. Метасов. – Одеса, 2023. – 27 с.
2. Процеров Ю. С. Методичні вказівки до курсу «Математичне моделювання деяких задач механіки і техніки» для студентів 3 курсу спеціальності «Прикладна математика» / Ю. С. Процеров, О. П. Мойсєєнок. – Одеса : Одеськ. нац. ун-т ім. І. І. Мечникова, 2015. – 61 с.
3. Попов Г. Я. Навчальний посібник з курсу "Рівняння математичної фізики. Метод інтегральних перетворень" : для студ. техн. спец. вузів / Г. Я. Попов, В. В. Реут, Н. Д. Вайсфельд ; відп. ред.: В. Є. Круглов ; ОНУ ім. І.І. Мечникова. – Одеса : Астропринт, 2005. – 183 с.
4. Рівняння математичної фізики. Метод ортогональних многочленів : навч. посіб. для студ. вузів, які навч. за напрямами підготовки "Прикладна математика", "Механіка" / Г. Я. Попов [та ін.]. – Одеса : Астропринт, 2010. – 115 с.

ДОДАТКИ

Додаток А

Програмний код на Python 3.11 для розрахунків та візуалізацій розв'язків

```
from typing import Literal, Tuple

import numpy as np  # can be changed to "cupy as np" for GPU acceleration
from numpy import pi
import matplotlib.pyplot as plt

# Default constants
a = 5
c = 15
beta = np.pi / 4
G1 = 1
G2 = 5
omega_1 = -np.pi / 8
omega_2 = np.pi / 8
f_multiplier = 1

G_star = (G2 - G1) / G1


def alpha_n(n: np.ndarray) -> np.ndarray:
    # computes alpha_n for array of n values
    return pi / (4 * beta) * (2 * n - 1)


def f_n(n: np.ndarray) -> np.ndarray:
    # computes f_n for array of n values, assuming f(theta) = 1 * f_multiplier
    alphas = alpha_n(n)
    return np.sin(2 * alphas * beta) / alphas * f_multiplier


def delta_n(n: np.ndarray) -> np.ndarray:
    # computes delta_n for array of n values
    return 1 + G_star / 2 * (1 + np.power(a / c, 2 * alpha_n(n)))


def H_star(t_values: np.ndarray, n_max: int) -> np.ndarray:
    # computes H_star for t values, used in B_m coefficients through H(theta)
    thetas = (omega_2 - omega_1) * t_values / 2 + (omega_2 + omega_1) / 2

    # compute the sum for n = 1,2,...,n_max
    n_indices = np.arange(1, n_max + 1)
    alphas = alpha_n(n_indices)
    deltas = delta_n(n_indices)

    coeffs_n = f_n(n_indices) * np.power(a / c, alphas) * (1 - G_star / (2 *
deltas) * (1 + np.power(a / c, 2 * alphas)))
    cosine_2d = np.cos(np.outer(alphas, thetas + beta))  # cosine matrix of
shape (n_max, t_values.size)
    H_sum = np.sum(coeffs_n[:, np.newaxis] * cosine_2d, axis=0)  # result of
shape (t_values.size,)
    return -a / (beta * G1 * c) * H_sum


def B_coeffs(vector_length: int, N_max: int, H_max_n: int) -> np.ndarray:
    # computes B_m coefficients for m = 1,2,...,vector_length
```

```
    i_indices = np.arange(1, N_max + 1)
    m_indices = np.arange(0, vector_length)

    coeffs_i = np.sin(pi * i_indices / (N_max + 1)) * H_star(np.cos(pi *
i_indices / (N_max + 1)), H_max_n)
    sin_2d = np.sin(np.outer((m_indices + 1), pi * i_indices / (N_max + 1)))  #
shape (vector_length, N_max)
    B_sum = np.sum(coeffs_i * sin_2d, axis=1)  # result of shape
(vector_length,)
    return (-c * (omega_2 - omega_1) * (G_star + 2) / (np.sqrt(m_indices + 1) *
(N_max + 1))) * B_sum


def K(theta_values: np.ndarray, eta_values: np.ndarray, max_n: int) ->
np.ndarray:
    # computes K for theta and eta values arrays of the same size
    n_indices = np.arange(1, max_n + 1)
    alphas = alpha_n(n_indices)
    deltas = delta_n(n_indices)

    coeffs_n = np.power(a / c, 2 * alphas) * (1 - G_star / (2 * deltas) *
np.power(a / c, 2 * alphas)) * alphas
    cosines_2d = np.cos(np.outer(alphas, theta_values + beta)) *
np.cos(np.outer(alphas, eta_values + beta))  # shape (max_n, theta_values.size)
    K_sum = np.sum(coeffs_n[:, np.newaxis] * cosines_2d, axis=0)  # result of
shape (theta_values.size,)
    return K_sum


def R_star(t_values: np.ndarray, xi_values: np.ndarray, max_n: int, K_max_n:
int) -> np.ndarray:
    # computes R_star for t and xi values arrays of the same size through
R(theta, eta)
    theta_values = (omega_2 - omega_1) * t_values / 2 + (omega_2 + omega_1) / 2
    eta_values = (omega_2 - omega_1) * xi_values / 2 + (omega_2 + omega_1) / 2
    n_indices = np.arange(1, max_n + 1)

    ln_deriv_1 = -pi ** 2 / (16 * beta ** 2 * (np.cos(pi * (theta_values -
eta_values) / (4 * beta)) + 1))
    ln_deriv_2 = -pi ** 2 / (16 * beta ** 2 * (np.cos(pi * (2 * beta +
eta_values + theta_values) / (4 * beta)) + 1))
    ln_deriv_3 = -pi ** 2 / (16 * beta ** 2 * (np.cos(pi * (2 * beta +
eta_values + theta_values) / (4 * beta)) - 1))

    K_values = K(theta_values, eta_values, K_max_n)

    # compute the sum for n = 1,2,...,max_n
    alphas = alpha_n(n_indices)
    deltas = delta_n(n_indices)

    coeffs_n = alphas / deltas * np.power(a / c, 2 * alphas)
    cosines_2d = np.cos(np.outer(alphas, theta_values + beta)) *
np.cos(np.outer(alphas, eta_values + beta))  # shape (max_n, theta_values.size)
    R_sum = np.sum(coeffs_n[:, np.newaxis] * cosines_2d, axis=0)  # result of
shape (theta_values.size,)

    return ln_deriv_1 + ln_deriv_2 - ln_deriv_3 - G_star ** 2 * pi / (2 * beta)
* R_sum + pi * (G_star + 2) / beta * K_values


def M(t_values: np.ndarray, xi_values: np.ndarray, R_max_n: int, K_max_n: int) -
> np.ndarray:
    # computes M for t and xi values arrays of the same size
```

```
    ln_deriv = 2 / np.power(t_values - xi_values, 2) - (pi ** 2 * (omega_1 -
omega_2) ** 2) / (128 * beta ** 2 * np.power(np.sin((pi * (omega_2 - omega_1) *
(t_values - xi_values)) / (16 * beta)),2))
    R_star_values = R_star(t_values, xi_values, R_max_n, K_max_n)

    return -1 / 2 * ln_deriv + 1 / 2 * ((omega_2 - omega_1) / 2) ** 2 *
R_star_values


def A_coeffs(matrix_size: int, max_N: int, R_max_n: int, K_max_n: int) ->
np.ndarray:
    # computes square (M x M) matrix A_mn for m = 1,2,...,matrix_size
    i_indices = np.arange(1, max_N + 1)
    j_indices = np.arange(1, max_N + 1)
    mn_indices = np.arange(0, matrix_size)  # matrix is square so we can use the
same indices for rows and columns

    # precompute sines and cosines for i and j values (separate variables for
readability, even though i/j are the same)
    sin_i = np.sin(pi * i_indices / (max_N + 1))
    sin_j = np.sin(pi * j_indices / (max_N + 1))
    cos_i = np.cos(pi * i_indices / (max_N + 1))
    cos_j = np.cos(pi * j_indices / (max_N + 1))

    # precompute M for all pairs of cosines
    t_values = cos_i[:, np.newaxis]  # shape (N_max, 1)
    xi_values = cos_j[np.newaxis, :]  # shape (1, N_max)
    M_matrix = M(t_values, xi_values, R_max_n, K_max_n)  # shape (N_max, N_max)
    M_matrix[np.isnan(M_matrix)] = 0

    # Compute the inner sums over j for each n
    inner_sum_coeff_j = sin_j[:, np.newaxis] * np.sin(np.outer(j_indices,
(mn_indices + 1) * pi / (max_N + 1)))  # shape (N_max, matrix_size)
    weighted_M = M_matrix[:, :, np.newaxis] * inner_sum_coeff_j[:, np.newaxis,:]
# broadcasting to shape (N_max, N_max, matrix_size)
    inner_sums = np.sum(weighted_M, axis=1)  # summation over j, shape (N_max,
matrix_size)

    # Compute the outer sum over i for each m
    outer_sum_coeff = sin_i[:, np.newaxis] * np.sin(np.outer(i_indices,
(mn_indices + 1) * pi / (max_N + 1)))  # shape (N_max, matrix_size)
    outer_sum_coeff_reshaped = outer_sum_coeff[:, :, np.newaxis]  # Shape
(N_max, matrix_size, 1)

    # Multiply element-wise and sum over the first axis (the N_max dimension)
    A_matrix_sums = np.sum(outer_sum_coeff_reshaped * inner_sums[:, np.newaxis,
:], axis=0)  # shape (matrix_size, matrix_size)

    factor = -2 / (np.sqrt(np.outer(mn_indices + 1, mn_indices + 1)) * (max_N +
1) ** 2)  # shape (matrix_size, matrix_size)
    return factor * A_matrix_sums


def solve_infinite_system(n_roots: int, A_max_n: int, B_max_n: int, R_max_n:
int, K_max_n: int, H_max_n: int) -> np.ndarray:
    # solve the system (A + I) * X = B
    A_matrix = A_coeffs(n_roots, A_max_n, R_max_n, K_max_n)
    B_vector = B_coeffs(n_roots, B_max_n, H_max_n)

    return np.linalg.solve(A_matrix + np.eye(n_roots), B_vector)
```

```
def SIF_3(max_n: int, A_max_n: int, B_max_n: int, series_max_n: int, plus: bool)
-> np.ndarray:
    # computes the stress intensity factor SIF_3 + or - for a given number of
terms
    n_indices = np.arange(0, max_n)

    roots = solve_infinite_system(max_n, A_max_n, B_max_n, series_max_n,
series_max_n, series_max_n)
    SIF_sum_terms = np.sqrt(n_indices + 1) * roots

    if not plus:
        # For SIF_3- additionally multiply each term by (-1)^n
        SIF_sum_terms *= np.power(-1, n_indices)

    return np.sqrt(2 * pi) * G2 / (c * (G_star + 2) * np.sqrt(omega_2 -
omega_1)) * np.sum(SIF_sum_terms)


#  Functions that are needed for the displacement field W(r, theta) and the
stress field

def chebyshev_U(max_n: int, x: np.ndarray) -> np.ndarray:
    # computes Chebyshev polynomials of the second kind up to degree max_n of
array x

    U = np.zeros((max_n, x.size))
    U[0, :] = 1
    U[1, :] = 2 * x

    for n in range(2, max_n):
        U[n, :] = 2 * x * U[n - 1, :] - U[n - 2, :]

    return U


def fi(eta: np.ndarray, max_n: int, A_max_n: int, B_max_n, series_max_n: int) ->
np.ndarray:
    # computes fi(eta) for array of eta values through fi_star(xi)
    xi_values = (omega_2 + omega_1 - 2 * eta) / (omega_1 - omega_2)
    n_indices = np.arange(0, max_n)

    X_n = solve_infinite_system(max_n, A_max_n, B_max_n, series_max_n,
series_max_n, series_max_n)
    U_n = chebyshev_U(max_n, xi_values)

    return np.sqrt(1 - xi_values ** 2) * np.sum((X_n / np.sqrt(n_indices +
1))[:, np.newaxis] * U_n, axis=0)


def fi_n(n: np.ndarray, n_points: int, fi_max_n: int, A_max_n: int, B_max_n:
int, series_max_n: int) -> np.ndarray:
    # computes fi_n as integral of fi(eta) * cos(alpha_n * (eta + beta)) deta
from omega_1 to omega_2 for all n
    eta_values = np.linspace(omega_1, omega_2, n_points)
    fi_values = fi(eta_values, fi_max_n, A_max_n, B_max_n, series_max_n)
    alphas = alpha_n(n)

    return np.trapz(fi_values * np.cos(np.outer(alphas, eta_values + beta)),
eta_values, axis=1)


def W(r: np.ndarray,
      theta: np.ndarray,
```

```
      Wn_count: int,
      fi_n_integration_points: int,
      fi_n_terms: int,
      A_n_terms: int,
      B_n_terms: int,
      general_series_n_terms: int) -> np.ndarray:

    # computes the displacement field W(r, theta) for given r and theta matrices
of the same shape
    n_indices = np.arange(1, Wn_count + 1)
    alphas = alpha_n(n_indices)
    deltas = delta_n(n_indices)
    f_n_values = f_n(n_indices)

    # For each Wn we need corresponding fi_n
    fi_n_values = fi_n(n_indices, fi_n_integration_points, fi_n_terms,
A_n_terms, B_n_terms, general_series_n_terms)

    # We also need A1_n = G_star * dWn/dr (c+0)
    Wn_prime_c = -a / (G1 * c * deltas) * f_n_values * np.power(a / c, alphas) +
alphas * fi_n_values / (2 * c * deltas) * (1 - np.power(a / c, 2 * alphas))
    A1_n = G_star * Wn_prime_c

    # prepare r and theta for the broadcasting
    r = r[..., np.newaxis]  # shape (r.shape, 1)
    theta = theta[..., np.newaxis]  # shape (theta.shape, 1)

    # finally compute the displacement field
    ln_r_c = np.log(r / c)
    first_summand = a / (alphas * G1) * f_n_values * np.power(a / r, alphas)
    second_summand = -fi_n_values / 2 * (np.exp(-alphas * np.abs(ln_r_c)) *
np.sign(ln_r_c) - np.power(a / r, alphas) * np.exp(alphas * np.log(a / c)))
    third_summand = A1_n * c / (2 * alphas) * (np.exp(-alphas * np.abs(ln_r_c))
+ np.power(a / r, alphas) * np.exp(alphas * np.log(a / c)))
    Wn = first_summand + second_summand + third_summand  # shape (r.shape,
Wn_count)

    # compute W over all combinations of r and theta
    cosines = np.cos(alphas[np.newaxis, np.newaxis, :] * (theta + beta))  #
shape (theta.shape, Wn_count)

    W_sums = np.sum(Wn * cosines, axis=2)  # shaped as r.shape or theta.shape
    return 1 / beta * W_sums


def Tau_r_z(r: np.ndarray,
            theta: np.ndarray,
            G: np.ndarray,
            Wn_count: int,
            fi_n_integration_points: int,
            fi_n_terms: int,
            A_n_terms: int,
            B_n_terms: int,
            general_series_n_terms: int) -> np.ndarray:

    # computes the shear stress Tau_r_z for a given r and theta of the same
shape
    n_indices = np.arange(1, Wn_count + 1)
    alphas = alpha_n(n_indices)
    deltas = delta_n(n_indices)
    f_n_values = f_n(n_indices)

    # For each dWn/dr we need corresponding fi_n
```

```
    fi_n_values = fi_n(n_indices, fi_n_integration_points, fi_n_terms,
A_n_terms, B_n_terms, general_series_n_terms)

    # We also need A1_n = G_star * dWn/dr (c+0)
    Wn_prime_c = -a / (G1 * c * deltas) * f_n_values * np.power(a / c, alphas) +
alphas * fi_n_values / (2 * c * deltas) * (1 - np.power(a / c, 2 * alphas))
    A1_n = G_star * Wn_prime_c

    # prepare r and theta for the broadcasting
    r = r[..., np.newaxis]  # shape (r.shape, 1)
    theta = theta[..., np.newaxis]  # shape (theta.shape, 1)

    # compute dWn/dr from three summands
    ln_r_c = np.log(r / c)
    first_summand = -a / (G1 * r) * f_n_values * np.power(a / r, alphas)
    second_summand = alphas * fi_n_values / (2 * r) * (np.exp(-alphas *
np.abs(ln_r_c)) - np.power(a / c, alphas) * np.power(a / r, alphas))
    third_summand = -A1_n * c / (2 * r) * (np.exp(-alphas * np.abs(ln_r_c)) *
np.sign(ln_r_c) + np.power(a / c, alphas) * np.power(a / r, alphas))
    Wn_prime = first_summand + second_summand + third_summand  # shape (r.shape,
Wn_count)

    # compute Tau_r_z over all combinations of r and theta
    cosines = np.cos(alphas[np.newaxis, np.newaxis, :] * (theta + beta))  #
shape (theta.shape, Wn_count)

    Tau_r_z_sums = np.sum(Wn_prime * cosines, axis=2)  # shaped as r.shape or
theta.shape
    return G / beta * Tau_r_z_sums


def get_field(field: Literal["displacement", "stress"], x_low: float, x_high:
float, y_low: float, y_high: float,
              resolution: int, Wn_count: int, fi_n_integration_points: int,
fi_n_terms: int, A_n_terms: int,
              B_n_terms: int, general_series_n_terms: int, visualize: bool =
True, return_SIFs: bool = False,
              custom_constants: dict = None) -> np.ndarray | Tuple[np.ndarray,
np.ndarray, np.ndarray]:
    """
    Computes and optionally draws the requested field for the given range,
accuracy and resolution
    :param field: type of field to compute, either "displacement" or "stress"
    :param x_low: lower bound of x coordinate
    :param x_high: higher bound of x coordinate
    :param y_low: lower bound of y coordinate
    :param y_high: higher bound of y coordinate
    :param resolution: number of points in each dimension (x and y)
    :param Wn_count: number of terms in the Wn series
    :param fi_n_integration_points: number of points for the fi_n integration
    :param fi_n_terms: number of terms in the fi_n series (under the integral)
    :param A_n_terms: number of terms in the A_n series
    :param B_n_terms: number of terms in the B_n series
    :param general_series_n_terms: number of terms in all other series
    :param visualize: whether to draw the field
    :param return_SIFs: whether to compute and return SIF_3+ and SIF_3- values
    :param custom_constants: dictionary of custom global constants to remotely
change the default ones
    :return: computed field values or field values and SIF_3+ and SIF_3- values
    """

    # set custom constants using globals() if provided
    if custom_constants is not None:
```

```
        for key, value in custom_constants.items():
            globals()[key] = value

        # change global G_star in case G1 or G2 is changed
        globals()['G_star'] = (G2 - G1) / G1

    # prepare the coordinate grid
    x_points = np.linspace(x_low, x_high, resolution)
    y_points = np.linspace(y_low, y_high, resolution)

    # prepare the meshgrid for the W function in polar coordinates
    X, Y = np.meshgrid(x_points, y_points)
    R = np.sqrt(X ** 2 + Y ** 2)
    Theta = np.arctan2(Y, X)

    field_values = None

    if field == "displacement":
        field_values = W(R, Theta, Wn_count, fi_n_integration_points,
fi_n_terms, A_n_terms, B_n_terms, general_series_n_terms)
    elif field == "stress":
        # G value for each point is either G1 or G2 depending on the material
        G_values = np.where(R < c, G1, G2)
        field_values = Tau_r_z(R, Theta, G_values, Wn_count,
fi_n_integration_points, fi_n_terms, A_n_terms, B_n_terms,
general_series_n_terms)

    # set all values outside the body (r < a or theta < -beta or theta > beta)
to NaN
    field_values[R < a] = np.nan
    field_values[Theta < -beta] = np.nan
    field_values[Theta > beta] = np.nan

    # convert field_values to numpy array if 'cupy' is imported as np (optional
for compatibility)
    if np.__name__ == 'cupy':
        field_values = np.asnumpy(field_values)

    if visualize:
        fig = plt.figure(figsize=(8, 6))
        ax = fig.add_subplot(1, 1, 1, xlabel='x', ylabel='y')
        ax.set_title(f'{field} field')

        im = ax.imshow(field_values, extent=(x_low, x_high, y_low, y_high),
origin='lower', cmap='viridis')

        fig.colorbar(im)
        plt.show()

    if return_SIFs:
        SIF_plus = SIF_3(Wn_count, A_n_terms, B_n_terms, general_series_n_terms,
plus=True)
        SIF_minus = SIF_3(Wn_count, A_n_terms, B_n_terms,
general_series_n_terms, plus=False)
        return field_values, np.round(SIF_plus, 5), np.round(SIF_minus, 5)
    else:
        return field_values



if __name__ == '__main__':
    # Example usage
```

```
    disp_field, SIF_3_Plus, SIF_3_Minus = get_field("displacement", 0, 2 * c, -
c, c, 768, 100, 200, 300, 500, 500, 500, return_SIFs=True)

    stress_field = get_field("stress", 0, 2 * c, -c, c, 768, 100, 200, 300, 500,
500, 500)

    print('SIF_3+: ', SIF_3_Plus)
    print('SIF_3-: ', SIF_3_Minus)
```

Додаток Б

Код-обгортка над Додатком А на Python 3.11 для додавання інтерфейсу

```
import math
import tkinter as tk
from tkinter import ttk
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg

from main import get_field

# Main application window
root = tk.Tk()
root.title("Field Visualization Application")

# Frame for the plot and text field
plot_frame = tk.Frame(root)
plot_frame.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)

# Canvas for the plot
fig, ax = plt.subplots(figsize=(5, 5))
canvas = FigureCanvasTkAgg(fig, master=plot_frame)
canvas.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=True)

# Text field below the plot on the left to display SIF values
text = tk.Text(plot_frame, height=2, wrap=tk.WORD)
text.pack(side=tk.BOTTOM, fill=tk.X, padx=5, pady=5)

# Control Panel on the right
control_frame = tk.Frame(root)
control_frame.pack(side=tk.RIGHT, fill=tk.Y)

# Field type dropdown
field_var = tk.StringVar(value="displacement")
ttk.Label(control_frame, text="Field Type").pack()
field_dropdown = ttk.Combobox(control_frame, textvariable=field_var,
values=["displacement", "stress"])
field_dropdown.pack()
field_dropdown.bind("<<ComboboxSelected>>", lambda e: update_plot())  # Update
plot when dropdown changes

# Input fields for each parameter with default values
x_low_var = tk.StringVar(value="0")
x_high_var = tk.StringVar(value="30")
y_low_var = tk.StringVar(value="-15")
y_high_var = tk.StringVar(value="15")
resolution_var = tk.StringVar(value="768")
wn_count_var = tk.StringVar(value="100")

a_var = tk.StringVar(value="5")
c_var = tk.StringVar(value="15")
beta_var = tk.StringVar(value="pi/4")
G1_var = tk.StringVar(value="1")
G2_var = tk.StringVar(value="5")
omega_1_var = tk.StringVar(value="-pi/8")
omega_2_var = tk.StringVar(value="pi/8")


def update_plot():
    global colorbar  # Track the colorbar instance
```

```
    # Update the title while computing
    root.title("Field Visualization Application - Computing...")
    root.update()

    field_type = field_var.get()
    x_low = float(x_low_var.get())
    x_high = float(x_high_var.get())
    y_low = float(y_low_var.get())
    y_high = float(y_high_var.get())
    resolution = int(resolution_var.get())
    Wn_count = int(wn_count_var.get())

    a = float(a_var.get())
    c = float(c_var.get())
    beta = eval(beta_var.get().replace("pi", str(math.pi)))
    G1 = float(G1_var.get())
    G2 = float(G2_var.get())
    omega_1 = eval(omega_1_var.get().replace("pi", str(math.pi)))
    omega_2 = eval(omega_2_var.get().replace("pi", str(math.pi)))

    custom_globals = {  # User-defined constants to pass to the main code
        'a': a,
        'c': c,
        'beta': beta,
        'G1': G1,
        'G2': G2,
        'omega_1': omega_1,
        'omega_2': omega_2
    }

    # Compute the field
    field_values, SIF_3_Plus, SIF_3_Minus = get_field(field_type, x_low, x_high,
y_low, y_high, resolution, Wn_count,
                                                      100, 100, 500, 500, 1000,
False, True, custom_globals)

    # Update the text field with SIF values
    text.delete(1.0, tk.END)
    text.insert(tk.END, f"SIF 3+ ≈ {SIF_3_Plus}\n")
    text.insert(tk.END, f"SIF 3- ≈ {SIF_3_Minus}")

    # Clear the plot and remove the old colorbar if it exists
    if 'colorbar' in globals() and colorbar is not None:
        colorbar.remove()

    ax.clear()

    ax.set_title(f'{field_type} field')
    im = ax.imshow(field_values, extent=(x_low, x_high, y_low, y_high),
origin='lower', cmap='viridis')
    colorbar = fig.colorbar(im, ax=ax)
    canvas.draw()

    root.title("Field Visualization Application")


# Helper function to create and pack a labeled entry
def add_labeled_entry(label_text, variable):
    ttk.Label(control_frame, text=label_text).pack()
    entry = ttk.Entry(control_frame, textvariable=variable)
    entry.pack()
    entry.bind("<Return>", lambda e: update_plot())
```

```
# Add labeled entries
add_labeled_entry("a", a_var)
add_labeled_entry("c", c_var)
add_labeled_entry("beta", beta_var)
add_labeled_entry("G1", G1_var)
add_labeled_entry("G2", G2_var)
add_labeled_entry("omega_1", omega_1_var)
add_labeled_entry("omega_2", omega_2_var)

add_labeled_entry("X Low", x_low_var)
add_labeled_entry("X High", x_high_var)
add_labeled_entry("Y Low", y_low_var)
add_labeled_entry("Y High", y_high_var)
add_labeled_entry("Resolution", resolution_var)
add_labeled_entry("Wn Count", wn_count_var)

# Initial plot
update_plot()

# Run the Tkinter event loop
root.mainloop()
```