Форма № Н- 9.02

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ ТА ІНФОРМАТИКИ

КАФЕДРА ТЕОРІЇ ОПТИМАЛЬНИХ ПРОЦЕСІВ

Магістерська робота

“ТРИКРОКОВИЙ РЕКУРСИВНИЙ МЕТОД НЬЮТОНА РОЗВ’ЯЗУВАННЯ ЗАДАЧ МІНІМІЗАЦІЇ”

Виконала:
студентка V курсу, групи ПМА-51м
напряму підготовки (спеціальності)
8.04030301 – Системний аналіз і управління

Грицик Ю. В.
(прізвище та ініціали)

Керівник проф. Бартіш М. Я.
(прізвище та ініціали)

Рецензент _____
(прізвище та ініціали)

Львів – 2015

РЕФЕРАТ

Дипломна робота містить 42 сторінки. У ній розміщено 7 ілюстрацій, 9 таблиць.

Об’єктом дослідження є задачі оптимізації, а саме класичний метод Ньютона та його модифікації у трикроковий рекурсивний метод Ньютона з глибиною рекурсії p.

Метою цієї роботи є: провести дослідження модифікації методу Ньютона у трикроковий рекурсивний метод Ньютона. Підтвердити теоретичні результати цього методу результатами отриманими на практиці за допомогою його програмної реалізації та порівняти ефективність цієї модифікації з класичним методом Ньютона.

Методи дослідження: аналітичні, теоретичні, моделювання, порівняння результатів, статистичних даних.

Все частіше з’являються все нові і нові математичні моделі та функції що їх описують. Їх складність стрімко зростає. Тому дослідження методів безумовної мінімізації та способів вибору оптимальних серед них і на далі займатимуть важливе місце серед задач чисельних методів.

Дуже важливо вибрати метод, який дасть змогу знайти розв’язок задачі при якомога менших обчислювальних затратах, оскільки не вдало вибраний метод, вимагає великої кількості обчислень.

На підставі порівняння результатів отриманих внаслідок проведення апробації класичного методу Ньютона та трикрокового рекурсивного методу Ньютона на низці тестових функцій, можна стверджувати, що обчислювальна складність трикрокового рекурсивного методу Ньютона, у порівнянні з класичним методом Ньютона суттєво не збільшується. Також значну перевагу модифікованого методу Ньютона видно при збільшенні розмірності простору, який є областю визначення функції, яку мінізуємо.

Результати отримані за допомогою програмної реалізації методів можна використати як підтвердження результатів отриманих теоретично.

Ключові слова: швидкість збіжності, трикроковий рекурсивний метод, глибина рекурсії, модифікація методу Ньютона.

Three-step recursive Newton's method for solving minimization problem

Yuliia Hrytsyk

Ivan Franko National University of Lviv

We study a three step recursive Newton method based on three step Newton method but is more effective in computational properties. We research a theorem where convergence of three step recursive Newton method is justified and rate of convergence is established.

Numerical results are presented on several functions. We compare results of two methods.

ЗМІСТ

ВСТУП

Люди завжди шукали способів відшукати найкращий розв’язок якомога швидше і легше. Перед здійсненням якихось заходів вони думали, які можуть бути результати і у залежності від цього приймали рішення, тим чи іншим способом вибираючи параметри - способи виконання завдань.

Тривалий час обрати спосіб можна було лише базуючись на власному досвіді, без застосування спеціального математичного аналізу. З прискореним розвитком різних галузей науки і техніки, з’явилась необхідність застосування матетаматичних методів оптимізації, вони набули дуже вагомого значення.

Оскільки поведінку будь-якого фізичного об'єкта можна описати рівнянням або системою рівнянь (тобто створити математичну модель реального об'єкта), то завданням інженера є підбір функції та методу її мінімізації з заданою точністю, для отримання оптимального розв’язку.

З впровадженням ЕОМ у всі сфери людського життя та стрімким розвитком технологій все частіше з’являються все нові і нові математичні моделі та функції що їх описують. Їх складність стрімко зростає. Якщо задачі мінімізації простіших функцій були дость простими, і їх розв’язок було легко знайти без застосування обчислювальної техніки, то знаходження мінімуму складних функцій, є досить трудомісткою задачею.

Тому важливе місце серед завдань математичного аналізу займають дослідження методів оптимізації та способів вибору оптимальних серед них для розв’язування конкретного типу задач. Оскільки ці методи є досить складними з обчислювальної точки зору, то для їх реалізації варто використати програмне забезпечення.

З іншої сторони, при використанні комп’ютера для знаходження розв’язку, ми неминуче отримуємо похибку в результатах, оскільки пам’ять комп’ютера обмежена і всі дані зберігаються з деяким заокругленням. З’являється нове

питання, про вибір точності обчислень, що також впливає на вибір методу, для отримання результатів, які будуть максимально близькими до точних.

Також важливо вибрати метод, який дасть змогу знайти розв'язок задачі при якомога менших обчислювальних затратах, оскільки не вдало вибраний метод, вимагає великої кількості обчислень.

Будь-який числовий метод розв'язування задачі оптимізації грунтується на точному або наближеному обчисленні певних характеристик (значення функції мети, вектора-градієнта, матриці Гессе). На основі наявної інформації будуємо наближення до розв'язку задачі.

Якщо об'єм пам'яті ЕОМ малий, то для розв'язуваня задачі великої розмірності не варто використовувати алгоритми, які вимагають обчислення других похідних. Доцільніше застосовувати методи нульового або першого порядку, які повльніше збігаються до розв'язку, але вимагають менше пам'яті для збереження даних. Для недиференційованих функцій не можна застосовувати методи, які вимагають обчислення вектора-градієнта.

Для розв'язування цієї задачі можна використати різницеві методи, метод простої ітерації, градієнтний метод, але ці методи вимагають досить складних обчислень. Градієнтний метод є методом першого порядку, у ньому для визначення напрямку зменшення функції, використовується значення похідної $f'(x)$, тобто значення x_{k+1} обчислюється за формулою

$$x_{k+1} = x_k - \alpha_k f'(x_k), \qquad k = 0,1,\dots$$

де параметр α_k, який задає довжину кроку, можна вибрати так:

$$\alpha_k = \arg\min_{\alpha>0} f\left(x_k - \alpha f'(x_k)\right)$$

Цей метод має лінійну швидкість збіжності. Якщо ж функція, аналогічно до нашої є двічі неперервно диференційовна і похідні $f'(x)$ та $f''(x)$ знаходяться досить легко, то краще застосовувати методи мінімізації другого порядку, які використовують квадратичну частину розкладу функції в ряд Тейлора.

$$f(x, x_*) = f(x_*) + (f'(x_*), x - x_*) + \frac{1}{2}(f''(x_*)(x - x_*), x - x_*)$$

Оскільки квадратична частина розкладу апроксимує функцію набагато точніше від лінійної, то методи другого порядку будуть збігатися швидше.

Методом мінімізації другого порядку є метод Ньютона, а його модифікації є методами першого порядку. У них матриця других похідних апроксимується і використовується інформація про значення градієнтів функції $f(x)$.

Нами запропонований до використання трикроковий рекурсивний метод Ньютона.

Метою цієї роботи є проведення дослідження модифікації методу Ньютона у трикроковий рекурсивний метод Ньютона. Підтвердження теоретичних результатів цього методу результатами отриманими на практиці за допомогою його програмної реалізації та порівняння ефективності цієї модифікації з класичним методом Ньютона.

РОЗДІЛ 1
РЕКУРСИВНИЙ АНАЛОГ ТРИКРОКОВОГО МЕТОДУ НЬЮТОНА

1.1. Постановка задачі

Розглянемо задачу мінімізації

$$f(x) \to \min, \qquad (1)$$

де, $x \in R^n$, $f: R \to R^n$, $f(x) \in C^2(R^n)$

Для розв'язування такого типу задач існує безліч методів, але найкращим вважається той, який дає точніший розв'язок при менших обчислювальних затратах.

Методи нульового порядку (при побудові методу використовують інформацію лише про саму функцію) і першого порядку (використовують інформацію про функцію та її першу похідну), є простішими, але погано апроксимують функцію, через що мають нижчу швидкість збіжності і вимагають великої кількості ітерацій для отримання розв'язку з великою точністю.

Функція, яку ми розглядаємо, є двічі диференційовною, що дозволяє використати інформацію про другу похідну мінімізуючої функції $f(x)$ при побудові наближення і побудувати метод другого порядку. Такі методи краще апроксимують функцію і відповідно мають вищу швидкість збіжності. Тому вони дають розв'язок задачі з високою точністю за меншу кількість ітерацій.

Методами другого порядку є метод Ньютона та його різницеві аналоги. Метод Ньютона має вигляд

$$x_{k+1} = x_k - \left(f''(x_k)\right)^{-1} f'(x_k), \qquad k = 0,1,\dots$$

Проте він має велику обчислювальну складність, тому використовують його модифікації, для повнішого використання наявної інформації про функцію та зменшення обчислювальної складності.

Модифікація класичного методу Ньютона у трикроковий метод Ньютона має вигляд

$$u_k = x_k - \left(f''(x_k)\right)^{-1} f'(x_k)$$

$$v_k = x_k - \alpha_k f'(x_k)$$

$$x_{k+1} = \underset{t}{\operatorname{argmin}}\ f(u_k - t(u_k - v_k)), \qquad k = 0,1,\ldots$$

тобто додатково використовує значення градієнта функції.

Нами розглядається рекурсивний аналог трикрокового методу

$$u_k = x_k - \alpha_k \left(f''(\tilde{x}_k)\right)^{-1} f'(x_k),$$

$$v_k = x_k - \beta_k f'(x_k),$$

$$x_{k+1} = \underset{t}{\operatorname{argmin}}\ f(u_k - t(v_k - u_k)), \qquad k = 0,1,\ldots$$

де $\tilde{x}_k = x_{\left[\frac{k}{p}\right]p}$, $[a]$ – ціла частина від a, $\alpha_k \in (0,1]$, $\beta_k > 0$ такі, що виконуються умови монотонності. Він вимагає ще менше обчислювальних затрат.

1.2. Класичний метод Ньютона

Маючи певне наближення x_k до розв'язку поставленої задачі мінімізації, функцію $f(x)$ в цій точці можна записати у вигляді

$$f(x) - f(x_k) = (f'(x_k), x - x_k) + \frac{1}{2}(f''(x_k)(x - x_k), x - x_k) + o(\|x - x_k\|^2)$$

Використавши квадратичну частину цього приросту

$$f_k(x) = (f'(x_k), x - x_k) + \frac{1}{2}(f''(x_k)(x - x_k), x - x_k)$$

можемо визначити x_{k+1} як розв'язок нашої задачі мінімізації. Очевидно, що $f_k''(x) = f''(x_k)$ і з опуклості функції $f(x)$ маємо, що $f_k(x)$ також опукла.

Тоді необхідні і достатні умови мінімуму мають вигляд

$$f'(x_k) + f''(x_k)(x - x_k) = 0.$$

Отже, класичний метод Ньютона має вигляд:

$$x_{k+1} = x_k - \left(f''(x_k)\right)^{-1} f'(x_k), \quad k = 0,1,\ldots. \tag{2}$$

Дослідимо його збіжність.

Теорема 1. Нехай $f(x) \in C^2(R^n)$ сильно опукла з константою $\theta > 0$ і $f''(x)$ задовольняє умову Ліпшиця

$$\|f''(x) - f''(y)\| \le M\|x - y\|, x, y \in R^n,$$

де $\infty > M > 0$, початкова точка x_0 вибрана так, що $\|f'(x_0)\| < 8q\Theta^2/M$, де $q \in (0,1)$. Тоді послідовність $\{x_n\}$ визначена за (2), збігається до точки x_* і виконується оцінка

$$\|x_k - x_*\| \le \frac{4\Theta q^{2^k}}{M}. \tag{3}$$

Доведення. Оскільки $f(x)$ сильно опукла, то точка x_* існує і єдина, крім того, виконується умова

$$(f''(x)h, h) \ge 2\Theta\|h\|^2, \text{ де } x, h \in R^n.$$

Матриця $f''(x)$ додатно визначена, тому невироджена і метод (2) коректно визначений. Оскільки виконується нерівність

$$(f'(x) - f'(x_*), x - x_*) \ge 2\Theta\|x - x_*\|^2,$$

то можемо записати

$$\|x_k - x_*\|^2 \le \frac{1}{2\Theta}\|f'(x_k)\|\|x_k - x_*\|,$$

Тобто

$$\|x_k - x_*\| \le \frac{1}{2\Theta}\|f'(x_k)\|.$$

Враховуючи співвідношення

$$f'(x_{k+1}) = f'(x_{k+1}) - f'(x_k) - f''(x_k)(x_{k+1} - x_k),$$

отримаємо

$$\|f'(x_{k+1})\| \le \left\| \int_0^1 \left(f''\left(x_{k+1} + \tau(x_{k+1} - x_k)\right) - f''(x_k) \right) d\tau \right\| \|x_{k+1} - x_k\| \qquad (4)$$
$$\le\le \frac{M}{2}\|x_{k+1} - x_k\|^2 = \frac{M}{2}\|[f''(x_k)]^{-1}\|^2 \|f'(x_k)\|^2.$$

Оцінимо величину $[f''(x_k)]^{-1}$.

Оскільки

$$(f''(x)h, h) \ge 2\Theta\|h\|^2,$$

то прийнявши

$$h = [f''(x_k)]^{-1} y$$

одержуємо

$$\|[f''(x_k)]^{-1} y\|^2 \le \frac{1}{2\Theta}(y, [f''(x_k)]^{-1} y) \le \frac{1}{2\Theta}\|y\| \|[f''(x_k)]^{-1} y\|,$$

отож,

$$\|[f''(x_k)]^{-1}\| = \max_{\|y\|=1} \frac{\|[f''(x_k)]^{-1} y\|}{\|y\|} \le \frac{1}{2\Theta}.$$

Враховуючи цю оцінку, з (4) запишемо

$$\|f'(x_{k+1})\| \le \frac{M}{8\Theta^2}\|f'(x_k)\|^2 \text{ при } k = 0,1,\ldots.$$

Тепер

$$\|f'(x_k)\| \le \frac{M}{8\Theta^2}\|f'(x_{k-1})\|^2 \le \left(\frac{M}{8\Theta^2}\right)^{2k-1}\|f'(x_k)\|^2\|f'(x_0)\|^{2^k} = \frac{8q\Theta^2}{M}\mathrm{q}^{2^k},$$

де

$$q = \frac{M}{8\Theta^2}\|f'(x_0)\|$$

і легко отримати оцінку (3).

Збіжність методу доведена [2].

Для доброго початкового наближення вона є квадратичною. Проте, вибір початкового наближення є досить складною задачею. Крім того, самі операції є досить трудомісткими в сенсі кількості обчислень. Тому застосовують модифікації методу Ньютона, які зменшують трудомісткість алгоритму та послаблюють вимоги до початкового наближення.

1.3. Трикроковий алгоритм мінімізації функції

У [4] був запропонований трикроковий алгоритм мінімізації функцій, побудований на базі методу Ньютона, який має високу швидкість збіжності для доброго початкового наближення, та градієнтного методу, для якого можна вибирати довільне початкове наближення. На кожномку кроці за допомогою даних методів обчислюються два проміжні наближення до розв’язку, а наступне наближення алгоритму шукається, як мінімум на прямій, що з’єднує отримані точки.

За цим методом, маючи наближення x_k, потрібно виконати один крок класичним методом Ньютона:

$$u_k = x_k - \left(f''(x_k)\right)^{-1} f'(x_k) \tag{5}$$

І один крок за градієнтним методом

$$v_k = x_k - \alpha_k f'(x_k) \tag{6}$$

Виконання одного кроку за градієнтним методом не вимагає значних додаткових обчислень, при обчисленні u_k – ми використовуємо вже відоме значення $f'(x_k)$, а отже ефективніше використовується інформація отримана на попередньому кроці.

Маючи значення u_k і v_k, визначаємо наближення x_{k+1} за формулою

$$x_{k+1} = u_k - \beta_k (u_k - v_k) \tag{7}$$

де

$$f(u_k - \beta_k(u_k - v_k)) = \min_\beta f(u_k - \beta_k(u_k - v_k)).$$

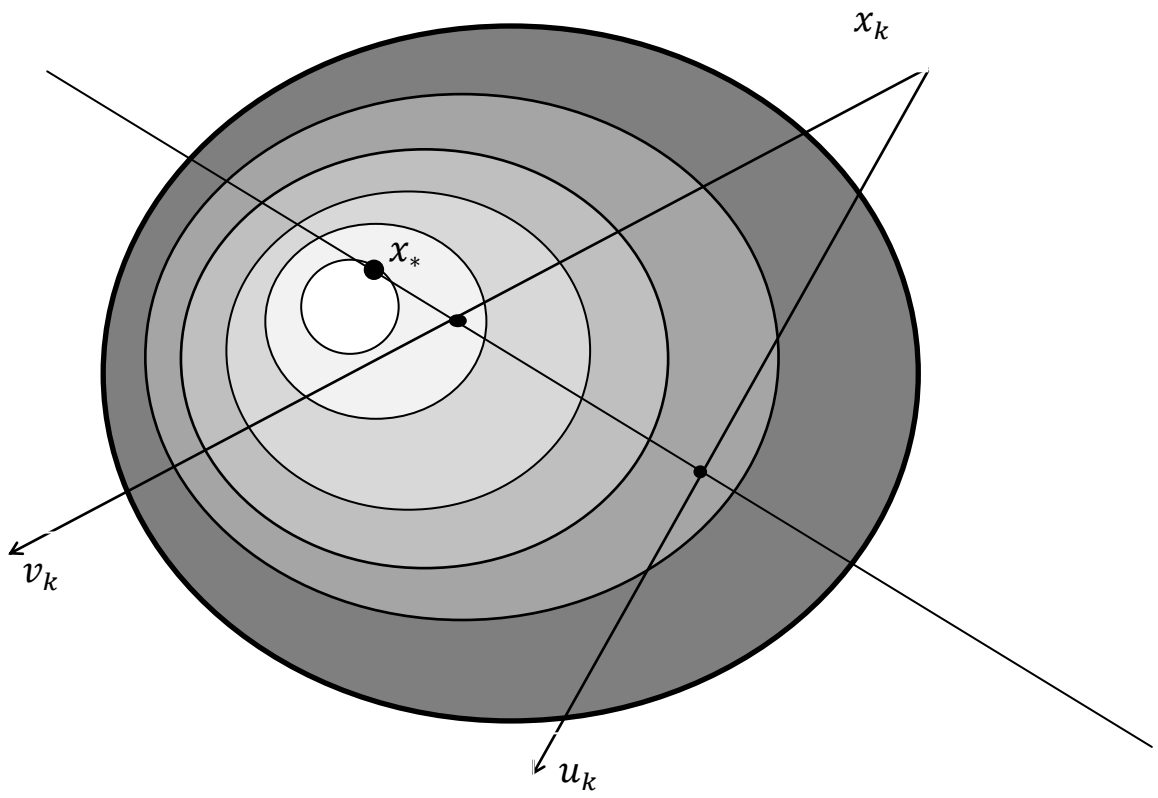

Рис. 1.3.1 – Графічне представлення трикрокового методу

Характеристику збіжності цієї модифікації дає наступна теорема.

Теорема 2. Нехай $D \subset R^n$ – відкрита опукла область, а також:

1. $f(x) \in C^2(R^n)$ і для всіх $x \in D, y \in R^n$ сильно-опукла з константою опуклості m>0 і
$$\|f''(x)\| \le M, 0 < m \le M; \tag{8}$$
2. для всіх $x, y \in D$ $f''(x)$ задовольняє умову Ліпшиця
$$\|f''(x) - f''(y)\| \le L\|x - y\|, 0 < L < \infty \tag{9}$$
3. початкове наближення x_0, вибрано так, що виконується умова
$$q = C\big(f(x_0) - f(x_*)\big) < 1, C = \frac{L^2 M^5}{2m^8}.$$

Тоді послідовність $\{x_k\}$, визначена алгоритмом, збігається до x_* - розв'язку задачі (1) і виконуються оцінки

$$f(x_k) - f(x_*) \le C^{-1} q^{2^{k+1}}, \qquad \|x_{k+1} - x_k\| \le C_1 q^{2^k}, \qquad C_1 = \sqrt{\frac{2}{mC}}.$$

Доведення. Існування і єдиність розв'язку задачі (1) випливають з умови сильної опуклості функції $f(x)$ у ряд Тейлора і умову (8), отримаємо

$$\begin{aligned} f(u_k) - f(x_*) &= (f'(x_*), u_k - x_*) + \frac{1}{2}(f''(\tilde{x}_k)(u_k - x_*), u_k - x_*) \\ &= \frac{1}{2}(f''(\tilde{x}_k)(u_k - x_*), u_k - x_*) \le \frac{M}{2}\|u_k - x_*\|^2 \end{aligned} \tag{10}$$

де $\tilde{x}_k = u_k + \xi(u_k - x_*),\ \xi \in (0,1)$.

Використовуючи властивість опуклих функцій можемо записати

$$(f'(u_k), u_k - x_*) = (f'(u_k) - f'(x_*), u_k - x_*) \geq m\|u_k - x_*\|^2.$$

Отже,

$$m\|u_k - x_*\|^2 \leq \|u_k - x_*\|\|f'(u_k)\|.$$

Звідки

$$\mu_k\|u_k - x_*\| \leq \frac{1}{m}\|f'(u_k)\|, \quad \mu_k < 1.$$

Враховуючи умову (9) і співвідношення

$$f'(u_k) = f'(u_k) - f'(x_k) - f''(x_k)(u_k - x_k),$$

отримаємо

$$\|f'(u_k)\| \leq \left\| \int_0^1 \left(f''\left(u_k + \tau(x_k - u_k)\right) - f''(x_k) \right) d\tau \right\| \|u_k - x_k\| \leq \frac{L}{2}\|u_k - x_k\|^2$$

$$= \frac{L}{2}\|[f''(x_k)]^{-1}\|^2\|f'(x_k)\|^2 \leq \frac{L}{2m^2}\|f'(x_k)\|^2,$$

$$\|f'(x_k)\| = \|f'(x_k) - f'(x_*)\| = \|f''(\tilde{x}_k)(x_k - x_*)\| \leq M\|x_k - x_*\|,$$

$$f(x_k) - f(x_*) = \frac{1}{2}(f''(\tilde{x}_k)(x_k - x_*), x_k - x_*) \leq \frac{m}{2}\|x_k - x_*\|^2,$$

$$\|x_k - x_*\|^2 \leq \frac{2}{m}\left(f(x_k) - f(x_*)\right).$$

Продовжимо оцінку (10). Враховуючи отримані оцінки, одержуємо

$$f(u_k) - f(x_*) \leq \frac{M}{2}\left(\frac{1}{m}\|f'(u_k)\|\right)^2 \leq \frac{M}{2}\left(\frac{1}{m}\frac{L}{2m^2}\|f'(x_k)\|^2\right)^2$$

$$= \frac{ML^2}{(2m^2)^3}\|f'(x_k)\|^4 \leq \frac{ML^2}{(2m^2)^3}(M\|x_k - x_*\|)^4$$

$$\leq \frac{M^5L^2}{(2m^2)^3}\left(\frac{2}{m}\right)^2\left(f(x_k) - f(x_*)\right)^2.$$

Отже,

$$f(u_k) - f(x_*) \le C\big(f(x_k) - f(x_*)\big)^2,$$

де

$$C = \frac{L^2 M^5}{2m^2 (m^2)^3}.$$

Зрозуміло, що

$$f(x_{k+1}) - f(x_*) \le \mu_k (f(u_k) - f(x_*)) \le C\big(f(x_k) - f(x_*)\big)^2.$$

Тоді

$$C\big(f(x_{k+1}) - f(x_*)\big) \le \Big(C\big(f(x_k) - f(x_*)\big)\Big)^2 \le \Big(C\Big(\big(f(x_k) - f(x_*)\big)\Big)^2\Big)^2 \le \cdots$$
$$\le C^{-1} q^{2^{k+1}},$$

Де

$$q = C\big(f(x_0) - f(x_*)\big) < 1.$$

Отже,

$$f(x_{k+1}) - f(x_*) \le C^{-1} q^{2^{k+1}}.$$

Метод (5)-(7) збігається до розв’язку. Теорему доведено.

Послідовність $\{x_k\}$, отримана за такою схемою, володіє кращими властивостями в сенсі швидкості збіжності, ніж послідовності отримані за допомогою застосування лише класичного методу Ньютона, або лише градієнтного методу.

Також метод є ефективнішим у випадку вироджених функцій, бо зачення, яке ми не можемо обчислити застосувавши класичний метод Ньютона, ми можемо компенсувати значенням, яке отримаємо на другій ітерації за

допомогою застосування градієнтного методу, тим самим зменшивши похибку методу.

В цей же час обчислювальні затрати у порівнянні з методом Ньютона зростають не істотно. Особливо переваги методу видно при застосуванні його для обчислення функцій великих розмірностей.

1.4. Рекурсивний аналог трикрокового методу Ньютона

Пізніше, на базі трикрокової модифікації методу Ньютона, був запропонований її покращений аналог, який з використовує глибину рекурсії. Тобто, як і в трикроковому методі Ньютона, на першому кроці, обчислюється значення u_k, за допомогою класичного методу Ньютона. Далі виконується один крок градієнтним методом для обчислення значення v_k і знаходимо мінімум функції на прямій побудованій на точкахu_k і v_k.

Суть методу полягає в тому, що матриця Гессе обчислюється на першій ітерації, і на p наступних залишається тою самою, а на $p+1$ ітерації знову перераховується. Величина p називається глибиною рекурсії.

Отже, ця модифікація матиме вигляд:

$$u_k = x_k - \alpha_k \left(f''(\tilde{x}_k)\right)^{-1} f'(x_k), \tag{11}$$

$$v_k = x_k - \beta_k f'(x_k), \tag{12}$$

$$x_{k+1} = \underset{t}{\operatorname{argmin}}\ f(u_k - t(v_k - u_k)), \tag{13}$$

де $\tilde{x}_k = x_{\left[\frac{k}{p}\right]p}$, $[a]$ – ціла частина від a, $\alpha_k \in (0,1]$, $\beta_k > 0$ такі, що виконуються умови монотонності.

Дослідимо збіжність цієї модифікації.

Теорема. Нехай:

1) $f(x) \in C^2(R^n)$;

2)$\forall x \in D, h \in R^n, = \{x \in R^n : f(x) \leq f(x_0)\}$

$$m\|h\| \leq \left(f''(x)h, h\right) \leq M\|h\|,$$

де $0 < m \leq M < \infty$;

3) $\forall x,\ y \in D$, $f''(x)$ задовільняє умову Ліпшиця

$$\left\|f''(x) - f''(y)\right\| \leq L\|x - y\|,$$

де $0 < L < \infty$;

4) початкове наближення x_0 вибирають таким, що виконується умова

$$\mu = \frac{1}{\sqrt[p]{9}} C\left(f(x_0) - f(x_*)\right) < 1,$$

де $C = \frac{9ML^2}{2m^4}$.

Тоді послідовність $\{x_k\}$ визначена за (11) - (13), збігається до розв'язку x_* задачі (1) і справджується оцінка

$$f(x_k) - f(x_*) \leq$$

$$\leq \mu^{(p+1)^k - 1} \left(\prod_{j=0}^{k-1} \left(\prod_{i=1}^{p} \gamma_j^{(i)} \right)^{(p+1)^{k-1-j}} \right) \left(f(x_0) - f(x_*)\right) \qquad (14)$$

де $\gamma_j^{(i)} \leq 1; k = 1,2, \ldots$.

Доведення. Оскільки згідно з умовою 2) функція $f(x)$ сильно опукла, то існує розв'язок x_* і $f'(x_*) = 0$, крім того, справджується оцінка [1]

$$\left\|\left(f^{"}(x)\right)^{-1}\right\| \leq \frac{1}{m}.$$

Можемо записати

$$f(x) - f(x_*) = \frac{1}{2}\int_0^1 \left(f^{"}\left(x_* + \tau(x - x_*)\right)(x - x_*), (x - x_*)\right) d\tau.$$

Звідси

$$\frac{m}{2}\|x - x_*\|^2 \leq f(x) - f(x_*) \leq \frac{M}{2}\|x - x_*\|^2$$

і

$$\|x - x_*\|^2 \leq \frac{2}{m}\left(f(x) - f(x_*)\right).$$

Тоді

$$u_0^{(1)} - x_* - \left(f^{"}(x_0)\right)^{-1}\left(f'(x) - f'(x_*)\right) =$$

$$= \left(f^{"}(x_k)\right)^{-1}\left(\int_0^1 \left(f^{"}(x_0) - f^{"}\left(x_* + \tau(x_0 - x_*)\right)\right) d\tau.\right)(x_0 - x_*)$$

або

$$\left\|u_0^{(1)} - x_*\right\| \leq \frac{1}{m} L \int_0^1 (1 - \tau) d\tau \|x_0 - x_*\|^2 \leq \frac{L}{2m}\|x_0 - x_*\|^2.$$

З іншого боку,

$$f\left(u_0^{(1)}\right) - f(x_*) \leq \frac{M}{2}\left\|u_0^{(1)} - x_*\right\|^2 \leq \frac{M}{2}\left(\frac{L}{2m}\|x_0 - x_*\|^2\right)^2 \leq$$

$$\leq \frac{M}{2}\left(\frac{L}{m^2}\left(f(x_0)-f(x_*)\right)\right)^2 = \frac{M}{2}\left(\frac{L}{m^2}\right)^2\left(f(x_0)-f(x_*)\right)^2 \leq$$

$$\leq \frac{C}{9}\left(f(x_0)-f(x_*)\right)^2.$$

Згідно з (11)-(13) отримаємо

$$f\left(x_0^{(1)}\right)-f(x_*) \leq \frac{C}{9}\gamma_0^{(1)}\left(f(x_0)-f(x_*)\right)^2,$$

де $\gamma_0^{(1)} \in (0,1]$.

Нехай

$$f\left(x_0^{(p-1)}\right)-f(x_*) \leq \frac{C^{p-1}}{9}\gamma_0^{(p-1)}\dots\gamma_0^{(1)}\left(f\left(x_0^{(0)}\right)-f(x_*)\right)^p.$$

Тут $\gamma_0^{(1)} \in (0,1], i=1,\dots,p$.

Тоді

$$u_0^{(p)} = x_0^{(p-1)} - \left(f^{"}(x_0)\right)^{-1}\left(f'\left(x_0^{(p-1)}\right)\right)$$

і одержуємо

$$u_0^{(p)} - x_* = x_0^{(p-1)} - \left(f^{"}(x_0)\right)^{-1}\left(f'\left(x_0^{(p-1)}\right) - f'(x_*)\right) =$$

$$= \left(f^{"}(x_0)\right)^{-1}\left(\int_0^1\left(f^{"}\left(x_* + \tau\left(x_0^{(p-1)}-x_*\right)\right)\right)d\tau\right)\left(x_0^{(p-1)}-x_*\right)$$

або

$$\left\|u_0^{(p)} - x_*\right\| \leq \frac{1}{m}L\int_0^1\left(\|x_0-x_*\| + \tau\left\|x_0^{(p-1)}-x_*\right\|\right)d\tau\left\|x_0^{(p-1)}-x_*\right\| \leq$$

$$\leq \frac{3L}{m^2}\sqrt{\left(f(x_0)-f(x_*)\right)\left(f\left(x_0^{(p-1)}\right)-f(x_*)\right)}.$$

Тоді

$$f\left(u_0^{(p)}\right)-f(x_*)\leq \frac{M}{2}\left\|u_0^{(p)}-x_*\right\|^2\leq$$

$$\leq \frac{9L^2M}{2m^4}\left(f(x_0)-f(x_*)\right)\left(f\left(x_0^{(p-1)}\right)-f(x_*)\right)\leq$$

$$\leq \frac{9L^2M}{2m^4}\left(f(x_0)-f(x_*)\right)\frac{1}{9}C^{p-1}\gamma_0^{(p-1)}\dots\gamma_0^{(1)}\left(f(x_0)-f(x_*)\right)^p\leq$$

$$\leq \frac{1}{9}C^{p-1}\gamma_0^{(p-1)}\dots\gamma_0^{(1)}\left(f\left(x_0^{(0)}\right)-f(x_*)\right)^{p+1}=\mu^p\gamma_0^{(p-1)}\dots\gamma_0^{(1)}\left(f(x_0)-f(x_*)\right).$$

Отже, можна записати так:

$$f(x_1)-f(x_*)\leq \mu^{(p+1)^1-1}\left(\prod_{i=1}^{p}\gamma_j^{(i)}\right)\left(f(x_0)-f(x_*)\right)$$

Нехай

$$f(x_k)-f(x_*)\leq \mu^{(p+1)^k-1}\left(\prod_{j=0}^{k-1}\left(\prod_{i=1}^{p}\gamma_j^{(i)}\right)^{(p+1)^{k-1-j}}\right)\left(f(x_0)-f(x_*)\right).$$

Аналогічно до попередніх викладок отримаємо

$$f(x_{k+1})-f(x_*)\leq \frac{1}{9}C^p\gamma_k^{(p)}\dots\gamma_k^{(1)}\left(f(x_k)-f(x_*)\right)^{p+1}\leq$$

$$\leq \frac{1}{9}C^p\gamma_k^{(p)}\dots\gamma_k^{(1)}\left(\mu^{(p+1)^k-1}\left(\prod_{j=0}^{k-1}\left(\prod_{i=1}^{p}\gamma_j^{(i)}\right)^{(p+1)^{k-1-j}}\right)\left(f(x_0)-f(x_*)\right)\right)^{p+1}\leq$$

$$\leq \mu^{(p+1)^{k+1}-1}\left(\prod_{j=0}^{k}\left(\prod_{i=1}^{p}\gamma_j^{(i)}\right)^{(p+1)^{k-j}}\right)\left(f(x_0)-f(x_*)\right).$$

Теорема доведена [1].

Частковими випадками методу трикрокового рекурсивного методу Ньютона є трикроковий метод Ньютона при $p=1$, і рекурсивний метод Ньютона з глибиною рекурсії p, де $u_k^{(j)}=x_k^{(j)}$, а також звичайний метод Ньютона, при $p=1$ і $u_k^1=x_{k+1}$.

Перевагами методу є -менша обчислювальна складність, ніж у трикроковому методі Ньютона, що є особливо вагомим при великих розмірностях, ефективність для вироджених функцій.

Головним недоліком модифікації, як і класичного методу Ньютона є складність вибору початкового наближення.

1.5. Вибір оптимальної глибини рекурсії

Розглянемо питаня про оптимальну глибину рекурсії в сенсі кількості обчислень, використавши оцінку (14). Для обчислення $x_k^{(1)}$ потрібно Q_1 обчислень, а для $x_k^{(j)}$ $(j>1)$ – $Q_2<Q_1$. Отже, знайдемо p як розв'язок задачі

$$k(Q_1+(p-1)Q_2)\to\min_p$$

$$\mu^{(p+1)^1-1}\left(\prod_{j=0}^{k-1}\left(\prod_{i=1}^{p}\gamma_j^{(i)}\right)^{(p+1)^{k-1-j}}\right)\left(f(x_0)-f(x_*)\right)\leq\varepsilon. \qquad (15)$$

Проте, обчислення в цій задачі є достатньо складними, через що знайти її розв'язок є майже не можливо. Спробуємо оцінити кількість обчислень по-іншому. За одиницю обчислення візьмемо знаходження значення функції $f(x)$ і приймемо $\gamma_j^{(i)}=1$. Тоді можна вважати, що

$$Q_1 = \frac{n(n+1)}{2} + n + \delta,$$

$$Q_2 = n + \delta$$

обчислень. У цьому випадку δ – кількість обчислень функції $f(x)$ при одновимірній мінімізації. Оскільки відомо, що вона є досить невеликою, візьмемо $10 < \delta \leq 20$. Оскільки решта обчислень не впливає на результат, то ними можна знехтувати. Тоді задачу (15) можна переписати так:

$$\left(\frac{n(n+1)}{2} + n + \delta + (p-1)(n+\delta)\right) \to \min_p$$

$$\mu^{(p+1)^1-1}\left(f(x_0) - f(x_*)\right) \leq \varepsilon, \qquad (16)$$

яка дає наближений розв'язок задачі (15) [1]

РОЗДІЛ 2
ПРОГРАМНА РЕАЛІЗАЦІЯ

2.1.Архітектура проекту

Результатом виконаної роботи є програмний проект Трикроковий рекурсивний метод Ньютона. Мета використання проекту – знаходження безумовного мінімуму функції за заданим початковим наближенням. Проект реалізовує класичний метод Ньютона та трикроковий рекурсивний метод Ньютона. Він демонструє ефективність рекурсивного методу у порівнянні з класичним методом Ньютона у сенсі кількості обчислень. Реалізація є простою у використанні і зручною для демонстрації результатів.

Для демонстрації створено шість демо прикладів, на яких ілюструється робота алгоритмів.

Проект реалізований на платформі .Net у середовищі розробки Visual Studio 2010 з використанням мови програмування C#.

Для реалізації алгоритмів було використано бібліотеку MathNet.Numerics.2.5.0, яка полегшує роботу з багатовимірними матрицями і векторами. У ній реалізовані методи для множення матриць, векторів, скалярних величин, обернення матриць.

2.2.Технічний опис проекту

При створенні проекту був використаний простір імен `ThreeStepsRecursiveNewtonMethod`. У ньому був написаний клас Newton, який знаходиться у файлі `Newton.cs`. У ньому реалізовані методи для роботи алгоритмів класичного методу Ньютона та його модифікації у трикроковий рекурсивний метод Ньютона:

- `Newton` (конструктор, для забезпечення коректності даних);

- `func`(функція, яку мімізуємо);
- `DCK` (релізує ДСК алгоритм для пошуку локалізованого відрізка, на якому будемо мінімізувати функцію);
- `argmin` (для знаходження аргументу, який мінімізує функцію методом золотого поділу, на проміжку отриманому методом ДСК);
- `NewtonMethod` (реалізує класичний метод Ньютона)
- `ThreeStepsRecursiveNewtonMethod` (реалізує алгоритм трикрокового рекурсивного методу Ньютона з глибиною);
- `GetDoc` (для отримання даних про результати отримані на кожній ітерації).

```
namespace ThreeStepsRecursiveNewtonMethod
{
    class Newton
    {
        int n;
        int functionNumber;
        DenseVector uk;
        DenseVector vk;
        DenseVector x;
        double gamma;
        double eps;
        int depth;
        Documentator doc;
        Function f;

        struct interval...

        //конструктор
        public Newton(DenseVector x1, int n1, double eps1, double gamma1, int depth1, int fNumber)

        //функція для мінімізації
        double func(int n, DenseVector x, double t)...

        //локалізація проміжку
        interval DCK(DenseVector x, double h0)...

        //знаходження аргументу, який мінімізує функцію
        double argmin()...

        //класичний метод Ньютона
        public DenseVector NewtonMethod()...

        //трикроковий рекурсивний метод Ньютона
        public DenseVector ThreeStepsRecursiveNewtonMethod()...

        //документування проміжних даних
        public Documentator GetDoc()...
    }
}
```

Рис.2.2.1. Програмний файл `Newton.cs`

Для додаткової зручності був реалізований клас Documentator, який полегшує збір інформації про проміжні дані під час роботи програми, та результатів.

```
class Documentator
{
    struct Record
    {
        public String alpha;
        public String beta;
        public String uk;
        public String vk;
        public String xNext;
        public String fxNext;
    }

    ArrayList List;
    //змінні для збереження кількості обчислень відповідних характеристик функції
    int numberOfCalcFunction;
    int numberOfCalcPrimeFunc;
    int numberOfCalcDoubPrimeFunc;

    //конструктор
    public Documentator()...

    //збільшити кількість обчислень певної характеристики
    public void incCalcFunc(int n)...
    public void incCalcPrimeFunc(int n)...
    public void incCalcDoublePrimeFunc(int n)...

    //отримати кількість обчислень характеристики
    public int GetCalcFunc()...
    public int GetCalcPrimeFunc()...
    public int GetCalcDoublePrimeFunc()...

    //додати запис
    public void Add(double alpha1, double beta1, DenseVector uk1, DenseVector vk1,

    //повертає найперший запис у вигляді масиву
    public String[] GetNextArray()...

    //кількість документованих записів
    public int GetSize()...

    //конвертує масив проміжних даних у текстовий рядок
    public String ConvertToString(DenseVector x, int n)...
}
```

Рис.2.2.2 Програмний файл Documentator.cs

Також був написаний клас `Function` для збереження гнучкої архітектури проекту і можливості швидкого додавання нових функцій. У ньому були реалізовані методи

- `Function` (конструктор, для коректного встановлення номера функції);
- `function` (для отримання результату обчислення функції);
- `prime` (для отримання результату обчислення похідної);
- `doublePrime` (для отримання другої похідної функції)
- також реалізовані самі функції, які обчислюють ці значення.

```
namespace ThreeStepsRecursiveNewtonMethod
{
    class Function
    {
        int numberOfFunction;

        public Function(int number)...

        public double functionBeil(int n, DenseVector x)...
        public double functionRosenbrok(int n, DenseVector x)...
        public double functionShtraf1(int n, DenseVector x)...
        public double functionShtraf2(int n, DenseVector x)...
        public double functionVaitXolst(int n, DenseVector x)...
        public double functionPayel(int n, DenseVector x)...
        public double function(DenseVector x, int n)...

        public DenseVector primeBeil(int n, DenseVector x)...
        public DenseVector primeRosenbrok(int n, DenseVector x)...
        public DenseVector primeShtraf1(int n, DenseVector x)...
        public DenseVector primeShtraf2(int n, DenseVector x)...
        public DenseVector primeVaitXolst(int n, DenseVector x)...
        public DenseVector primePayel(int n, DenseVector x)...
        public DenseVector prime(DenseVector x, int n)...

        public DenseMatrix doublePrimeBeil(int n, DenseVector x)...
        public DenseMatrix doublePrimeRosenbrok(int n, DenseVector x)...
        public DenseMatrix doublePrimeShtraf1(int n, DenseVector x)...
        public DenseMatrix doublePrimeShtraf2(int n, DenseVector x)...
        public DenseMatrix doublePrimeVaitXolst(int n, DenseVector x)...
        public DenseMatrix doublePrimePayel(int n, DenseVector x)...
        public DenseMatrix doublePrime(DenseVector x, int n)...

    }
}
```

Рис.2.2.3. Програмний файл Function.cs

2.3 Опис інтефейсу

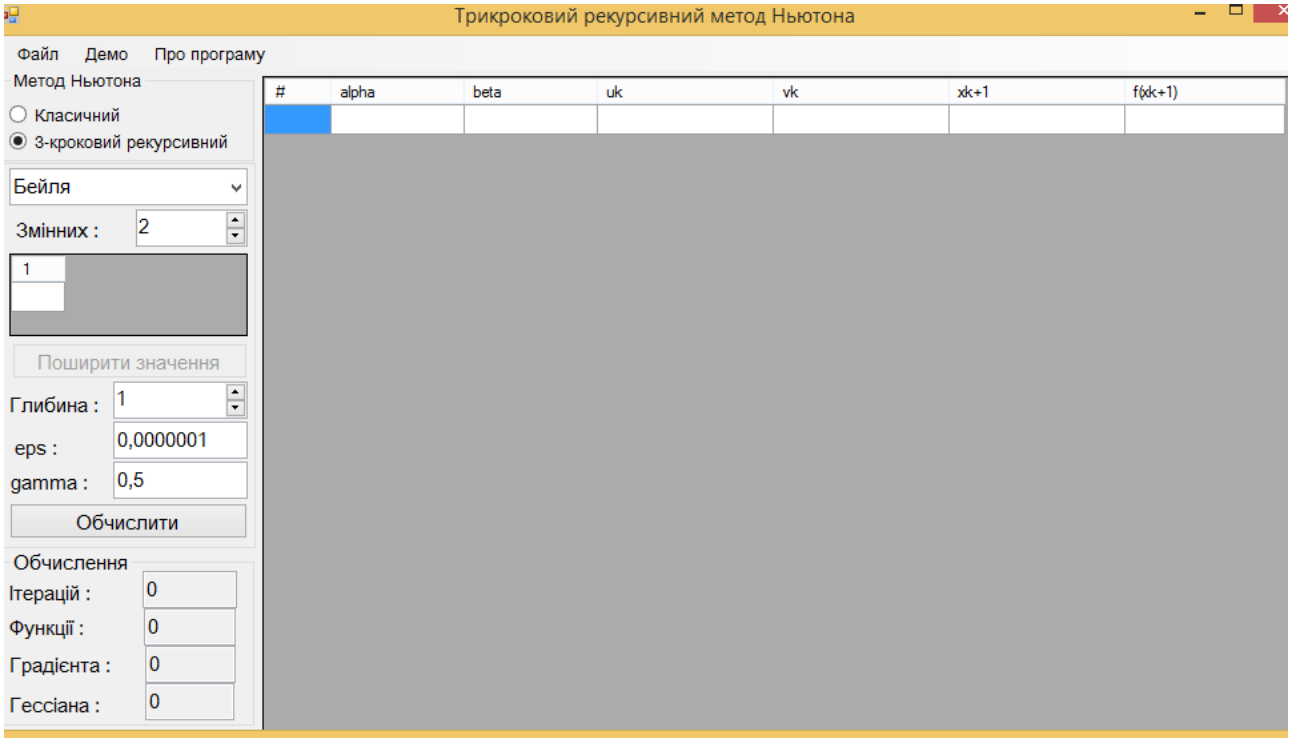

Рис.2.3.1. Загальний вигляд проекту

На формі розташовані такі кнопки:

Поширити значення (butnExtend) – доступна лише при вибраній парній кількості змінних, заповнює весь початковий вектор першими двома значеннями.

`Обчислити (buttonCalc)` – для запуску алгоритмів

Перемикачі:

`Класичний (radioClassical)` – для вибору класичного методу Ньютона

`3-кроковий рекурсивний (radioRecursive)` – для вибору трикрокового рекурсивного методу Ньютона.

`Випадаючий список (comboBFunc)` – для вибору функції

`Табляця 1 (dataG_x)` – для введення вектора початкового наближення

`Табляця 2 (dataGridRezults)` – для відображення результатів

`Поле 1 (textB_n)` – для вибору кількості змінних.

`Поле 2 (textB_depth)` – для введення глибини рекурсії

`Поле 3 (textB_eps)` – для введення точності обчислень

Поле 4 (textBGamma) – для введення коефіцієнта, на який будуть зменшуватися коефіцієнти в алгоритмах

Поле 5 (textB_IterNum) – для відображення кількості ітерацій

Поле 6 (textB_FCount) – для відображення кількості обчислень функції

Поле 7 (textB_FPCount) - для відображення кількості обчислень вектора-градієнта функції

Поле 8 (textB_FDPCount) - відображення кількості обчислень матриці Гессе

Меню (menuStrip1) – для виходу з програми, вибору демо прикладів та перегляду інформації про програму

2.4 Інструкція користувачеві

Після запуску проекту, користувач може запустити один з шести демо-прикладів. Для цього в меню Демо потрібно вибрати один з варіантів функцій. Після цього у формі буде автоматично задано вид функції, вибрано кількість змінних, заповнено дані про початкове наближення, вибрана глибина рекурсії, та задана точність обчислень.

Також користувач має можливість змінити ці дані, або ввести нові самостійно. В цьому випадку, функцію потрібно вибирати з випадаючого списку. Для тестування доступні такі функції: розширені функції Бейля та Розенброка, два види штрафних функцій, функція Вайта і Холста та Пауела.

Потрібно врахувати, що кнопка Поширити значення, яка заповнює значення початкового вектора наближення, працює лише для парної кількості змінних, для непарної кількості вона не доступна.

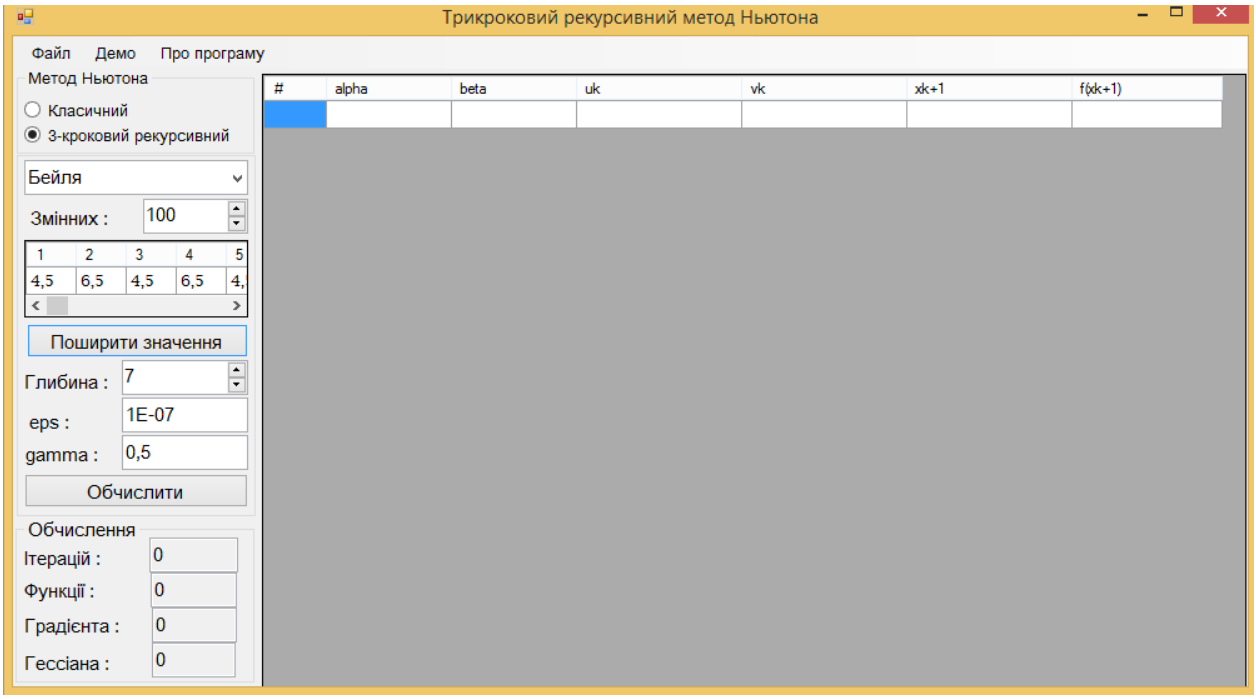

Рис.2.4.1. Запущений демо приклад

Далі потрібно натиснути кнопку `Обчислити.` Після натиснення цієї кнопки в таблиці результатів буде відображено проміжні дані роботи обраного користувачем методу. У відповідних полях буде виведено кількість ітерацій, обчислення функції, її похідної та матриці Гессе.

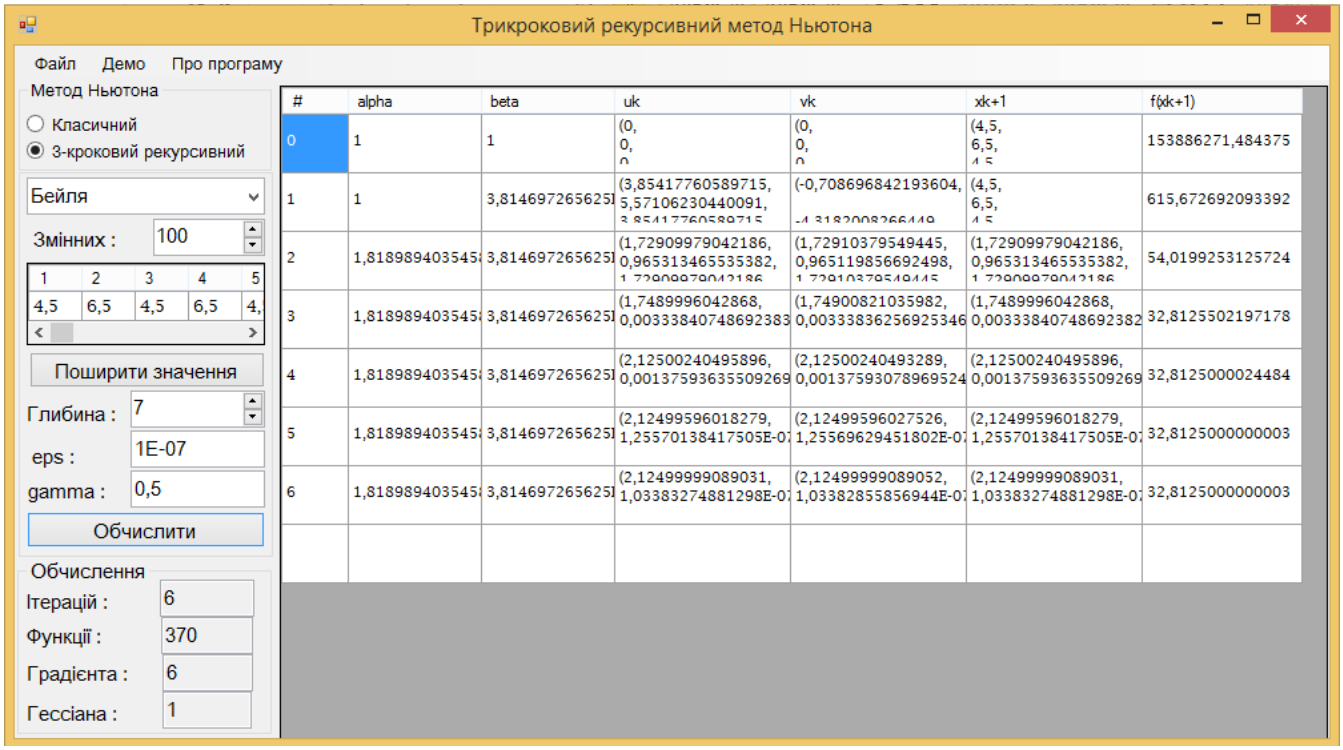

Рис.2.4.2. Відображення результатів

РОЗДІЛ 3
АПРОБАЦІЯ МЕТОДІВ

3.1 Розширена функція Бейля

У таблицях подані такі результати:

(1) - результати роботи класичного методу Ньютона;

p - глибина рекурсії у трикроковому рекурсивному методі Ньютона;

n – кількість змінних;

i -кількість ітерацій;

f –кількість обчислень функції;

f' - кількість обчислень вектора-градієнта;

f'' - кількість обчислень матриці Гессе;

k – загальна трудомісткість алгоритму.

Знайти мінімум розширеної функції Бейля:

$$f(x) = \sum_{i=1}^{n/2} \left[\left(1.5 - x_{2i-1}(1 - x_{2i}^3)\right)^2 + \left(2.25 - x_{2i-1}(1 - x_{2i}^2)\right)^2 + \left(2.625 - x_{2i-1}(1 - x_{2i}^3)\right)^2 \right];$$

з початковим наближенням

а) $x^0 = (4.5, 6.5, \dots, 4.5, 6.5),\ n = 2,4,\dots;$

Таблиця 3.1.1 Результати виконання програми:

n\p	(1)	1	2	3	4	5	6	7	8
2	i: 16 f: 0 f': 16 f'':16 k : 80	i: 6 f: 384 f': 6 f'': 6 k :414	i: 6 f: 370 f': 6 f'': 3 k :391	i: 6 f: 370 f': 6 f'': 2 k :388	i: 6 f: 370 f': 6 f'': 2 k :388	i: 6 f: 370 f': 6 f'': 2 k :388	i: 6 f: 370 f': 6 f'': 1 k :385	i: 6 f: 370 f': 6 f'': 1 k :385	i: 6 f: 370 f': 6 f'': 1 k :385

10	i: 16 f: 0 f': 16 f'':16 k:1040	i: 6 f: 384 f': 6 f'': 6 k:774	i: 6 f: 370 f': 6 f'': 3 k:595	i: 6 f: 370 f': 6 f'': 2 k:540	i: 6 f: 370 f': 6 f'': 2 k:540	i: 6 f: 370 f': 6 f'': 2 k:540	i: 6 f: 370 f': 6 f'': 1 k:485	i: 6 f: 370 f': 6 f'': 1 k:485	i: 6 f: 370 f': 6 f'': 1 k:485
50	i: 16 f: 0 f': 16 f'':16 k:21200	i: 6 f: 384 f': 6 f'': 6 k:8334	i: 6 f: 370 f': 6 f'': 3 k:4495	i: 6 f: 370 f': 6 f'': 2 k:3220	i: 6 f: 370 f': 6 f'': 2 k:3220	i: 6 f: 370 f': 6 f'': 2 k:3220	i: 6 f: 370 f': 6 f'': 1 k:1945	i: 6 f: 370 f': 6 f'': 1 k:1945	i: 6 f: 370 f': 6 f'': 1 k:1945
100	i: 16 f: 0 f': 16 f'':16 k:82400	i: 6 f: 384 f': 6 f'': 6 k:31284	i: 6 f: 370 f': 6 f'': 3 k:16120	i: 6 f: 370 f': 6 f'': 2 k:11070	i: 6 f: 370 f': 6 f'': 2 k:11070	i: 6 f: 370 f': 6 f'': 2 k:11070	i: 6 f: 370 f': 6 f'': 1 k:6020	i: 6 f: 370 f': 6 f'': 1 k:6020	i: 6 f: 370 f': 6 f'': 1 k:6020

б) $x^0 = (4.4, 5.8, \dots, 4.4, 5.8),\ n = 2,4, \dots;$

Таблиця 3.1.2 Результати виконання програми:

n\p	(1)	1	2	3	4	5	6	7	8
2	i: 16 f: 0 f': 16 f'': 16 k : 80	i: 5 f: 292 f': 5 f'': 5 k :317	i: 5 f: 283 f': 5 f'': 3 k :302	i: 5 f: 283 f': 5 f'': 2 k :299	i: 5 f: 283 f': 5 f'': 2 k :299	i: 5 f: 283 f': 5 f'': 1 k :296	i: 5 f: 283 f': 5 f'': 1 k :296	i: 5 f: 283 f': 5 f'': 1 k :296	i: 5 f: 283 f': 5 f'': 1 k :296
10	i: 16 f: 0 f': 16 f'': 16 k :1040	i: 5 f: 292 f': 5 f'': 5 k :617	i: 5 f: 283 f': 5 f'': 3 k :497	i: 5 f: 283 f': 5 f'': 2 k :443	i: 5 f: 283 f': 5 f'': 2 k :443	i: 5 f: 283 f': 5 f'': 1 k :388	i: 5 f: 283 f': 5 f'': 1 k :388	i: 5 f: 283 f': 5 f'': 1 k :388	i: 5 f: 283 f': 5 f'': 1 k :388

50	i: 16 f: 0 f': 16 f'': 16 k:21200	i: 5 f: 292 f': 5 f'': 5 k:6917	i: 5 f: 283 f': 5 f'': 3 k:4358	i: 5 f: 286 f': 5 f'': 2 k:3086	i: 5 f: 286 f': 5 f'': 2 k:3086	i: 5 f: 286 f': 5 f'': 1 k:1811	i: 5 f: 286 f': 5 f'': 1 k:1811	i: 5 f: 286 f': 5 f'': 1 k:1811	i: 5 f: 286 f': 5 f'': 1 k:1811
100	i: 16 f: 0 f': 16 f'': 16 k:82400	i: 5 f: 291 f': 5 f'': 5 k:26041	i: 5 f: 283 f': 5 f'': 3 k:15933	i: 5 f: 286 f': 5 f'': 2 k:10886	i: 5 f: 286 f': 5 f'': 2 k:10886	i: 5 f: 286 f': 5 f'': 1 k:5836	i: 5 f: 286 f': 5 f'': 1 k:5836	i: 5 f: 286 f': 5 f'': 1 k:5836	i: 5 f: 286 f': 5 f'': 1 k:5836

$$x_* = (2.125, 0, \dots, 2.125, 0);$$

$$f(x_*) = 0.6562\frac{n}{2};$$

3.2 Розширена функція Розенброка

Знайти мінімум розширеної функції Розенброка:

$$f(x) = \sum_{i=1}^{n/2} \left[100\left(x_{2i} - x_{2i-1}^2\right)^2 + (1 - x_{2i})^2\right]$$

з початковим наближенням

а) $x^0 = (0.9, -3.2, \dots, -3.2), n = 2,4, \dots;$

Таблиця 3.2.1 Результати виконання програми:

n\p	(1)	1	2	3	4	5	6	7	8
2	i: 5 f: 0 f': 5 f'': 5 k :25	i: 6 f: 94 f': 6 f'': 6 k :124	i: 6 f: 99 f': 6 f'': 3 k :120	i: 5 f: 85 f': 5 f'': 2 k :101	i: 6 f: 109 f': 6 f'': 2 k :127	i: 7 f: 138 f': 7 f'': 2 k : 158	i: 8 f: 162 f': 8 f'': 2 k : 184	i: 9 f: 191 f': 9 f'': 2 k :215	i: 10 f: 215 f': 10 f'': 2 k :241

10	i: 5 f: 0 f': 5 f'': 5 k :325	i: 7 f: 108 f': 7 f'': 7 k :563	i: 6 f: 99 f': 6 f'': 3 k :324	i: 5 f: 85 f': 5 f'': 2 k :245	i: 6 f: 109 f': 6 f'': 2 k :279	i: 7 f: 138 f': 7 f'': 2 k :318	i: 8 f: 162 f': 8 f'': 2 k :352	i: 9 f: 191 f': 9 f'': 2 k :391	i: 10 f: 215 f': 10 f'': 2 k :425
50	i: 5 f: 0 f': 5 f'': 5 k:6625	i: 7 f: 108 f': 7 f'': 7 k:9383	i: 6 f: 99 f': 6 f'': 3 k:4224	i: 6 f: 99 f': 6 f'': 2 k:2959	i: 7 f: 123 f': 7 f'': 2 k:3023	i: 7 f: 138 f': 7 f'': 2 k:3038	i: 9 f: 176 f': 9 f'': 2 k:3176	i: 9 f: 191 f': 9 f'': 2 k:3191	i: 10 f: 215 f': 10 f'': 2 k:3265
100	i: 5 f: 0 f': 5 f'': 5 k:25750	i: 7 f: 108 f': 7 f'': 7 k:36158	i: 6 f: 99 f': 6 f'': 3 k:15849	i: 6 f: 99 f': 6 f'': 2 k:10799	i: 7 f: 123 f': 7 f'': 2 k:10923	i: 7 f: 138 f': 7 f'': 2 k:10938	i: 9 f: 176 f': 9 f'': 2 k:11176	i: 9 f: 191 f': 9 f'': 2 k:11191	i: 11 f: 229 f': 11 f'': 2 k:11429

б) $x^0 = (1.05, 0.7, \ldots, 1.05, 0.7), n = 2,4, \ldots$

Таблиця 3.2.2 Результати виконання програми:

n\p	(1)	1	2	3	4	5	6	7	8
2	i: 5 f: 0 f': 5 f'': 5 k :25	i: 7 f: 100 f': 7 f'': 7 k :135	i: 6 f: 104 f': 6 f'': 3 k :125	i: 6 f: 114 f': 6 f'': 2 k :132	i: 6 f: 129 f': 6 f'': 2 k :147	i: 7 f: 135 f': 7 f'': 2 k :155	i: 7 f: 150 f': 7 f'': 2 k :170	i: 7 f: 150 f': 7 f'': 1 k :167	i: 7 f: 150 f': 7 f'': 1 k :167
10	i: 5 f: 0 f': 5 f'': 5	i: 7 f: 100 f': 7 f'': 7	i: 6 f: 104 f': 6 f'': 3	i: 6 f: 114 f': 6 f'': 2	i: 6 f: 129 f': 6 f'': 2	i: 7 f: 135 f': 7 f'': 2	i: 7 f: 150 f': 7 f'': 2	i: 7 f: 150 f': 7 f'': 1	i: 7 f: 150 f': 7 f'': 1

	k :325	k :555	k :329	k :284	k :299	k :315	k :330	k :275	k :275
50	i: 5 f: 0 f': 5 f'': 5 k:6625	i: 7 f: 100 f': 7 f'': 7 k:9375	i: 6 f: 104 f': 6 f'': 3 k:4229	i: 6 f: 114 f': 6 f'': 2 k:2964	i: 6 f: 129 f': 6 f'': 2 k:2979	i: 7 f: 135 f': 7 f'': 2 k:3035	i: 7 f: 150 f': 7 f'': 2 k:3050	i: 7 f: 150 f': 7 f'': 1 k:1775	i: 7 f: 150 f': 7 f'': 1 k:1775
100	i: 5 f: 0 f': 5 f'': 5 k:25750	i: 7 f: 100 f': 7 f'': 7 k:36150	i: 6 f: 104 f': 6 f'': 3 k:15854	i: 7 f: 128 f': 7 f'': 3 k:15978	i: 6 f: 129 f': 6 f'': 2 k:10829	i: 7 f: 135 f': 7 f'': 2 k:10935	i: 7 f: 150 f': 7 f'': 2 k:10950	i: 7 f: 150 f': 7 f'': 1 k:5900	i: 7 f: 150 f': 7 f'': 1 k:5900

$$x_* = (1,1,\dots,1);$$
$$f(x_*) = 0;$$

3.3 Штрафна функція

Знайти мінімум штрафної функції

$$f(x) = \sum_{i=1}^{n}(x_i - 1)^2 + 10^{-3}\left(\sum_{i=1}^{n} x_i^2 - 0{,}25\right)^2$$

з початковим наближенням

а) $x^0 = (10,10,\dots,10), n = 2,3,\dots;$

Таблиця 3.3.1 Результати виконання програми:

n\p	(1)	1	2	3	4	5	6	7	8
2	i: 12 f: 0 f': 12 f'': 12	i: 3 f: 147 f': 3 f'': 3	i: 3 f: 156 f': 3 f'': 2	i: 3 f: 148 f': 3 f'': 1	i: 3 f: 148 f': 3 f'': 1	i: 3 f: 148 f': 3 f'': 1	i: 3 f: 148 f': 3 f'': 1	i: 3 f: 148 f': 3 f'': 1	i: 3 f: 148 f': 3 f'': 1

	k :64	k: 158	k :168	k :157	k :157	k :157	k :157	k :157	k :157
10	i: 13 f: 0 f': 13 f'': 13 k :845	i: 4 f: 159 f': 4 f'': 4 k :419	i: 4 f: 170 f': 4 f'': 2 k :320	i: 5 f: 188 f': 5 f'': 2 k :348	i: 6 f: 174 f': 6 f'': 2 k :344	i: 6 f: 171 f': 6 f'': 2 k :341	i: 6 f: 166 f': 6 f'': 1 k :281	i: 6 f: 166 f': 6 f'': 1 k :281	i: 6 f: 166 f': 6 f'': 1 k :281
50	i: 14 f: 0 f': 14 f'': 14 k:18550	i: 4 f: 171 f': 4 f'': 4 k:5471	i: 3 f: 172 f': 3 f'': 2 k:2872	i: 3 f: 162 f': 3 f'': 1 k:1587	i: 3 f: 162 f': 3 f'': 1 k:1587	i: 3 f: 162 f': 3 f'': 1 k:1587	i: 3 f: 162 f': 3 f'': 1 k:1587	i: 3 f: 162 f': 3 f'': 1 k:1587	i: 3 f: 162 f': 3 f'': 1 k:1587
100	i: 15 f: 0 f': 15 f'': 15 k:77250	i: 4 f: 170 f': 4 f'': 4 k:20770	i: 5 f: 176 f': 5 f'': 3 k:15826	i: 6 f: 179 f': 6 f'': 2 k:10879	i: 7 f: 201 f': 7 f'': 2 k:11001	i: **7** f: 192 f': 7 f'': 2 k:10992	i: 6 f: 195 f': 7 f'': 2 k:10995	i: 7 f: 186 f': 7 f'': 1 k:5936	i: 7 f: 186 f': 7 f'': 1 k:5936

б) $x^0 = (4,4,\dots,4), n = 2,3,\dots$;

Таблиця 3.3.2 Результати виконання програми:

n\p	(1)	1	2	3	4	5	6	7	8
2	i: 9 f: 0 f': 9 f'': 9 k :45	i: 3 f: 127 f': 3 f'': 3 k :142	i: 3 f: 146 f': 3 f'': 2 k :158	i: 3 f: 139 f': 3 f'': 1 k :148	i: 3 f: 139 f': 3 f'': 1 k :148	i: 3 f: 139 f': 3 f'': 1 k :148	i: 3 f: 139 f': 3 f'': 1 k :148	i: 3 f: 139 f': 3 f'': 1 k :148	i: 3 f: 139 f': 3 f'': 1 k :148
10	i: 11 f: 0 f': 11	i: 2 f: 129 f': 2	i: 2 f: 123 f': 2	i: 2 f: 123 f': 2	i: 2 f: 123 f': 2	i: 2 f: 123 f': 2	i: 2 f: 123 f': 2	i: 2 f: 123 f': 2	i: 2 f: 123 f': 2

	f'': 11 k :715	f'': 2 k :259	f'': 1 k :198	f'': 1 k :198	f'': 1 k :198	f'': 1 k :198	f'': 1 k :198	f'': 1 k :198	f'': 1 k :198
50	i: 12 f: 0 f': 12 f'': 12 k:15900	i: 3 f: 137 f': 3 f'': 3 k:4112	i: 6 f: 159 f': 6 f'': 3 k:4282	i: 4 f: 142 f': 4 f'': 2 k:2892	i: 4 f: 137 f': 4 f'': 1 k:1812	i: 4 f: 137 f': 4 f'': 1 k:1812	i: 4 f: 137 f': 4 f'': 1 k:1812	i: 4 f: 137 f': 4 f'': 1 k:1812	i: 4 f: 137 f': 4 f'': 1 k:1812
100	i: 13 f: 0 f': 13 f'': 13 k :66950	i: 3 f: 134 f': 3 f'': 3 k:18284	i: 5 f: 137 f': 5 f'': 3 k:15787	i: 4 f: 146 f': 4 f'': 2 k:10646	i: 4 f: 141 f': 4 f'': 1 k:5591	i: 4 f: 141 f': 4 f'': 1 k:5591	i: 4 f: 141 f': 4 f'': 1 k:5591	i: 4 f: 141 f': 4 f'': 1 k:5591	i: 4 f: 141 f': 4 f'': 1 k:5591

3.4 Штрафна функція

Знайти мінімум штрафної функції

$$f(x) = 10^{-5}\sum_{i=1}^{n}(x_i - 1)^2 + 10^{-3}\left(\sum_{i=1}^{n} x_i^2 - 0{,}25\right)^2$$

з початковим наближенням

а) $x^0 = (5,5,\dots,5), n = 2,3,\dots;$

Таблиця 3.4.1 Результати виконання програми:

n\p	(1)	1	2	3	4	5	6	7	8
2	i: 12 f: 0 f': 12 f'': 12 k :60	i: 5 f: 177 f': 5 f'': 5 k :202	i: 5 f: 187 f': 5 f'': 3 k :206	i: 4 f: 179 f': 4 f'': 2 k :193	i: 4 f: 172 f': 4 f'': 1 k :183	i: 4 f: 172 f': 4 f'': 1 k :183	i: 4 f: 172 f': 4 f'': 1 k :183	i: 4 f: 172 f': 4 f'': 1 k :183	i: 4 f: 172 f': 4 f'': 1 k :183

10	i: 14 f: 0 f': 14 f'': 14 k :910	i: 3 f: 159 f': 3 f'': 3 k :354	i: 4 f: 204 f': 4 f'': 2 k :354	i: 4 f: 155 f': 4 f'': 2 k :305	i: 5 f: 191 f': 5 f'': 2 k :351	i: 5 f: 184 f': 5 f'': 1 k :289	i: 5 f: 184 f': 5 f'': 1 k :289	i: 5 f: 184 f': 5 f'': 1 k :289	i: 5 f: 184 f': 5 f'': 1 k :289
50	i: 16 f: 0 f': 16 f'': 16 k:21200	i: 3 f: 215 f': 3 f'': 3 k:4190	i: 3 f: 203 f': 3 f'': 2 k:2903	i: 3 f: 194 f': 3 f'': 1 k:1619	i: 3 f: 194 f': 3 f'': 1 k:1619	i: 3 f: 194 f': 3 f'': 1 k:1619	i: 3 f: 194 f': 3 f'': 1 k:1619	i: 3 f: 194 f': 3 f'': 1 k:1619	i: 3 f: 194 f': 3 f'': 1 k:1619
100	i: 17 f: 0 f': 17 f'': 17 k:87550	i: 4 f: 162 f': 4 f'': 4 k:20762	i: 3 f: 212 f': 3 f'': 2 k:10612	i: 3 f: 205 f': 3 f'': 1 k:5555	i: 3 f: 205 f': 3 f'': 1 k:5555	i: 3 f: 205 f': 3 f'': 1 k:5555	i: 3 f: 205 f': 3 f'': 1 k:5555	i: 3 f: 205 f': 3 f'': 1 k:5555	i: 3 f: 205 f': 3 f'': 1 k:5555

3.5 Розширена функція Вайта і Холста

Знайти мінімум функції

$$f(x) = \sum_{i=1}^{n/2} \left[100(x_{2i} - x_{2i-1}^3)^2 + (1 - x_{2i-1})^2\right]$$

з початковим наближенням

а) $x^0 = (1.25, -1, \dots, 1.25, -1), n = 2,4, \dots;$

Таблиця 3.5.1 Результати виконання програми:

n\p	(1)	1	2	3	4	5	6	7	8
2	i: 5 f: 0 f': 5 f'': 5 k :25	i: 8 f: 128 f': 8 f'': 8 k :168	i: 9 f: 157 f': 9 f'': 5 k :190	i: 9 f: 147 f': 9 f'': 3 k :174	i: 10 f: 181 f': 10 f'': 3 k :210	i: 12 f: 204 f': 12 f'': 3 k :237	i: 14 f: 267 f': 14 f'': 3 k :304	i: 16 f: 310 f': 16 f'': 3 k :351	i: 18 f: 363 f': 18 f'': 3 k :408
10	i: 5 f: 0 f': 5 f'': 5 k:325	i: 8 f: 128 f': 8 f'': 8 k :648	i: 9 f: 157 f': 9 f'': 5 k :522	i: 9 f: 147 f': 9 f'': 3 k :402	i: 11 f: 195 f': 11 f'': 3 k :470	i: 12 f: 204 f': 12 f'': 3 k :489	i: 14 f: 267 f': 14 f'': 3 k :572	i: 16 f: 310 f': 16 f'': 3 k :635	i: 18 f: 363 f': 18 f'': 3 k :708
50	i: 5 f: 0 f': 5 f'': 5 k:6625	i: 8 f: 128 f': 8 f'': 8 k:10728	i: 9 f: 157 f': 9 f'': 5 k:6982	i: 10 f: 161 f': 10 f'': 4 k:5761	i: 11 f: 195 f': 11 f'': 3 k:4570	i: 12 f: 204 f': 12 f'': 3 k:4629	i: 14 f: 267 f': 14 f'': 3 k:4792	i: 16 f: 310 f': 16 f'': 3 k:4935	i: 18 f: 363 f': 18 f'': 3 k:5088
100	i: 5 f: 0 f': 5 f'': 5 k:25750	i: 8 f: 128 f': 8 f'': 8 k:41328	i: 9 f: 157 f': 9 f'': 5 k:26307	i: 10 f: 161 f': 10 f'': 4 k:21361	i: 11 f: 195 f': 11 f'': 3 k:16445	i: 12 f: 204 f': 12 f'': 3 k:16554	i: 14 f: 267 f': 14 f'': 3 k:16817	i: 16 f: 310 f': 16 f'': 3 k:17060	i: 18 f: 363 f': 18 f'': 3 k:17313

3.6 Розширена функція Пауела

Знайти мінімум функції

$$f(x) = \sum_{i=1}^{n/4} [(x_{4i-3} + 10x_{4i-2})^2 + 5(x_{4i-1} + x_{4i})^2 + (x_{4i-2} - 2x_{4i-1})^4 + 10(x_{4i-3} - x_{4i})^4]$$

з початковим наближенням

а) $x^0 = (5,1,0,-2,\dots,5,1,0,-2), n = 4,8\dots$;

Таблиця 3.6.1 Результати виконання програми:

n\p	(1)	1	2	3	4	5	6	7	8
4	i: 43 f: 0 f': 43 f'': 43 k:602	i: 29 f: 235 f': 29 f'': 29 k :641	i: 30 f: 279 f': 30 f'': 15 k :549	i: 32 f: 714 f': 32 f'': 11 k :952	i: 38 f: 408 f': 38 f'': 10 k :660	i: 38 f: 436 f': 38 f'': 8 k :668	i: 47 f: 541 f': 47 f'': 8 k :809	i: 44 f: 763 f': 44 f'': 7 k :1009	i: 53 f: 647 f': 53 f'': 7 k :929
12	i: 44 f: 0 f': 44 f'':44 k :3960	i: 30 f: 241 f': 30 f'': 30 k :2941	i: 32 f: 297 f': 32 f'': 16 k :1929	i: 35 f: 786 f': 35 f'': 12 k:2142	i: 42 f: 449 f': 42 f'': 11 k:1811	i: 40 f: 464 f': 40 f'': 8 k :1568	i: 50 f: 575 f': 50 f'': 9 k :1877	i: 46 f: 803 f': 46 f'': 7 k:1901	i: 55 f: 675 f': 55 f'': 7 k:1881
52	i: 46 f: 0 f': 46 f'': 46 k:65780	i: 31 f: 247 f': 31 f'': 31 k:44577	i: 32 f: 297 f': 32 f'': 16 k:24009	i: 38 f: 863 f': 38 f'': 13 k:20753	i: 46 f: 487 f': 46 f'': 12 k:19415	i: 45 f: 521 f': 45 f'': 9 k:15263	i: 53 f: 607 f': 53 f'': 9 k:15765	i: 46 f: 806 f': 46 f'': 7 k:12844	i: 58 f: 709 f': 58 f'': 8 k:14749
100	i: 47 f: 0 f': 47 f'': 47 k :24250	i: 32 f: 265 f': 32 f'': 32 k:165065	i: 34 f: 315 f': 34 f'': 17 k :89565	i: 38 f: 863 f': 38 f'': 13 k:70313	i: 46 f: 487 f': 46 f'': 12 k:65687	i: 47 f: 541 f': 47 f'': 10 k:55741	i: 56 f: 641 f': 56 f'': 10 k:56741	i: 46 f: 806 f': 46 f'': 7 k:40756	i: 58 f: 709 f': 58 f'': 8 k:46909

ВИСНОВКИ

Метою цієї роботи було проведення дослідження модифікації методу Ньютона у трикроковий рекурсивний метод Ньютона. Підтвердження теоретичних результатів цього методу результатами отриманими на практиці за допомогою його програмної реалізації та порівняння ефективності цієї модифікації з класичним методом Ньютона.

У роботі розглядається класичний метод Ньютона, його трикрокова модифікація та рекурсивний аналог трикрокового методу Ньютона. Досліджена збіжність цих методів та їх ефективність.

Рекурсивний аналог трикрокового методу Ньютона є найефективнішим серед розглянутих і в сенсі кількості обчислень і в сенсі вимог до вхідних даних. У ньому ми рідше перераховуємо матрицю других похідних. Повторно використовуємо інформацію про значення вектора-градієнта, отриманого на попередньому кроці. Тим самим ми зменшуємо затрати на обчислення на кожній ітерації, та маємо змогу поправити значення отримане на першому кроці.

Проект реалізований на платформі .Net у середовищі розробки Visual Studio 2010 з використанням мови програмування C#. Інтерфейс є зручним і зрозумілим користувачеві.

Апробація виконана на штрафних функціях, функціях Бейля, Розенброка, Вайта і Холста та Пауела. Протестувавши програму, було зроблено висновок про те, що трикроковий рекурсивний метод Ньютона переважає за своїми обчислювальними властивостями класичний метод Ньютона.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Бартіш В. Рекурсивний аналог трикрокового методу Ньютона // Вісник Львів.ун-ту, Сер. прикл. матем. та інформ., 2013. Вип.19.С.3-9

2. Бартіш М. Я. Методи оптимізації. Теорія і алгоритми: Навчальний посібник // Львів: Видавничий центр ЛНУ імені Івана Франка 2006. – 223с.

3. Бартіш М. Я. Модифікації рекурсивного методу Ньютона / М. Бартіш, Н. Огородник // Вісник Львів.ун-ту, Сер. прикл. матем. та інформ., 2010. Вип.16.С.3-9

4. Бартіш М. Я. Про один трикроковий метод розв'язування задач мінімізації / М. Бартіш, Н. Огородник // Вісник Львів.ун-ту, Сер. прикл. матем. та інформ., 2009. Вип.15.С.20-25

5. Бартіш М. Я. Трикрокові методи розв'язування задач безумовної мінімізації / М. Бартіш, О. Ковальчук, Н. Огородник // Вісник Львів.ун-ту, Сер. прикл. матем. та інформ., 2007. Вип.13.С.3-10

6. Васильев Ф. П. Методы оптимизации / Ф. П. Васильев. – М.: Факториал Пресс, 2002. – С. 824.

7. Габасов Р. Методы оптимизации / Р. Габасов, Ф. М. Кириллова, В. В. Альсевич, А. И. Калинин, В. В. Крахотко, Н. С. Павленок // Четыре четверти, 2011. – 474 с.

8. Пшеничный Б. Н. Численные методы в єкстремальных задачах / Б. Н. Пшеничный, Ю. М. Данилин. – М.: Наука, 1975. – 320 с.

9. Троелсен Є. Язык програмирования C# 2010 и платформа .Net 4.0 5 – е изд. // М.: ООО „И.Д. Вильямс", 2011. – 1392 с.

10. Химмельблау Д. Прикладное нелинейное программирование // М.: Мир, 1975.

11. Шилдт Г. C# 4.0: Полное руководство // Вильямс, 2011. - 1056 с.