**МІНІСТЕРСТВО ОСВІТИ І НАУКИ,
МОЛОДІ ТА СПОРТУ УКРАЇНИ
Запорізький національний технічний університет**

МЕТОДИЧНІ ВКАЗІВКИ

до лабораторних робіт з дисципліни “Оптимізація проектних рішень обладнання енергоємних виробництв”
для студентів спеціальності 8.05070207
“Електромеханічне обладнання енергоємних виробництв”
денної форми навчання

2013

Методичні вказівки до лабораторних робіт з дисципліни “Оптимізація проектних рішень обладнання енергоємних виробництв” для студентів спеціальності 8.05070207 “Електромеханічне обладнання енергоємних виробництв” денної форми навчання / Укл.: М. І. Коцур. – Запоріжжя: ЗНТУ, 2013. 42 с.

Укладач: М. І. Коцур, ст. викладач, канд. техн. наук

Рецензент: О. В. Близняков, доцент, канд. техн. наук

Відповідальний
за випуск: П. Д. Андрієнко, професор, д-р. техн. наук

Затверджено
на засіданні кафедри
“Електричні та електронні апарати”

Протокол № 6
від “ 16 “ січня 2013

ЗМІСТ

ВСТУП

При експериментальних дослідженнях часто зустрічається завдання представлення якоїсь залежності, заданої окремими точками, у вигляді гладкої функції. Завдання оптимізації - завдання, що дозволяють вибрати на множині можливих напрямків ті з них, які забезпечують найбільш ефективне (з погляду певного критерію) досягнення до поставленої мети.

У розв'язку будь-якої практичної оптимізаційного завдання існує кілька етапів.

На першому етапі визначають границі досліджуваної системи, що дозволяє сформулювати деяке завдання виду $f(x) \rightarrow min$, яке необхідно розв'язати.

Наступним етапом є вибір математичного методу, який би забезпечував одержання кінцевих результатів з найменшими витратами на обчислення або ж давав можливість одержати найбільший обсяг інформації про шуканий розв'язок. Вибір того або іншого методу в значній мірі визначається постановкою оптимального завдання, а також використанням математичних моделей об'єкта оптимізації.

У цей час розроблене безліч чисельних методів для завдань як безумовної, так і умовної оптимізації. Природнім є прагнення вибрати для розв'язку конкретного завдання найкращий метод, що дозволяє за найменший час використання ПК одержати розв'язок із заданою точністю.

Якість чисельного методу характеризується багатьма факторами: швидкістю збіжності, часом виконання однієї ітерації, обсягом пам'яті ПК, необхідним для реалізації методу, класом розв'язуваних завдань і т.д. Розв'язувані завдання також досить різноманітні: вони можуть мати високу й малу розмірність, бути унімодальними і багато екстремальними і т.д. Той самий метод, ефективний для розв'язку завдань одного типу, може виявитися зовсім неприйнятним для завдань іншого типу. Тому для розв'язку поставлених завдань дуже корисно знати основні властивості, специфіку методів оптимізації. Це забезпечує здатність правильно орієнтуватися в різних ситуаціях розрахунків, що виникають у їх процесі, і щонайкраще розв'язати завдання.

ЛАБОРАТОРНА РОБОТА 1

ВИВЧЕННЯ МЕТОДІВ МІНІМІЗАЦІЇ ФУНКЦІЇ ОДНІЄЇ ЗМІННОЇ

Мета роботи: Вивчення та надбання навичок пошуку оптимальних значень функції однієї змінної.

1.1 Короткі теоретичні відомості

1.1.1 Знаходження екстремальних значень функції *f(x)*

Функція *f(x)* має ***локальний мінімум*** у точці $x = p$, якщо існує такий відкритий інтервал *I*, що містить *p*, при умові $f(p) \le f(x)$ для всіх $x \in I$.

Функція *f(x)* має ***локальний максимум*** у точці $x = p$, при умові, що $f(x) \le f(p)$ для всіх $x \in I$. Якщо функція *f(x)* має або локальний мінімум, або локальний максимум у точці $x = p$, то говорять, що вона має ***локальний екстремум*** у точці $x = p$.

Критерій першої похідної. Припустимо, що функція $f(x)$ безперервна на відрізку $I = [a; b]$. Крім того, припустимо, що $f'(x)$ визначена для всіх $x \in (a, b)$ за винятком, можливо, точки $x = p$.

Якщо $f'(x) < 0$ на інтервалі (a, p) і $f'(x) > 0$ на (p, b), то $f(p)$ — локальний мінімум.

Якщо $f'(x) > 0$ на інтервалі (a,p) і $f(x) < 0$ на (p, b), то $f(p)$ — локальний максимум.

Критерій другої похідної. Припустимо, що функція $f(x)$ безперервна на відрізку $[a; b]$ і $f'(x)$ і $f''(x)$ визначені на (a, b). Також припустимо, що $p \in (a, b)$ — критична крапка, у якій $f'(p) = 0$.

Якщо $f''(p) > 0$, то значення $f(p)$ є локальним мінімумом *f(x)*.

Якщо $f''(p) < 0$, значення $f(p)$ є локальним максимумом *f(x)*.

Якщо $f''(p) = 0$, то цей критерій не є остаточним.

1.1.2 Метод золотого перетину

Нехай $r \in [0; 1]$ — початковий інтервал. Якщо $0{,}5 < r < 1$, то $0 < 1 - r < 0{,}5$ і інтервал ділиться на три під інтервалів: $[0; 1 - r]$, $[1 - r; r]$ і $[r; 1]$. У процесі розв'язку використовується або стиск вправо

й одержання нового інтервалу [0; r], або стиснення уліво й одержання інтервалу [1- *r*; 1]. Потім отримані під інтервали далі діляться на три під інтервалів в такому ж співвідношенні, як у випадку з інтервалом [0; 1].

Таким чином, необхідно так вибрати *r*, щоб одна зі старих точок була в правильному положенні щодо нового інтервалу. Із цього випливає, що відношення (1 - *r*) : *r* таке ж, як і *r* : 1. Отже, *r* задовольняє рівнянню $1 - r = r^2$, яке можна записати у вигляді квадратного рівняння виду $r^2 + r - 1 = 0$.

Розв'язок *r*, що задовольняє нерівності 0,5 < *r* < 1, дорівнює

$$r = (\sqrt{5} - 1)/2.$$

Функція $f(x)$ повинна задовольняти особливим умовам, які гарантують існування дійсного мінімуму на інтервалі, щоб можна було використовувати пошук мінімуму функції $f(x)$ методом золотого перетину.

1.1.3 Знаходження мінімуму методом інтервалів

Для знаходження мінімуму методом інтервалів необхідно одержати три значення для перевірки:

$$p_0, \quad p_1 = p_0 + h \quad \text{і} \quad p_2 = p_0 + 2h, \tag{1.1}$$

для яких

$$f(p_0) > f(p_1) \quad \text{і} \quad f(p_1) < f(p_2) \tag{1.2}$$

Припустимо, що $f'(p_0) < 0$, тоді $p_0 < p$ і довжина кроку *h* вибирається позитивною. Легко знайти таке значення *h*, щоб три крапки з (1.1) задовольняли нерівностям (1.2).

Перший крок ітерації починається з $h = 1$ (за умови, що $a + 1 < b$)

Випадок (а). Якщо виконуються нерівності (1.2), то крок визначено.

Випадок (б). Якщо $f(p_0) > f(p_1)$ і $f(p_1) > f(p_2)$, то $p_2 < p$. У цьому випадку перевіряються точки, які лежать далі області праворуч ($p > p_2$). При цьому необхідно подвоїти крок і повторити процес.

Випадок (в). Якщо $f(p_0) < f(p_1)$, то невірно обраний крок *h*, при якому шукана крапка *p* була "перестрибнута". Необхідно перевірити значення, найближчі до крапки p_0, зменшити крок в 2 рази й повторити процес.

Коли $f'(p0) > 0$, то довжина кроку h слід вибирати від'ємним й потім розглядати випадки, подібні (а)-(в).

1.1.4 Знаходження мінімуму p - квадратичним наближенням

Якщо точки (1.1) задовольняють нерівностям (1.2), то для знаходження мінімуму функції за допомогою квадратичного наближення, необхідно знайти таке значення $p_{\min}$, яке є наближенням до p. Для цього використовується поліном Лагранжа, побудований на вузлах (1), має вигляд:

$$Q(x) = \frac{y_0(x-p_1)(x-p_2)}{2h^2} - \frac{y_1(x-p_0)(x-p_2)}{h^2} + \frac{y_2(x-p_0)(x-p_1)}{2h^2} \quad (1.3)$$

Похідна від $Q(x)$ дорівнює

$$Q'(x) = \frac{y_0(2x-p_1-p_2)}{2h^2} - \frac{y_1(x-p_0-p_2)}{h^2} + \frac{y_2(2x-p_0-p_1)}{2h^2}; \quad (1.4)$$

Запишемо $Q'(x) = 0$ у вигляді $Q'(p_0+h_{min})=0$:

$$0 = \frac{y_0(2(p_0+h_{min})-p_1-p_2)}{2h^2} - \frac{y_1(4(p_0+h_{min})-2p_0-2p_2)}{2h^2} + \frac{y_2(2(p_0+h_{min})-p_0-p_1)}{2h^2} \quad (1.5)$$

Помножимо кожний член в (1.5) на $2h^2$ та поєднаємо члени, які містять h_{min}:

$$-h_{min}(2y_0-4y_1+2y_2) = y_0(2p_0-p_1-p_2) - y_1(4p_0-2p_0-2p_2) + y_2(2p_0-p_0-p_1) = y_0(-3h) - y_1(-4h) + y_2(-h). \quad (1.6)$$

Останнє рівняння легко одержати відносно h_{min}:

$$h_{min} = \frac{h(4y_1-3y_0-y_2)}{4y_1-2y_0-2y_2}. \quad (1.7)$$

Значення $p_{min} = p_0 + h_{min}$ є кращим наближенням до p, чому p_0. Тому можна замінити p_0 на p_{min} і повторити схему двох описаних вище процесів, щоб визначити нову довжину кроку h і нове h_{min}. Ітерація триває до необхідної точності.

1.2 Хід роботи

1.2.1 Визначити інтервали функції $f(x)$, де вона зростає (зменшується) згідно заданого варіанта (табл. 1.1);

1.2.2 Доведіть, що функція $f(x)$ є унімодальною у межах заданого інтервалу (табл. 1.1)

1.2.3 Визначити локальний мінімум методом золотого перетину функції виду *f(x)* та *f(x,y)* з точністю до восьми десятинних знаків (табл.1.1). Приклад алгоритму програми наведено у додатку А1.

1.2.4 Визначити локальний мінімум, використавши квадратичне інтерполювання функції виду *f(x)* та *f(x,y)* з точністю до восьми десятинних знаків (табл.1.1). Приклад алгоритму програми наведено у додатку А2.

Таблиця 1.1 – Варіанти завдання

Номер завдання	Номер варіанта	
1.2.1	1	$f(x) = 2x^3 - 9x^2 + 12x - 5;$
	2	$f(x)=x/(x + l);$
	3	$f(x) = (x + l)/x;$
	4	$f(x)=x^x;$
1.2.2	1	$f(x) =x^2- 2x +1; [0;4]$
	2	$f(x) =cos(x); [0;3]$
	3	$f(x) = x^x; [1; 10]$
	4	$f(x) = -x(3-x)^{5/3}; [0;3]$
1.2.3/ 1.2.4	1	$f(x) = 4x^3 - 8x^2 - 11x + 5; [0; 2]$
	2	$f(x) = x + 3/x^2; [0,5; 3]$
	3	$f(x) = (x + 2,5)/(4 - x^2); [-1,9; 1,9]$
	4	$f(x) = e^x/x^2; [0,5; 3]$
	5	$f(x) = -2 \sin(x) - \sin(3x)/3; [0; 2]$
	6	$f(x) = -2 \sin(x) + \sin(2x) - 2\sin(3x)/3; [1; 3]$

1.3 Контрольні запитання

1.3.1. Надайте визначення локального мінімуму та локального максимуму функції *f(x)*;

1.3.2 Надайте визначення зростаючої та спадаючої функції;

1.3.3. Надайте визначення унімодальної функції;

1.3.4. У чому полягає метод золотого перетину?

1.3.5. Метод пошуку мінімуму р - квадратичним наближенням.

ЛАБОРАТОРНА РОБОТА 2

ВИВЧЕННЯ МЕТОДІВ МІНІМІЗАЦІЇ ФУНКЦІЇ ДЕКІЛЬКОХ ЗМІННИХ

Мета роботи: Вивчення та надбання навичок застосування методів пошуку оптимальних значень функції декількох змінних.

2.1 Короткі теоретичні відомості

2.1.1 Знаходження екстремальних значень функції $f(x, y)$

Функція $f(x, y)$ має локальний мінімум у точці (p, q), якщо $f(p, q) < f(x, y)$ для кожної точки $(x, y) \in R$.

Функція $f(x, y)$ має локальний максимум у точці (p,q), якщо $f(x, y) < f(p,q)$ для кожної точки $(x, y) \in R$.

Критерій другої похідної. Припустимо також, що функція $f(x, y)$ і її перша й друга часткові похідні безперервні в області R. Припустимо, що $(p, q) \in R$ — критична точка, у якій і $f_x(p, q) = 0$, і $f_y(p, q) = 0$. Часткові похідні вищого порядку використовуються для визначення природи критичної точки.

Якщо $f_{xx}(p, q) \cdot f_{yy}(p, q) - f^2_{xy}(p, q) > 0$ і $f_{xx}(p, q) > 0$, то $f(p, q)$ – локальний мінімум функції $f(x,y)$.

Якщо $f_{xx}(p, q) \cdot f_{yy}(p, q) - f^2_{xy}(p, q) > 0$ і $f_{xx}(p, q) < 0$, то $f(p, q)$ – локальний максимум функції $f(x,y)$.

Якщо $f_{xx}(p, q) \cdot f_{yy}(p, q) - f^2_{xy}(p, q) < 0$, то функція $f(x, y)$ не має локального екстремуму в точці (p, q).

Якщо $f_{xx}(p, q)\, f_{yy}(p, q) - f^2_{xy}(p, q) = 0$, цей критерій не є остаточним.

2.1.2 Метод Нелдера-Мида

Симплекс-Метод знаходження локального мінімуму функції від декількох змінних винайдений Нелдером і Мидом. Для двох змінних симплексом є трикутник, і метод — це схема пошуку, який порівнює значення функції в трьох вершинах трикутника. Найгірша

вершина, у якій функція $f(x,y)$ приймає найбільше значення, відкидається й заміняється новою вершиною. Формується новий трикутник, і пошук триває. При цьому будується послідовність трикутників (вони можуть мати різну форму), значення функції у вершинах якої стають усе менше й менше. Зменшується розмір трикутника, і координати крапки мінімуму знайдені.

У формулюванні алгоритму використовується термін "симплекс" (узагальнений N-мірний трикутник). З його допомогою перебуває мінімум функції від N змінних. Він ефективний і компактний при обчисленні.

2.1.3 Метод найшвидшого спуску (градієнтний метод)

Звернемося до мінімізації функції $f(X)$ від N змінних, де $X = (x_1, x_2, \ldots, x_N)$. Градієнт $f(X)$ — це вектор (векторна функція), визначений як:

$$grad(f_1, f_2, \ldots, f_N), \qquad (2.1)$$

де частинна похідні $f_k = \partial f / \partial x_k$ обчислюються в точці X.

Градієнт (2.1) указує напрямок найбільшої швидкості зростання функції $f(X)$. Отже, $-\ grad\ f(X)$ указує напрямок найбільшого убування. Пошук починається із точки P_0 уздовж лінії, що проходить через P_0 у напрямку $S_0 = -G/||G||$, де $G = grad\ /(P_0)$. При знаходженні точки P_1, де перебуває локальний мінімум, точка X змушена буде потрапити на лінію $X = P_0 + ts_0$. Потім можна обчислити $G = grad\ f(P_1)$ і рухатися в напрямку $S_1 = -G/||G||$. При знаходженні точки P_2, крапка X змушена буде потрапити на лінію $X = P_1 + ts_1$. Ітерація породжує послідовність точок $\{P_k\}$, що володіють властивістю:

$$f(P_0) > f(P_1) > \ldots > f(P_k) > \ldots . \qquad (2.2)$$

Якщо $\lim_{k\to\infty} P_k = P$, то $f(P)$ буде локальним мінімумом для $f(X)$.

Схема методу градієнта

Припустимо, що послідовність точок P_k отримана.

Крок 1. Обчислення градієнта $G = grad\ f(P_k)$.

Крок 2. Визначення напрямку пошуку $S = -G/||G||$.

Крок 3. Визначається єдиний параметр мінімізації $\Phi(t) = f(P_k + ts)$ на інтервалі $[0, b]$, де $t = h_{min}$. Для $\Phi(t)$ перебуває

локальний мінімум. Співвідношення $\Phi(h_{min}) = f(\boldsymbol{P_k} + h_{mins})$ визначає мінімум для $f(\boldsymbol{X})$ уздовж обраної лінії $\boldsymbol{X} = \boldsymbol{P_k} + h_{mins}$.

Крок 4. Побудова наступних крапок $\boldsymbol{P_{k+1}} = \boldsymbol{P_k} + h_{mins}$.

Крок 5. Визначається критерій достатньої мінімізації, тобто чи досить близькі значення функції $f(\boldsymbol{P_k})$ і $f(\boldsymbol{P_{k+1}})$ і чи досить мала відстань $\|\boldsymbol{P}_{k+1} - \boldsymbol{P_k}\|$.

2.2 Хід роботи

2.2.1 Визначити мінімум функції виду *f(x,y)* методом Нелдера-Мида з точністю до восьми десятинних знаків (табл.2.1, 2.2). Приклад алгоритму програми наведено у додатку Б1.

2.2.2 Визначити мінімум функції виду *f(x,y)* методом найскорішого спуску з точністю до восьми десятинних знаків (табл.2.1, 2.2). Приклад алгоритму програми наведено у додатку Б2.

2.2.3 Визначити мінімум функції виду *f(x,y,z,)* або *f(x,y,z,u)* методом Нелдера-Мида з точністю до восьми десятинних знаків (табл.2.3).

2.2.4 Визначити мінімум функції виду *f(x,y,z,)* або *f(x,y,z,u)* градієнтним методом з точністю до восьми десятинних знаків (табл.2.3).

Таблиця 2.1 – Варіанти завдання

Номер варіанта	
1	*f(x,y) = x3 + y3-3x-3y + 5*
2	*f(x,y) =x2+y2+ x-2y-xy + 1*
3	*f(x,y) = x2+xy2-3xy)*
4	*f(x,y) = (x-y)/(x2+y2 + 2)*
5	*f(x,y) = 100(y - x2)2+ (1 - x)2*

Таблиця 2.2 – Варіанти завдання початкових умов

Номер варіанта	Початкові умови	
	до п. 2.2.1	до п. 2.2.2
	(1;2), (2;0) и (2;2)	(1;2)

	(0;0), (2;0) и (2;1)	(0;0,3)
	(0;0), (2;0) и (2;1)	(0,1;0,1)

Продовження таблиці 2.2

	0;0), (0;1) и (1;1)	(0,5;0,11)
	(0;0), (1;0) и (0;2)	(0;0)

Таблиця 2.3 – Варіанти завдання до п.2.2.3 – п.2.2.4

Номер варіанта	Функція	Початкові умови
1	f($x; y; z$) = $2x^2 + 2y^2 + z^2 - 2xy + yz - 7y - 4z$;	(1;1;1);
2		(0;1;0);
3		(1;0;1);
4		(0;0;1);
5	$f(x; y; z; u) = 2(x^2 + y^2 + z^2 + u^2) - x(y + z - u) + yz - 3x - 8$	(1;1;1;1);
6		(0;1;0;0);
7		(1;0;1;0);
8		(0;0;1;1);
9	$f(x; y; z; u) = xyzu + \frac{1}{x} + \frac{1}{y} + + \frac{1}{z} + \frac{1}{u}$	(0.7; 0.7; 0.7; 0.7);
10		(0.5; 0.5; 0.5; 0.5);
11		(0; 0; 0; 0);
12		(1;1;1;1);

2.3 Контрольні запитання

2.3.1. Надайте визначення локального мінімуму та локального максимум функції *f(x;y)*;
2.3.2 Критерій другої похідної функції *f(x;y)*;
2.3.3. Метод Нелдера-Мида**.**
2.3.4. Градієнтний метод.

ЛАБОРАТОРНА РОБОТА 3

СТВОРЕННЯ ЦІЛЬОВОЇ ФУНКЦІЇ ОДНО ТА БАГАТОКОНТУРНИХ КІЛ "ІНДУКТОР – N-РЕЗОНАНСНИЙ КОНТУР" (частина 1)

Мета роботи: Освоїти та реалізувати для подальшого використання цільову функцію одно - та багато резонансних кривих за допомогою пакета MatLAB.

3.1 Завдання на лабораторну роботу

3.1.1 Стосовно свого індивідуального завдання скласти рівняння співвідношення для активної провідності резонансного контуру;

3.1.2 Скласти математичну модель, яка дозволить проводити експерименти для визначення параметрів функції R, L, C з метою накопичення експериментальних даних. Скласти два варіанта програми, з введенням параметрів функції з клавіатури, та посередньо читанням з текстового файлу.

3.1.3. Для реалізація створення емпіричних даних необхідно до вектора теоретичних даних внести похибку **er**, яка може складати 3…15%.

3.1.4. Построїти графічну залежність $G(\omega)$ векторів теоретичних та емпіричних даних.

3.1.5. Скласти математичну модель цільової функції для n-резонансних кривих.

3.2 Короткі теоретичні відомості

У процесах індукційної плавки для ефективного перемішування розплаву співвідношення потужностей визначається співвідношенням частот $P^*_{НЧ}=P_{НЧ}/P_{ВЧ}=\sqrt{\omega_{НЧ}/\omega_{ВЧ}}$. При цьому рядом теоретичних досліджень і багаторазових практичних розробок установлене, що ефективне нагрівання металу при мінімальних масо габаритних показниках перетворювача досягається на високій частоті, що перебуває в діапазоні $\omega_{ВЧ}$ =[10…20кГц]. Тому істотне зменшення потужності перемішування, переданої на низькій частоті, можливо при застосуванні частот $\omega_{НЧ}$ =[20…100Гц];

Індуктор в одно- багаточастотном режимі має різні значення імпедансу на різних частотах, тобто складові імпедансу індуктора залежать від частоти $R_n(\omega)$, $L_n(\omega)$.

3.3 Створення цільової функції n-резонансної кривої

Залежність активної частини провідності від частоти в загальному виді описується для n-резонансної кривої як (рис. 3.1):

$$G(\omega)=\sum_n \frac{R_n}{R_n^2+\left(\omega\cdot L_n-\frac{1}{\omega\cdot C_n}\right)^2}. \qquad (3.1)$$

де $G(\omega)$ – активна частина провідності контуру; R_n – активний опір n-го елемента; L_n – індуктивність n-го елемента; C_n – ємність n-го елемента.

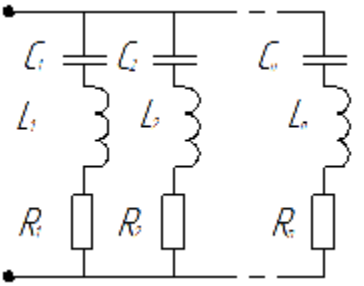


Рисунок 3.1 – Еквівалентна n- контурна схема

3.4 Приклади

Математична модель апроксимації для одно-контурної кривої:

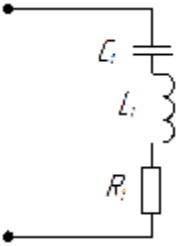
```
%matematicheskaja model experimentov
[c,r,l,a,b,k,er] = textread ('matlab1.txt','%f%n%n%n%n%n%n');
w = 2*pi*a : 2*pi*k : 2*pi*b;
g = r./ (r.^2+(w*l-1 ./(w*c)).^2);
pogr = (rand(1,length(g))-0.5)*2*er;
new = g+g.*pogr;
figure (1);
plot (w,g,w,new)
legend ('teor. izmerenia','empirich.izmerenia')
s = fopen('_file','wb');
c = fwrite(s,a,'int');
j = fwrite(s,b,'int');
d = fwrite(s,k,'int');
u = fwrite(s,new,'double');
fclose(s);
```

Математична модель апроксимації для двох-контурної кривої:

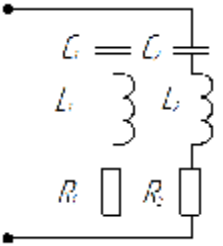
```
%vichodnie parametri po r,l,c,c0
clear all;
```

```
clc;
[c1,c2,r1,r2,L1,L2,a,b,k,er]=textread('matla.txt';%n%n%n%n%n%n%n%n%n%n');
wp1 = sqrt(1/(L1*c1))/(2*pi)
wp2 = sqrt(1/(L2*c2))/(2*pi)
w = 2*pi*a : 2*pi*k : 2*pi*b;
g = r1 ./ (r1^6 2+(w*L1-1 ./(w*c1)).^2) + r2 ./ (r2^2+(w*L2-1./(w*c2)).^2);
pogr = (rand(1 ,length(g))-0.5)*2*er;
new = g+g.*pogr;
pogrl = (rand(1,length(y))-0.5)*2*er;
figure (1);
plot (ws g,ws new)
legend ('teor. aktiw. sostawl. ', 'empirich.aktiwn. sostawl.')
s =fopen('_file2', 'wb');
c = fwrite(s,a, 'int');
j = fwrite(s,b,'int');
d в fwrite(s,k,,'int');
u = fwrite(s,new,'double');
fclose (s);
```

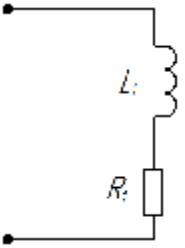
3.5 Індивідуальне завдання



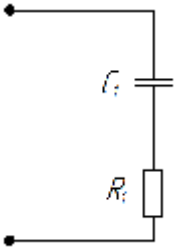
варіант №1



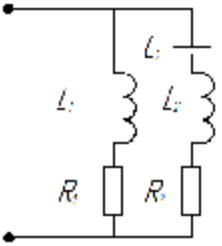
варіант №2



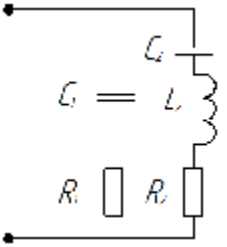
варіант №3



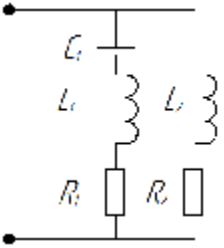
варіант №4



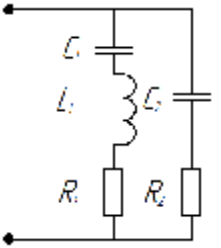
варіант №5



варіант №6



варіант №7



варіант №8

3.6 Контрольні запитання

1. Що таке цільова функція?
2. Критерії оптимізації функцій одно та декількох змінних.
3. Загальне поняття чисельної апроксимації.
4. Загальне поняття квадратичної інтерполяції.
5. Загальне поняття кубічної інтерполяції.

ЛАБОРАТОРНА РОБОТА 4

ОПТИМІЗАЦІЯ ПАРАМЕТРІВ ЕКВІВАЛЕНТНОГО КОЛА "ІНДУКТОР – N-РЕЗОНАНСНИЙ КОНТУР" (частина 2)

Мета роботи: Освоїти та реалізувати алгоритм оптимізації цільової функції одно - та багато резонансних кривих за допомогою пакета MatLAB.

4.1 Короткі теоретичні відомості

В Matlab існує багато методів мінімізації як одномірних *(fminbnd*), так і багатомірних *(fminsearch, lsqnonlin, fminmax, fminuns, fmincon)* функцій, які реалізують різні чисельні методи.

У функції *fminsearch* використовується метод Нелдера-Мида. Перевагою цієї функції є можливість її використання для негладких і розривних цільових функцій.

Форма звертання до цієї функції має вигляд:

$$\mathbf{x = fminsearch(fun,\ x_0,\ options,\ p_1,\ p_2,...).} \qquad (4.1)$$

У тому випадку, коли функція є досить гладкої, для пошуку її мінімуму можна скористатися процедурою *fminunc.* Дана функція реалізує метод Ньютона.

Функція *lsqnonlin* застосовується в тих випадках, коли цільова функція має вигляд:

$$F = 1/2 \cdot \sum f_i^2. \qquad (4.2)$$

У цьому випадку градієнт g і гессіан H функції F виражаються через якобіан J ***вектор***-функції $f|f_1, f_2, ..., f_m|'$:

$$g = J' \times f; \qquad (4.3)$$

$$H = J' \times J + R. \qquad (4.4)$$

Залишковий член R включає другі похідні від F, але в околиці мінімуму їм звичайно зневажають у порівнянні з $J' \times J$. Це дає можливість не обчислювати другі похідні, що значно прискорює роботу з порівнянням із загальним випадком.

Система рівнянь для відшукання вектора зрушення $H \times p = -g$ при заміні H на $J' \times J$ перетворюється в $J' \times J \times p = -J' \times f$. Цей спосіб називається методом Гаусса-Ньютона.

У методі Левенберга-Марквардта матриця $J' \times J$ у лівій частині системи рівнянь заміняється на $J' \times J + \lambda \times I$, де I - одинична матриця, *а* λ – деяке від'ємне число. Для вектора зсуву задається обмеження $|p| < \Delta$, де Δ і λ взаємозалежні.

В функції *Isqnonlin* застосовуються обидва метода: метод Гаусса-Ньютона й Левенберга-Марквардта.

У зверненні до **x=lsqnonlin(fun,x$_0$)** мінімізується сума квадратів компонент вектора - стовпця, який виробляє функція fun (це може бути також матриця). Другий аргумент x_0 - стартова точка для пошуку.

При використанні функції *Isqnonlin,* у якій цільова функція є згорток вектора, можна задавати обмеження. Для функції *Isqnonlin* доступні тільки найпростіші обмеження типу $lb \le x \le ub$. Звертання до функції *Isqnonlin* у цьому випадку має вигляд:

x=lsqnonlin(fun, x$_0$, lb, ub, options, p$_1$,p$_2$,...) (4.5)

Функція *Isqnonlin* може повернути досить багато вихідних параметрів:

[x, resnorm, residual, e_flag, inform, lambda, jacobian]=lsqnonlin(...)

Вихідних аргументів:

x – вектор-розв'язок;

resnorm – значення цільової функції в знайденій точці;

residual - компоненти функції fun(x) у знайденій точці;

e_flag - e_flag=l - розв'язок системи знайдений, e_flag=0 - розв'язок системи не знайдений, e_flag=-l - досягнутий мінімум не є розв'язком системи;

inform – містить три поля:

iterations - кількість ітерацій, виконаних при пошуку кореня;

funccount - кількість звертань до функції fun;

algorithm - найменування алгоритму, використаного для знаходження кореня;

lambda - вектор множників Лагранжа;

jacobian – якобіан функції fun у знайденій точці.

До функції *Isqnonlin* ідейно близька функція *fminimax,* в обох скалярна цільова функція не задається безпосередньо у зверненні, а формується шляхом згортки з компонентів вектора (або матриці), переданого функції. В *Isqnonlin* мінімізується сума квадратів компонент, а в *fminimax* - максимальний компонент. Алгоритм, реалізований в *fminimax,* багаторазово використовує квадратичне програмування, а також пошук за допомогою градієнта й гесіана.

Усі функції оптимізації включають у списку своїх вхідних параметрів перелік властивостей, що впливають на хід ітераційних процесів. Ці властивості представлені структурою options, поля якої формуються за допомогою функції *optimset*. Завдання значення будь-якої властивості проводиться парою параметрів функції *optimset*, перший з яких представляє найменування властивості, а другий - його значення:

$$\textbf{options=optimset('name1',vall,'name2',val2,...);} \qquad (4.6)$$

$$\textbf{x=fxxx(fun,x}_{0,\ldots}\textbf{,options,p}_1\textbf{,p}_2\textbf{,...)} \qquad (4.7)$$

Перед вхідним параметром options може розташовуватися деяка кількість вхідних параметрів, перелік яких для кожної функції оптимізації індивідуальний.

Параметри p_1, p_2,..., розташовувані слідом за структурою options, передаються оптимізованій функції разом з незалежним аргументом x – fun(x,pl,p2,...). У табл. 4.1 наведений список параметрів, керуючих процесом знаходження мінімуму функцій.

Функція мінімізації *fminsearch*, вимагає, щоб оптимізаційна функція мала вигляд:

$$f = \Sigma(G_э - C_m)^2, \tag{4.8}$$

де $G_э$ - вектор емпіричних даних, C_m – теоретично отримані значення.

Для функції *lsqnonlin* необхідно створити цільову функцію у вигляді:

$$f = \Sigma(G_э - C_m) \ . \tag{4.9}$$

тому що цей метод сам будує необхідну функцію по (4.8).

4.2 Приклади

Формування цільової функції оптимізації методом Нелдера-Мида (для функції *fminsearch)*:

```
function opt = mfs(V,g,w)
r = V(1);
l = V(2)*1e-3;
c = V(3)*1e-12;
ot = (g-r./(r^2+(w*L-1 ./(w*c)).^2)).^2;
opt = sum(ot,2);
```

Формування цільової функції оптимізації методом Гаусса-Ньютона и Левенберга-Марквардта (для функції *Isqnonlin)*:

```
function opt = rew(V,g,w);
r = V(1);
I = V(2)*1e-3;
c = V(3)*1e-12;
opt = g-r./(r^2+(w*L-1 ./(w*c)).^2);
```

Таблиця 4.1. Перелік параметрів, які керують процесом пошуку мінімуму функції

MaxFunEvals	максимальна кількість звернень до функції fun
Maxlter	максимальна кількість ітерацій
TolFun	припинення ітерацій при досягненні точності по значенню функції
TolX	припинення ітерацій при досягненні мінімального кроку по X

4.3 Завдання на лабораторну роботу

4.3.1 Відповідно свого індивідуального завдання створити цільові функції для застосування методів оптимізації.

4.3.2 Скласти математичну модель оптимізації цільової функції методом Нелдера-Мида застосовуючи функцію ***fminsearch***. Приклад програми наведено у додатку В.

4.3.3. Скласти математичну модель оптимізації цільової функції метод Гаусса-Ньютона и Левенберга-Марквардта, застосовуючи функцію **lsqnonlin**. Приклад програми наведено у додатку Г.

4.3.4. Провести порівняльний аналіз двох методів оптимізації при заданої точності 1e-3, 1e-5 та 1e-10 за критеріями: кількість ітерацій та час розрахунку .

4.4 Контрольні запитання

1. Чим, та в яких умовах застосовуються методи оптимізації Нелдера-Мида та Гаусса-Ньютона и Левенберга-Марквардта?
2. У якій формі має вигляд цільова функція при застосуванні метода Нелдера-Мида?
3. У якій формі має вигляд цільова функція при застосуванні метода Гаусса-Ньютона та Левенберга-Марквардта.
4. Який метод має найменшу кількість ітерації та час розрахунку?

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

Основна література

1. Реклейтис Г. Оптимизация в технике [Текст] / Реклейтис Г., Рейвиндран А., Рэгедел К., М: Мир, 1986. – 351с. (рос. мовою).
2. Черногудский И.Г. Методы оптимизации в теории управления [Текст] / И.Г. Черногудский, - СПб.: Питер, 2004. – 256с. (рос. мовою).
3. Бойко И.В. Методы и алгоритмы решения задач оптимизации [Текст] / И.В. Бойко, Б.Н. Бублик, П.Н. Зинько – К.: Вища школа. Головное изд-во, 1983 – 512с. (рос. мовою)
4. Сухарев А.Г. Курс методов оптимизации [Текст] / А.Г. Сухарев, А.В. Тимохов, В.В. Федоров, М.: ФИЗМАТЛИТ, 2005. – 368с. (рос. мовою)
5. Банди Б. Методы оптимизации [Текст] / Б. Банди. – М.: Радио и связь, 1988-128с. (рос. мовою)

Додаткова література

6. Акулич И.Л. Математическое программирование в примерах и задачах [Текст] / И.Л. Акулич. М.: Высш. шк., 1986. – 319с. (рос. мовою).

7. Руденко В.С. Преобразовательная техника [Текст] / В.С. Руденко, В.И. Сенько, И.М. Чиженко. – Киев: Вища школа, 1983. – 431с. (рос. мовою).

8. Шавьолкін О.О. Перетворювальна техніка: навчальний посібник / О.О. Шавьолкін, О.М.Наливайко. – Краматорськ: ДДМА, 2008. - 326с. (рос. мовою)

9. Дащенко А. Ф. MATLAB в инженерных и научных расчетах [Текст] / А. Ф. Дащенко, В. Х. Кириллов, Л. В. Коломиец, В. Ф. Оробей. – Одесса: Астропринт, 2003. – 210с. (рос. мовою).

Додаток А

Програма А1. Метод пошуку мінімуму методом золотого перетину. (приклад).

```
function[S,E,G]=golden(f,a,b,delta,epsilon)
%Вход       - f - функція, яка вводиться як рядок ‘ f ’
%           - a и b - крайні точки інтервалу
%           - delta – припустиме відхилення для абсцис
%           - epsilon - припустиме відхилення для ординат
%Выход   - S=(p,yp) - містить абсцису p й ординату yp мінімуму
%           - E=(dp,dy) - містить грані похибки для p та yp
%         - G - матриця розміру n x 4: k-та строка містить
%           [ak ск dk bk]; значення a, c, d и b на к-й ітерації
rl=(sqrt(5)-1)/2;
r2=rl^2;
h=b-a;
      ya=feval(f,a);
      yb=feval(f,b);
      c=a+r2*h;
```

```
        d=a+rl*h;
        yc=feval(f,c);
yd=feval(f,d);
к=1;
A(k)=a;B(k)=b;C(k)=c;D(k)=d;
while(abs(yb-ya)>epsilon)|(h>delta) k=k+l;
      if(yc<yd)
         b=d;
         yb=yd;
         d=c;
         yd=yc;
         h=b-a;
         c=a+r2*h;
         yc=feval(f,c);
      else
         a=c;
         ya=yc;
         c=d;
         yc=yd;
         h=b-a;
         d=a+rl*h;
         yd=feval(f,d);
      end
      A(k)=a;B(k)=b;C(k)=c;D(k)=d;
end
dp=abs(b-a);
dy=abs(yb-ya);
p=a;
yp=ya;
if(yb<ya)
      p=b;
      yp=yb;
end
G=[A' C' D' B'] ;
S=[p yp];
E=[dp dy];
```

Програма А2. Метод пошуку локального мінімуму з використанням квадратичного інтерполювання. (приклад).

```
function [p, yp, dp, dy, P] =quadmin (f,a,b,delta, epsilon)
%Вход       - f - функція, яка вводиться як рядок >f>
%          - a и b - крайні точки інтервалу
%          - delta – припустиме значення для абсцис
%          - epsilon - припустиме значення для ординат
%Выход      - p - абсциса мінімуму
%          - yp - ордината мінімуму
%          - dp - грань похибки для p
%          - dy - грань похибки для yp
%          - P - вектор ітерації

p0=a;
maxj=20;
maxk=30;
big=le6;
err=l;
k=l;
```

```
P(k)=p0;
cond=0;
h=l;
if(abs(p0)>le4),h=abs(pO)/1e4; end
while(k<maxk&err>epsilon&cond~=5)
    f1=(feval(f,p0+0.00001)-feval(f,p0-0.00001))/0.00002 ;
    if(f1>0),h=-abs(h);end
    pl=p0+h; p2=p0+2*h;
    pmin=p0;
    y0=feval(f,p0);
    yl=feval(f,pl);
    y2=feval(f,p2);
    ymin=y0;
    cond=0;
    j=0;
    % Визначення такого h, що yl<y0&yl<y2
    while(j<maxj&abs(h)>delta&cond==0)
        if (y0<=yl),
        p2=pl;
        y2=y1;
        h=h/2;
        pl=p0+h;
        yl=feval(f,pl);
     else
        if(y2<yl),
          pl=p2;
          yl=y2;
          h=2*h;
          p2=p0+2*h;
           y2=feval(f,p2);
         else
            cond=-l;
         end
      end
j=j+1;
if (abs(h)>big | abs(p0)>big), cond=5;end
end
if(cond==5) ,
```

```
   pmin=pl;
   ymin=feval(f,pl);
else
     % Квадратичне інтерполювання для знаходження yp
     d=4*y1-2*y0-2*y2;
     if(d<0),
        hmin=h*(4*yl-3*y0-y2)/d;
            else
               hmin=h/3;
               cond=4;
            end
            pmin=p0+hmin;
            ymin=feval(f,pmin);
            h=abs(h);
            h0=abs(hmin);
            hl=abs(hmin-h);
            h2=abs(hmin-2*h);
% Визначення величини наступного h
            if(h0<h),h=h0;end
            if(hl<h),h=hl;end
            if(h2<h),h=h2;end
            if(h==0),h=hmin;end
            if(h<delta),cond=l;end
            if (abs(h)>big|abs(pmin)>big),cond=5;end
% Критерій зупинення для мінімізації
            e0=abs(y0-ymin);
            el=abs(yl-ymin) ;
            e2=abs(y2-ymin);
            if(e0~=0 к eO<err),err=eO;end
            if(el~=0 к eKerr) ,err=el;end
            if (e2~=0 к 2<err), err=e2 ; end
            if (e0~=0 к el==0 к e2==0), error=0 ; end
            if(err<epsilon),cond=2; end
            p0=pmin;
            k=k+l;
            P(k)=p0;
        end
        if(cond==2&h<delta),cond=3; end
```

```
end
p=p0;
dp=h;
yp=feval(f,p);
dy=err;
```

Додаток Б

Програма Б1. Метод мінімізації Нелдера-Мида (приклад).

```
function z=f(V)
z=0; x=V(l); y=V(2);
z=x.^2-4x+y.^2-y-x.^y;
function [V0, y0, dV, dy] =nelder (F, V, mini, maxi, epsilon,
show)
%Вход          - F - функція, яка вводиться як рядок >F>
%              - V - матриця розміру 3 x n, яка містить вихідний
%                    симплекс
%              - mini & maxi - мінімальна та максимальна
%                    кількість ітерацій
%               - epsilon – припустиме відхилення
%              - show == 1 - показує число ітерацій (P и Q)
%Выход         - V0 - вершина для мінімуму
%              - y0 - значення функції F(V0)
%               - dV - розмір остаточного симплекса
%               - dy - грань похибки для мінімуму
%             - P - матриця, яка містить ітерації вершин
```

```
%                - Q - масив, який містить ітерації для F(P)
if nargin==5,
   show=0;
end
[mm n]=size(V) ;
% Послідовність вершин
for
j=l:n+l
Z=V(j,1:
n);
Y(j)=fev
al(F,Z);
end
[mm lo]=min(Y);
[mm hi]=max(Y);
li=hi;
h
o
=
l
o
;
for j=l:n+l
     if ( j ~=lo& j ~=hi&Y (j)<=Y(li))
     li=j;
     end
     if ( j ~=hi& j ~=lo&Y ( j)>=Y(ho))
     ho=j ;
     end
end
cnt=0;
% Початок алгоритма Нелдера-Мида
while(Y(hi)>Y(lo)+epsilon&cnt<max1)|cnt<minl
      S=zeros(l,l:n);
      for j=l:n+l
           S=S+V(j,l:n);
      end
```

```
        M=(S-
        V(hi,1:n))/n;
        R=2*M-V(hi,1:n);
        yR=feval(F,R);
        if(yR<Y(ho))
             if(Y(li)<yR)
             V(hi,1:n)=R;
             Y(hi)=yR;
        else
                E=2*R-M;
                yE=feval(F,E);
                if(yE<Y(li))
                   V(hi,l:n)=E;
                   Y(hi)=yE;
                else
                   V(hi,1:n)=R;
                   Y(hi)=yR;
                end
        end
else
        if(yR<Y(hi))
           V(hi,l:n)=R;
           Y(hi)=yR;
        end
        C=(V(hi,1:n)+M)/2;
        yC=feval(F,C);
        C2=(M+R)/2;
        yC2=feval(F,C2);
        if(yC2<yC)
           C=C2;
           yC=yC2;
        end
        if(yC<Y(hi))
           V(hi,1:n)=C;
           Y(hi)=yC;
        else
           for j=l:n+l
                if (j~=lo)
```

```
                V(j,l:n)-(V(j,l:n)+V(lo,l:n))/2;
                  Z=V(j,1:n);
                  Y(j)=feval(F,Z);
                end
            end
        end
    end
    [mm lo]=min(Y);
    [mm hi]=max(Y);
    li=hi;
    ho=lo;
    for j=l:n+l
        if (j~=lo&j ~=hi&Y (j) <=Y (li))
            li=j;
        end
        if (j~=hi&j ~=lo&Y (j) >=Y (ho))
            ho=j ;
        end
    end
    cnt=cnt+l;
    P(cnt,:)=V(lo,:);
    Q(cnt)=Y(lo);
end
% Кінець алгоритма Нелдера-Мида
% Визначення розміру симплекса
snorm=0;
for j=l:n+l
  s=norm(V(j)-V(lo));
  if(s>=snorm)
    snorm=s;
  end
end
Q=Q';
V0=V(lo,l:n);
y0=Y(lo);
dV=snorm;
dy=abs(Y(hi)-Y(lo));
if (show==l)
```

```
   disp(P);
   disp(Q);
end
```

Програма Б2. Градієнтний метод. (приклад).

```
function z=G(V)
z=zeros(l,2);
x=V(l) ;y=V(2);
g=[2x-4-y  2*y-l-x];
z=-(l/norm(g))*g;

function[P0,y0,err]=grads(F,G,P0,maxi,delta,epsilon,show)
%Вход      - F - функція, яка вводиться, як рядок F
%             - G = - (l/norm(gradF)) * gradF; вибір напрямку,
%              вводиться, як рядок 'G'
%             - PO - начальна точка
%             - maxl - максимальна кількість ітерацій
%             - delta - припустиме відхилення для hmin у єдиному
%               параметрі мінімізації у вибраному напрямку
%            - epsilon - припустиме відхилення для похибки у y0
%            - show; якщо show==l, ітерації виводяться на дісплей
% Выход     - P0 - точка мінімума
%            - y0 - значення функції F(P0)
%            - err - грань похибки для y0
```

```
%              - P - вектор, який містить ітерації

if nargin==5,show=0;end
[mm n]=]size(P0);
maxj=10; big=le8; h=l;
P=zeros(maxj,n+l);
len=norm(P0);
y0=feval(F,P0);
if (len>e4),h=len/le4;end
err=l;cnt=0;cond=0;
P(cnt+1,: ) = [P0 y0];
while (cnt<maxl&cond~=5& (h>delta | err>epsilon))
    %Розрахунок напрямку пошуку
    S=feval(G,P0);
    %Початок вибору параметра квадратичної мінімізації
    Pl=P0+h*S;
    P2=P0+2*h*S;
    yl=feval(F,Pl);
    y2=feval(F,P2);
    cond=0;j=0;
    while(j <maxj &cond==0)
          len=norm(P0);
          if (y0<yl);
          P2=P1;
          y2=yl;
          h=h/2;
          Pl=P0+h*S;
          yl=feval(F,Pl);
    else
if(y2<yl)
   P1=P2;
   yl=y2;
   h=2*h;
P2=P0+2*h*S; y2=feval(F,P2);
else
   cond=-l;
end
            end
```

```
    j=j+1;
    if(h<delta),cond=l;end
    if(abs(h)>bigIlen>big),cond=5;end
end
if(cond==5)
    Pmin=Pl;
    ymin=yl;
else
    d=4*yl-2*y0-2*y2;
    if(d<0)
       hmin=h*(4*yl-3*y0-y2)/d;
    else
       cond=4;
       hmin=h/3;
    end
    % Построение следующей точки
     Pmin=P0+hmin*S;
    ymin=feval(F,Pmin);
    % Визначення величини наступного h
    h0=abs(hmin);
    hl=abs(hmin-h);
    h2=abs(hmin-2*h);
    if(h0<h),h=h0;end
    if(hl<h),h=hl;end
    if(h2<h),h=h2;end
    if(h==0),h=hmin;end
    if(h<delta),cond=1;end
    % Критерій останова для мінімізації
    e0=abs(y0-ymin);
    el=abs(yl-ymin);
    e2=abs(y2-ymin);
    if (e0~=0&e0<err) ,err=e0;end
    if (e0&el<err),err=el;end
    if (e2~=0&e2<err) ,err=e2;end
    if(e0==0&el==0&e2==0),err=0;end
    if(err<epsilon),cond=2;end
    if(cond==2&h<delta),cond=3;end
end
```

```
        cnt=cnt+l;
        P(cnt+1,:)=[Pmin ymin];
        P0=Pmin;
Y0=y
min;
end
if(show==
l)
    disp(P
);
end
```

Додаток В

Програма В1. Приклад оптимізації параметрів одно контурного кола за методом Нелдера-Мида.

```
%vhodnie parametri po izmerennim znachenijam g; method fminsearch;
 tic % контроль часу виконання програми
 s = fopen('Jile7rb');
 a = fread(s,1 ,'int');
j = fread(s,1/int');
k = fread(s,1,'int');
h = fix((j-a)/k+1);
 [g,i] = fread(s,h,'double');
w = 2*pi*a : 2*pi*k : 2*pi*j;
 p = max(g) n = 1;
while g(n)~=p,
n = n+1;
end;
n %maximum_1
g(n)
wp1 = w(n)/(2*pi)
```

```
p1 = 9(n)/2;
ep = 2.5e-4;
kj = 1;
while kj<=n&abs(g(kj)-p1)>ep,
kj = kj+1;
end;
kl = length(g);
while kl>=n&abs(g(kl)-p1)>ep,
kl = kl-1;
end;
kl
raz1 = w(kl)-w(kj)
r1 = 1/p;
q = wp1/raz1;
l1 = (q*r1/wp1)*1e3;
c1 = (1/(4*pi.^2*wp1.^2*L1))*1e15;
options=optimset('MaxFunEvalsM0000,'MaxlterM0000,'ToIX',1e-10,
'Tolfun',1e-10);
V = [r1 L1 c1]
[X,FVAL,EXITFLAG,OUTPUT] = fminsearch('mfs',V, options, g',w);
X
OUTPUT
r = X(1);
I = X(2)*1e-3;
c = X(3)*1e-12;
g1 = r./ (r.^2+(w*l-1 ./(w*c)).^2);
figure(3);
plot (w,g,w,g1 ,w(kj),g(kj),'*r',w(kl),g(kl),'*r');
legend ('teoretic g'.'praktic g');
toc % контроль часу виконання програми
```

Програма В2. Приклад оптимізації параметрів одно контурного кола за методом Нелдера-Мида.

```
%vhodnie parametri po izmerennim znachenijam g;
%method fminsearch;
tic
s=fopen('_file2','rb)
a = fread(s,1,'int');
```

```
j = fread(s,1 ,'int');
k = fread(s,1 ,'inf);
h = fix((j-a)/k+1);
[g,i] = fread(s,h,'double');
[y,d] = fread(s,h,'double');
w = 2*pi*a : 2*pi*k : 2*pi*j;
ep = 1e-4;
s = maxi(g);
wp1 = w(s(2)) / (2*pi)
p1 = g(s(2))/2; ep = 5e-4;
kj = 1;
while kj<=s(2)&abs(g(kj)-p1)>ep,
kj = kj+1;
end;
raz1 = 2*(w(s(2))-w(kj));
% _____________________________
wp2 = w(s(1)) / (2*pi)
p2 = g(s(1))/2;
ep = 5e-4;
ki = length(g);
while ki>=s(1)&abs(g(ki)-p2)>ep,
ki = ki-1;
end;
raz2 = 2*(w(ki)-w(s(1)));
% _____________________________
%kontur_1
r1 = 1/g(s(2))
q1 =wp1 /raz1;
l1= (q1*r1/wp1)*1e3
c1 = (1/(4*pi^2*wp1^2*L1))*1e15
% _____________________________
%kontur_2
r2 = 1/g(s(1));
q2 = wp2 / raz2;
l2= (q2*r2/wp2)*1e3
c2 = (1 /(4*pi ^2*wp2 ^2* L2))* 1e15
% _____________________________
V0 = [r1 L1 c1 r2 L2 c2]
```

```
options=optimset('MaxFunEvals',20000,'Maxlter',10000, 'TolX',1 e-
10,Tolfun',1 e-10);
[X,FVAL,EXITFLAG,OUTPU"n=fminsearch('mfs2',V0,options,g',w);
X
OUTPUT
r1 = X(1);
l1= X(2)*1e-3;
c1 = X(3)*1e-12;
r2 = X(4);
l2= X(5)*1e-3;
c2 = X(6)*1e-12;
g1 = r1 ./ (r1 ^2+(w*l1-1 ./(w*c1)).^2) + r2 ./ (r2^2+(w*l2-1 ./(w*c2)).^2);
figure (2)
plot (w,g,w,g1 ,w(kj),g(kj),'*r',w(ki),g(ki),'*r')
legend ('teoretic g'.'praktic g')
toc
```

Додаток Г

Програма Г.1. Приклад оптимізації параметрів одно контурного кола за методами Гаусса-Ньютона и Левенберга-Марквардта

```
%vhodnie parametri po izmerennim znachenijam g
%method Isqnonlin;
tic
s = fopen('_file', 'rb');
a = fread(s,1,'int');
j = fread(s,1 ,'int');
k = fread(s,1 ,'inf);
h = fix((j-a)/k+1);
[g,i] = fread(s,h,'double');
s = fopen('_file','rb');
a = fread(s,1 ,'int');
j = tread (s,1 ,'int');
k = fread(s,1,'int');
h = fix((j-a)/k+1);
[g,i] = fread(s,h,'double');
w = 2*pi*a : 2*pi*k : 2*pi*j;
```

```
p = max(g)
n  =  1;
while g(n)~=p,
n = n+1;
end;
n %maximum_1
fd = g(n)
wp1 = w(n)/(2*pi)
p1 = 9(n)/2
ep = 2.3e-4;
kj = 1;
while kj<=n&abs(g(kj)-p1)>ep,
kj = kj+1;
end;
kl = length(g);
while kl>=n&abs(g(kl)-p1)>ep,
kl = kl-1; end; kl
raz1 = w(kl)-w(kj)
q = wp1/raz1;
r1 = 1/p;
l1 = (q*r1/wp1)*1e3;
c1 = (1/(4*pi^2*wp1^2*L1))*1e15;
V0 = [r1 L1 c1]
beg = [0 0 0];
en = [1000 1000 1000];
options=optimset('MaxFunEvals',10000,'Maxlter',10000,TolX',1e10,'Tolfu
n\1e-10);
[X,RESNORM,RESIDUAL,EXITFLAG,OUTPU"n=IsqnonlinCrew'.VO,b
eg,en,options,g',w);
X
OUTPUT
r = X(1);
I = X(2)*1e-3;
c = X(3)*1e-12;
gl = r./ (r^2+(w*l-1 ./(w*c)).^2);
figure (2)
w(kj)
plot (w,g,w,g1 ,w(kj),g(kj),'*r',w(kl),g(kl),'*r')
```

```
legend ('teoretic g'/praktic g')
toc
```

Програма Г.2. Приклад оптимізації параметрів двох контурного кола за методами Гаусса-Ньютона и Левенберга-Марквардта

```
%vhodnie parametri po izmerennim znachenijam g
%method Isqnonlin;
tic
s = fopen('_file2','rb');
a = fread(s,1,'int');
j = fread(s,1,'int');
k = fread(s,1,'int');
h = fix((j-a)/k+1);
[g,i] = fread(s,h,'double');
[y,d] = fread(s,h,'double');
w = 2*pi*a : 2*pi*k : 2*pi*j;
ep = 1e-4;
s = maxi (g);
wp1 = w(s(2)) / (2*pi)
p1 = g(s(2))/2;
ep = 5e-4;
kj = 1;
while kj<=s(2)&abs(g(kj)-p1)>ep,
kj = kj+1;
end;
raz1 = 2*(w(s(2))-w(kj));
% _____

wp2 = w(s(1)) / (2*pi)
p2 = g(s(1))/2; ep = 5e-4;
ki = length(g);
while ki>=s(1)&abs(g(ki)-p2)>ep,
ki = ki-1;
end;
raz2 = 2*(w(ki)-w(s(1)));
```

```
% _____________________________

%kontur_1
r1 = 1/g(s(2))
q1 =wp1 /raz1;
l1= (q1*r1/wp1)*1e3
c1 = (1/(4*pi^2*wp1^2*L1))*1e15

% _____________________________
%kontur_2
r2 = 1/g(s(1));
q2 = wp2 / raz2;
l2= (q2*r2/wp2)*1e3
c2 = (1 /(4*pi ^2*wp2 ^2* L2))* 1 e 15

% _____________________________
V0 = [r1 L1 c1 r2 L2 c2]
beg = [0 0 0 0 0 0];
en = [1000 1000 1000 1000 1000 1000];
options=optimset('MaxFunEvals',20000,'Maxlter',10000, TolX',
1e-10,'Tolfun',1e-10);
[X,RESNORM,RESIDUAL,EXITFLAG,OUTPUT]=lsqnonlin('rew2',V0,b
eg,en,options,g',w);
X
OUTPUT
r1 = X(1);
l1 = X(2)*1e-3;
c1 = X(3)*1e-12;
r2 = X(4);
12 = X(5)*1e-3;
c2 = X(6)*1e-12;
g1 = r1 ./ (r1^2+(w*l1-1./w*c1)).^2) + r2 ./ (r2^2+(w*l2-1 ,/(w*c2)).^2);
% _____________________________
figure (2)
p1ot (w,g,w,g1,w(κj),g(κj),'*r',w(ki),g(ki),'*r')
1egend ('teoretik g','praktic g')
toc
```