УДК 519.688

ОЦІНКА ЕФЕКТИВНОСТІ РОЗПАРАЛЕЛЕННЯ ОПТИМІЗАЦІЙНОЇ ПРОЦЕДУРИ ПОБУДОВИ ДИСКРЕТНИХ ДИНАМІЧНИХ МОДЕЛЕЙ

І.П. Струбицька

Тернопільський національний економічний університет

iryna.str@gmail.com

У статті проведено оцінку ефективності розпаралелення оптимізаційної процедури побудови дискретних динамічних моделей. Проведено порівняння часу виконання паралельної програми в залежності від порядку моделі, кількості дискрет, кількості потоків та блоків графічного процесора.

Ключові слова: дискретні динамічні моделі, метод оптимізації, паралельні обчислення, розпаралелення, графічні процесори.

The evaluation of efficiency of parallelization of optimization procedure of constructing of discrete dynamical models was conducted in this paper. Comparison of the execution time of parallel program depending on the order of model, the number of discretes, the number of threads and blocks of GPU was conducted.

Keywords: discrete dynamic models, optimization methods, parallel computing, parallelization, GPU.

В статье проведена оценка эффективности распараллеливания оптимизационной процедуры построения дискретных динамических моделей. Проведено сравнение времени выполнения параллельной программы в зависимости от порядка модели, количества дискрет, количества потоков и блоков графического процессора.

Ключевые слова: дискретные динамические модели, метод оптимизации, параллельные вычисления, распараллеливания, графические процессоры.

Вступ

Методи побудови дискретних динамічних моделей різноманітних систем є достатньо розроблені та широко використовуються. Задачі параметричної ідентифікації дискретних моделей описані в працях Л. Люнга [1], Л. Заде, Ч. Дезоера [2], В. Стрейца [3], В. М. Кунцевича [4]. З точки зору комп'ютерного моделювання найбільш перспективним є метод моделювання на основі дискретних рівнянь стану [5-8]. З математичної точки зору цей підхід є найбільш формалізованим і має практичне застосування в різних галузях.

У працях П. Г. Стахіва та Ю. Я. Козака [9-10] побудова динамічних моделей електричних та електронних кіл здійснюється з використанням оптимізаційного підходу. Такий підхід дає можливість зробити універсальною побудову моделей. Однак, така універсалізація призводить до появи складних оптимізаційних задач, які важко розв’язати за прийнятний час навіть з використанням сучасних засобів обчислювальної техніки.

Актуальним є створення таких методів побудови моделей, які б піддавалися реалізації на доступних засобах обчислювальної техніки і водночас забезпечували необхідну швидкодію.

Тому існує потреба в розробці достатньо універсальних алгоритмів побудови дискретних динамічних моделей з використанням розпаралелення, за допомогою яких можна буде ефективно будувати моделі екологічних, електроенергетичних та інших складних систем.

1. Метод розпаралелення оптимізаційної процедури побудови дискретних динамічних моделей

Розглянемо узагальнену математичну модель у формі рівнянь стану (1):

$$\begin{cases} \vec{x}^{(k+1)} = F\vec{x}^{(k)} + G\vec{v}^{(k)} + \vec{\Phi}(\vec{x}^{(k)}, \vec{v}^{(k)}), \\ \vec{y}^{(k+1)} = C\vec{x}^{(k+1)} + D\vec{v}^{(k+1)}, \end{cases} \quad (1)$$

де $\vec{x}^{(k)}$ – вектор змінних стану; $\vec{v}^{(k)}$ – вектор вхідних значень; $\vec{y}^{(k+1)}$ – вектор вихідних значень; F, G, C, D – матриці, з невідомими коефіцієнтами, які необхідно знайти при побудові моделі; $\vec{\Phi}$ – деяка нелінійна вектор-функція багатьох змінних, форму і коефіцієнти якої також потрібно знайти.

Така форма моделі (1) характеризується деяким вектором невідомих параметрів $\vec{\lambda}$. Для даної моделі цей вектор складається з елементів матриць F, G, C, D і коефіцієнтів вектор-функції $\vec{\Phi}(\vec{x}^{(k)}, \vec{y}^{(k)})$.

Введемо деякий критерій для точності моделі $Q(\vec{\lambda}) > 0$, який позначає відхилення поведінки моделі від поведінки модельованого об’єкту для відомих проміжків часу. Функція $Q(\vec{\lambda})$, яка називається функцією мети, розраховується як середньоквадратична похибка значень моделі від значень модельованої системи:

$$Q(\vec{\lambda}) = \sum (\vec{y} - \vec{y}^{*}(\vec{\lambda}))^2,$$

де $\vec{y}$ – відомі характеристики модельованого об'єкта, $\vec{y}^{*}(\vec{\lambda})$ – перехідні характеристики розраховані за допомогою моделі.

Тому побудову моделі можна звести до знаходження значень вектора $\vec{\lambda}$, при яких функція мети буде мінімальною. Ця задача розв’язується за допомогою алгоритму оптимізації.

Задача знаходження мінімальної точки нелінійної функції $Q(\vec{\lambda})$ багатьох змінних є складним завданням. При побудові дискретних динамічних моделей функція мети має чітко виражений яровий характер з великою кількістю локальних мінімумів. Для розв’язку таких задач найкращими характеристиками володіє метод напрямного конуса Растригіна [11]. За допомогою цього підходу

можна провести цілеспрямований перебір локальних мінімумів, що прискорює знаходження глобального мінімуму цільової функції. Але обчислювальна складність такої задачі буде досить велика. Також для побудови якісної моделі використовується значна кількість вхідних даних. Отже, значними будуть і затрати машинного часу на реалізацію оптимізаційних процедур.

У праці [12] запропоновано розпаралелення даної задачі з використанням SIMD-архітектури, яка дозволяє виконати один і той самий потік команд для багатьох потоків даних.

З практично доступних сьогодні пристроїв з SIMD-архітектурою та за критерієм ціна/продуктивність є GPU (Graphics Proccesing Unit) [13-14]. Тому пропонований алгоритм адаптовано саме для цих акселераторів обчислень.

З програмної точки зору реалізація даного розпаралелення не вимагає великих затрат завдяки програмній архітектурі графічного процесора. Для всіх потоків пересилається ідентичний програмний код. Вхідними даними для кожного потоку виступають параметри моделі. Вони для всіх потоків однакові. На виході кожного потоку отримується значення функції мети. Для кожного потоку це значення своє. Усі вихідні дані зберігаються у спільній пам'яті потоків для подальшого обчислення мінімуму. Схематично порівняння процесів виконання обчислювального алгоритму на центральному та графічному процесорах представлено на рис. 1.

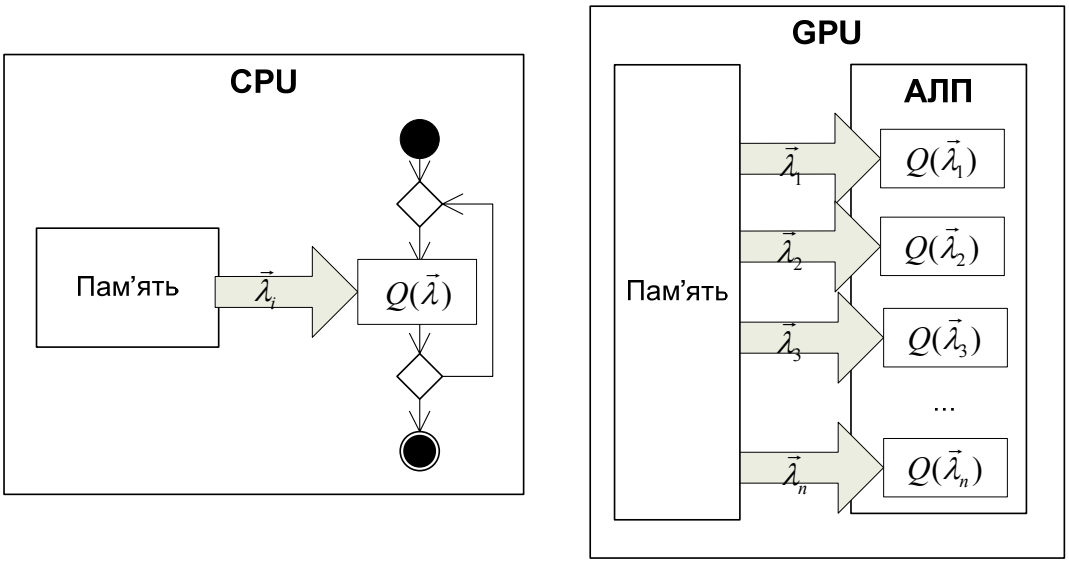

Рис. 1. Схема обчислювального алгоритму на CPU та GPU

Для програмної реалізації розпаралеленого алгоритму використано технологію CUDA (Compute Unified Device Architecture). На даний час обчислення на графічних процесорах з технологією CUDA — це інноваційне поєднання обчислювальних особливостей нового покоління графічних процесорів NVIDIA, що обробляють відразу тисячі потоків з високим рівнем інформаційного завантаження, які доступні через стандартну мову програмування C [13].

2. Дослідження ефективності розпаралелення оптимізаційної процедури побудови дискретних динамічних моделей

Для того, що оцінити ефективність розпаралелення оптимізаційної процедури побудови дискретних динамічних моделей проведено власне тестування розробленого програмного забезпечення. Оскільки обчислення функції мети в різних точках проводилось у різних потоках графічного процесора, то досліджено саме цю частину програми. Послідовна частина коду і так виконується на центральному процесорі, тобто час буде практично незмінним.

Для проведення тестування швидкодії паралельної програми розрахунку функції мети побудуємо автономну дискретну динамічну модель третього порядку, яка матиме наступний вигляд:

$$\begin{pmatrix} x_1^{k+1} \\ x_2^{k+1} \\ x_3^{k+1} \end{pmatrix} = \begin{pmatrix} F_{11} & F_{12} & F_{13} \\ F_{21} & F_{22} & F_{23} \\ F_{31} & F_{32} & F_{33} \end{pmatrix} \begin{pmatrix} x_1^{k} \\ x_2^{k} \\ x_3^{k} \end{pmatrix},$$

$$\begin{pmatrix} y_1^{k+1} \\ y_1^{k+1} \\ y_1^{k+1} \end{pmatrix} = \begin{pmatrix} C_1 & C_2 & C_3 \end{pmatrix} \begin{pmatrix} x_1^{k+1} \\ x_2^{k+1} \\ x_3^{k+1} \end{pmatrix},$$

де $k = 1,...,52$.

Входом моделі є середні значення потижневих викидів SO_2 (двоокису сульфуру) в атмосферу в м. Живєц (Польща) за 2011 р., а виходом – середні значення потижневих викидів в 2013 р. [15].

Функція мети матиме наступний вигляд:

$$Q(\vec{\lambda}_i) = \sum_i \left| \begin{pmatrix} y_{11} \\ y_{21} \\ y_{31} \end{pmatrix} - \begin{pmatrix} y_{11}^* \\ y_{21}^* \\ y_{31}^* \end{pmatrix} \right|^2 .$$

Проведемо тестування програми обчислення функції мети. Для цього використано наступне апаратне та системне забезпечення:

- графічний процесор NVIDIA GeForce GTS250 (16 мультипроцесорів по 8 ядер);
- оперативна пам’ять 1024 Мб;
- центральний процесор Core2Duo E8400, 3 ГГц;

– материнська плата ASRock G41M-S3;
– операційна система Windows XP.

Для цієї моделі дослідимо час виконання паралельної програми розрахунку функції мети на 128 ядрах графічного процесора. При 100 запусках середній час паралельного обчислення функції мети – 2,74 мс.

Дослідимо зміну часу виконання програми в залежності від зміни порядку моделі, для якої проводиться оптимізація (рис. 2). Очевидно, що час виконання паралельного розрахунку функцій мети буде поступово збільшуватись із збільшенням порядку побудованої моделі.

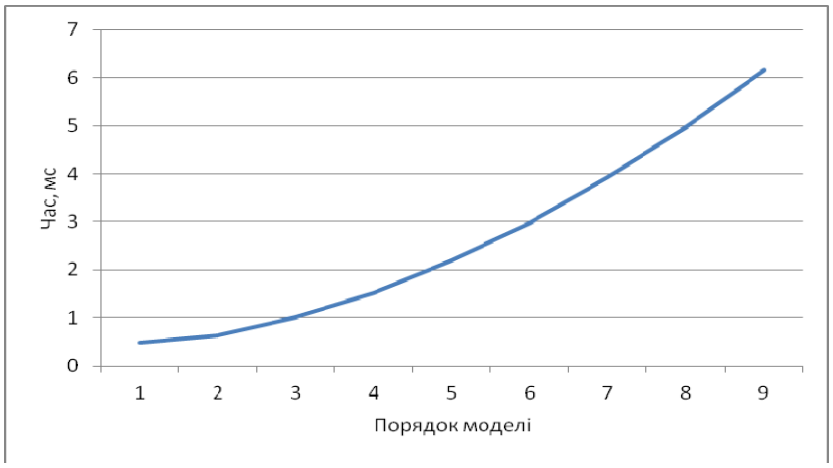

Рис. 2. Залежність часу виконання паралельної програми від порядку моделі

При збільшенні кількості дискет час виконання паралельного обчислення функції мети також буде поступово зростати (рис. 3).

Рис. 3. Залежність часу виконання паралельної програми від кількості дискрет моделі

Цікаво також порівняти залежність часу, що необхідний для всіх розрахунків функцій мети, від кількості обчислень цільової функції. Тобто, при різній кількості потоків на графічному процесорі. Очевидно, що для

послідовного обчислення така залежність має лінійний характер. Для паралельних обчислень ця залежність показана на рис. 4.

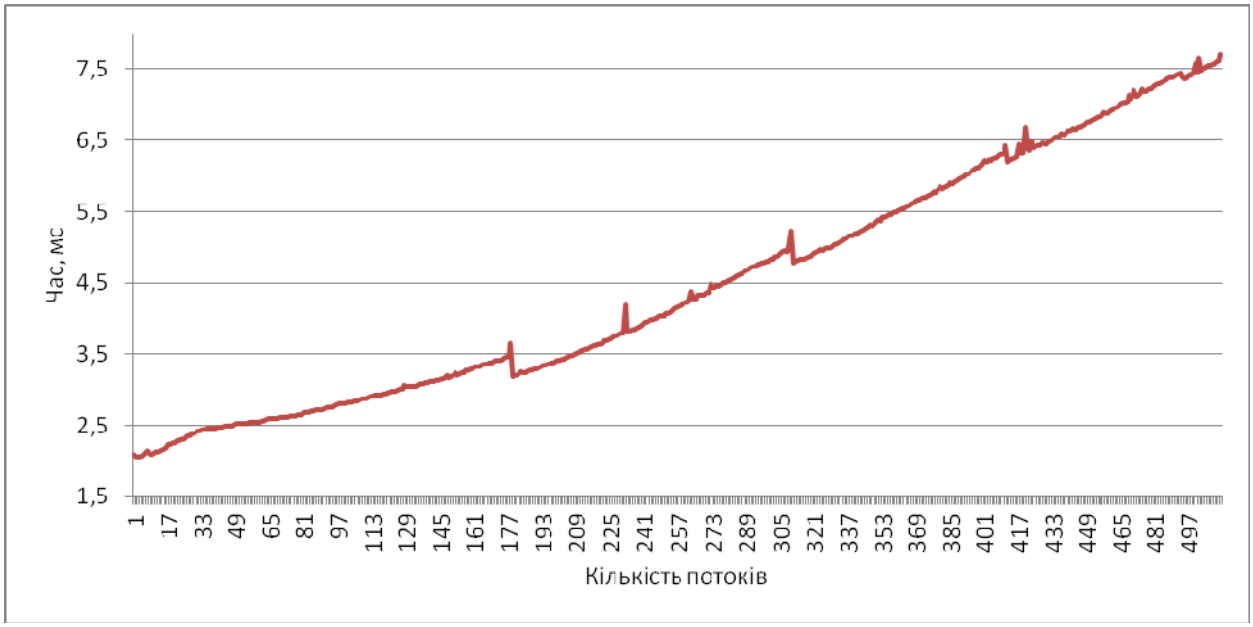

Рис. 4. Залежність часу виконання паралельної програми від кількості потоків

При цьому випадкових значень чи похибок при замірі часу виконання паралельної програми не може бути, оскільки всі дані усереднювались по 100 значеннях.

Як видно з рис. 4, така залежність має періодичний характер. Оскільки час програми обчислювався разом із пересилкою даних, то на кожному циклі є постійні затримки виконання оптимізаційного алгоритму. Якщо відкинути ці затримки, то побачимо, що періодичність повторюється через 128 потоків. Тому доцільно використовувати розпаралелення при кількості потоків кратній кількості ядер графічного процесора. У нашому випадку це 128 ядер. У такому разі розпаралелення буде найефективнішим.

Для кращого уявлення побудуємо 3D-графік залежності часу виконання паралельної програми від кількості потоків і кількості блоків графічного процесора (рис. 5).

Запропонований підхід паралельних обчислень може використовуватись з різними оптимізаційними алгоритмами. Тому до уваги не береться час розрахунку, який необхідний для самого алгоритму. На практиці час обчислень самого алгоритму є меншим ніж час, який необхідний для розрахунку функції мети.

Очевидно, що продуктивність центрального процесора значно вища, ніж продуктивність одного ядра графічного процесора. Проте, паралельна реалізація буде ефективнішою при одночасному обчисленні функції мети для багатьох наборів коефіцієнтів моделі [16]. Тобто чим більше проводиться

обчислень цільової функції, тим ефективнішим буде розпаралелення процесу побудови дискретних динамічних моделей.

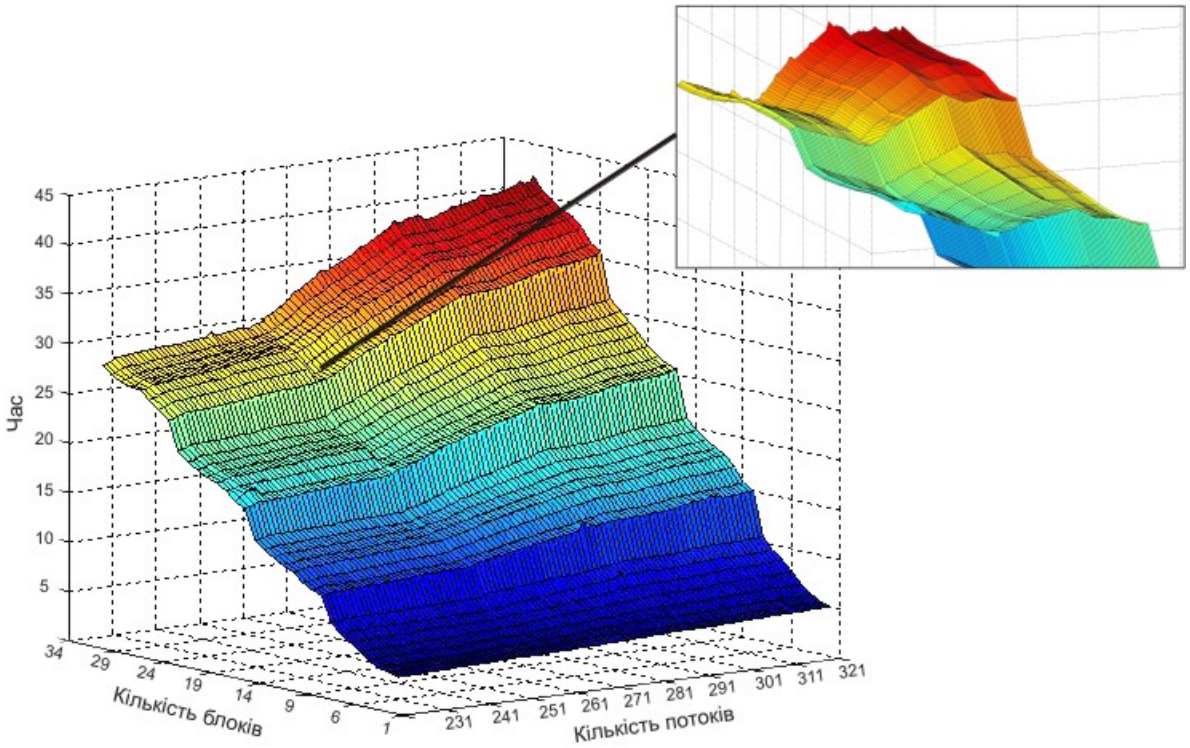

Рис. 5. Залежність часу виконання паралельної програми від кількості потоків і кількості блоків

Для оцінювання ефективності запропонованого підходу розпаралелення порівняємо час, що необхідний для розрахунку функції мети з використанням розпаралелення та без нього. Обчислимо прискорення паралельної програми за такою формулою:

$$S = \frac{t_{посл}}{t_{пар}},$$

де S – прискорення; $t_{посл}$ – час виконання послідовної програми; $t_{пар}$ – час виконання паралельної програми.

Залежність такого прискорення від кількості паралельних обчислень цільової функції на графічному процесорі показана на рис. 6.

Як бачимо, для великої кількості паралельних потоків обчислення функції мети досягло прискорення в 5 разів у порівнянні з центральним процесором. Хоча для тестування використовували один із найпростіших графічних процесорів із підтримкою технології CUDA.

Додаткові обмеження накладаються для обчислювальної точності. Сучасні графічні процесори можуть виконувати обчислення з подвійною

точністю, але в цьому випадку швидкість розрахунків значно знижується. Проте GPU швидко прогресує, так що ця проблема буде вирішена в найближчому майбутньому.

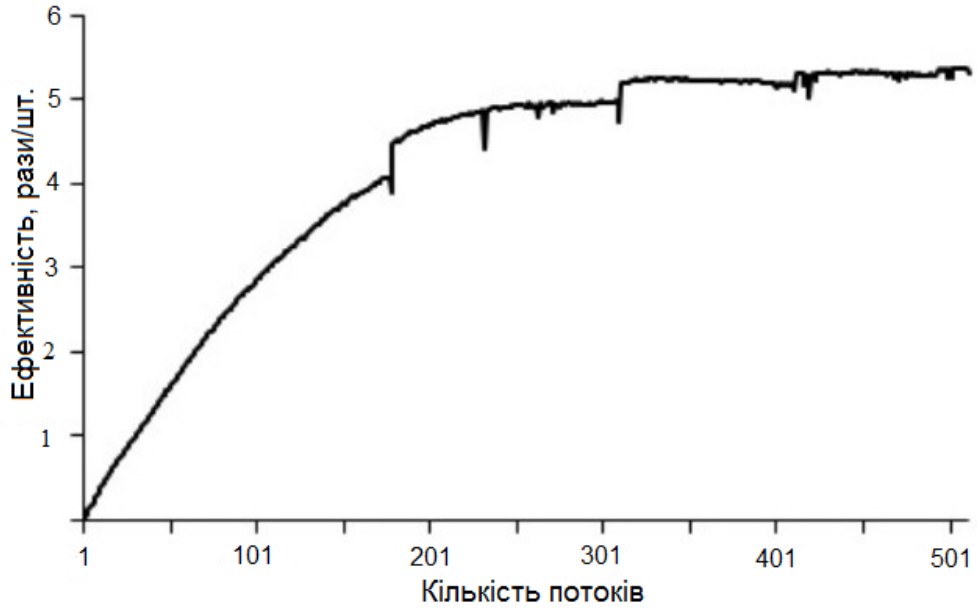

Рис. 6. Ефективність розпаралелення з використанням GPU порівняно з центральним процесором

Висновки

У результаті цього дослідження проведено оцінку ефективності розпаралелення оптимізаційної процедури побудови дискретних динамічних моделей. Проведено порівняння часу виконання паралельної програми в залежності від порядку моделі, кількості дискрет, кількості потоків та блоків графічного процесора. Також досліджено прискорення виконання паралельної програми на графічному процесорі порівняно з послідовною на центральному процесорі. Обчислене прискорення показало ефективність розпаралелення обчислювального процесу побудови дискретних динамічних моделей.

Література

1. Люнг Л. Идентификация систем. Теория для пользователя / Л. Люнг; [пер. с англ. под ред. Я.З. Цыпкина]. – М.: Наука. Гл. ред. физ.-мат. лит, 1991. – 432 с.

2. Заде Л. Теория линейных систем. Метод пространства состояний / Л. Заде, Ч. Дезоер – М.: Наука, 1970. – 704 с.

3. Стрейц В. Метод пространства состояний в теории дискретних линейных систем управления/ В. Стрейц [Пер. с англ. под ред. Я.З. Цыпкина] – М.: наука. Главная редакция физико-математической литературы, 1985. – 296 с.

4. Кунцевич Р.М. О редуцированных моделях дискретных динамических объектов и их гарантированных оценках в задачах управления / Р.М. Кунцевич // Проблемы управления и информатики. – 2001. – № 1. – С. 42-50.

5. Стахив П.Г. Анализ динамических режимов в электронных схемах с многополюсниками / П.Г. Стахив – Львов: Высш. школа, 1988. – 154 с.

6. Hinamoto T. Approximation of polynomial state-affine discrete-time systems / T. Hinamoto, S. Mackava – IEEE Trans. Circ. and Syst. – 1984. – Vol. 33, № 8. – P. 713-721.

7. Isidori A. Direct Construction of minimal bilinear realization from nonlinear input-output maps / A. Isidori – IEEE. – 1973. – vol. AC-18, № 6. – P. 626-631.

8. Мельник Б.К. Алгоритм побудови білінійних макромоделей багатополюсних елементів електронних схем / Б.К. Мельник, П.Г. Стахів // Теоретична електротехніка. – 1995. – № 52 – С. 94-98.

9. Стахів П.Г, Побудова макромоделей електромеханічних компонент із використанням оптимізації / П.Г. Стахів, Ю.Я. Козак // Технічна електродинаміка. – 2001. – №4. – С. 33-36.

10. Стахів П.Г. Побудова математичної макромоделі електромеханічного перетворювача вентильного двигуна з використанням оптимізації / П.Г. Стахів, Ю.Я. Козак, В.Г. Гайдук // Електроенергетичні та електромеханічні системи, вісник національного університету «Львівська політехніка». – 2001. – №418. – С. 159-164.

11. Растригин Л.А. Адаптация сложных систем / Л.А. Растригин - Рига: Зинатне, 1981. – 375 с.

12. Козак Ю.Я. Розпаралелення алгоритму оптимізації параметрів дискретних динамічних моделей на масивно-паралельних процесорах / Ю.Я., П.Г. Стахів, І.П. Струбицька // Відбір і обробка інформації. - 2010. – Вип. 32 (108). – С. 126-130.

13. Боресков А. В. Параллельные вычисления на GPU. Архитектура и программная модель CUDA/ А. В. Боресков, А. А. Харламов и др.. – М.: Издательство Московского университет, 2012. – 336 с.

14. Струбицька І.П. Аналіз сучасних графічних процесорів з підтримкою CUDA / І.П. Струбицька, А.І. Цигипало // Матеріали III Всеукраїнської школи-семінару молодих вчених і студентів «Сучасні комп'ютерні інформаційні технології". – Тернопіль: ТНЕУ, 2013. – С. 162-163.

15. Śląski Monitoring Powietrza [Electronic Resource]. – Mode of access: http://stacje.katowice.pios.gov.pl/iseo/.

16. Stakhiv P. Parallelization of calculations using GPU in optimization approach for macromodels construction / Petro Stakhiv, Iryna Strubytska, Yuriy Kozak // Przegląd elektrotechniczny. – 2012. – № 3a. – P. 7-9.