УДК 004.89

О. В. Олещук, О. Є. Попель, М. Б. Копитчук
Одеський національний політехнічний університет

МЕТОД ВИМІРЮВАННЯ ЕНЕРГЕТИЧНОЇ ЕФЕКТИВНОСТІ ОБЧИСЛЕНЬ НА ГРАФІЧНОМУ ЯДРІ

Запропоновано метод вимірювання енергетичної ефективності обчислень на CPU і GPU, який не потребує спеціалізованого вимірювального обладнання. За результатами проведених експериментів порівняно ефективність обчислень на CPU і на GPU з погляду енерговитрат.

Ключові слова: зелені технології, паралельні обчислення, GPU загального призначення, технологія CUDA.

THE METHOD OF MEASURING OF THE ENERGY EFFECTIVENESS OF CALCULATIONS IN GRAPHICAL CORE

This paper proposes a method for measuring the energy efficiency of computing on CPU and GPU, which does not require specialized instrumentation. The results of the experiments carried out to the comparison of the calculation effectiveness on the CPU and GPU in terms of energy consumption.

Key words: green technology, parallel computing, general purpose GPU, technology CUDA.

Постановка проблеми

Однією з задач, що десятиріччями не втрачає актуальності, є підвищення продуктивності обчислень на ЕОМ. Але в останні роки стало очевидним, що через технологічні обмеження еволюція центральних процесорів (CPU) за збереження наявної архітектури не дає бажаного приросту продуктивності при вирішенні реальних завдань. І водночас за продуктивністю графічних процесорних пристроїв (GPU) завдяки їхнім архітектурним особливостям експоненціально зростає [1]. З цієї причини GPU починають використовуватися як обчислювач для розв’язання задач загального призначення, що мають високий природний паралелізм. Як програмну платформу при цьому використовують такі технології, як OpenCL і CUDA [1], розроблені спеціально для неграфічних обчислень на графічному ядрі.

Іншим актуальним завданням є енергозбереження. І виникає питання, як саме перенесення обчислень на GPU відповідатиме вимогам щодо енергетичної ефективності. Розробники апаратури можуть надати лише орієнтовні відповіді, вказавши максимальну потужність, що споживається обчислювальним пристроєм. А для того, щоб визначити, яка частка енергії витрачається саме на неграфічні обчислення і як особливості алгоритмічної реалізації впливають на енергетичну ефективність, потрібно експериментально дослідити цю проблему.

Аналіз літературних джерел

Основною метою перенесення обчислень з CPU на GPU переважно є отримання істотного приросту продуктивності. Але, як виявилося, архітектура GPU має ще одну важливу властивість – високу енергетичну ефективність.

У літературі [1, 3, 4] є багато матеріалів, що стосуються питань продуктивності технології CUDA, але питанням енергетичної ефективності CUDA і можливостям її вимірювання досі не приділяли достатньо уваги. Якщо говорити про особливості архітектури, які забезпечують значну продуктивність, то насамперед слід зазначити високий паралелізм GPU із можливістю швидкого перемикання потоків. При цьому використовується система команд SIMD [1], яка дозволяє виконувати одну і ту ж саму операцію на багатьох АЛУ одночасно.

З цих архітектурних особливостей випливають і чинники, що зумовлюють енергетичну ефективність. Такими чинниками є:

- велика частина площі кристала GPU відведена під АЛУ, що займаються безпосередньо обробкою даних, а не завданнями управління послідовністю команд, причому власне АЛУ є доволі простими, оскільки в них реалізовано спрощену систему команд, що складається з найхарактерніших операцій;
- за рахунок масового паралелізму обчислень з'являється можливість зниження частоти і, як наслідок, енергоспоживання із збереженням високої продуктивності.

Мета

Метою статті є визначення методу для емпіричного визначення ефективності обчислень з погляду витрат електроенергії. Запропонований метод дає змогу без використання спеціалізованих вимірювальних засобів емпіричним шляхом визначити енергетичну ефективність готового програмного рішення, що ґрунтується на технології CUDA, оцінити вплив різних параметрів на продуктивність і енергетичну ефективність і порівняти їх з аналогічною реалізацією, що використовує як обчислювач CPU.

Однією з задач, що розглядаються, є визначення того, як деталі реалізації алгоритму обчислень впливають на завантаженість CPU, GPU і каналу передавання даних між цими пристроями.

Загальний опис методу визначення енергетичної ефективності

Щоб оцінити енергетичну ефективність програмно-апаратних рішень, що використовують як обчислювач GPU, а як програмний інтерфейс – засоби, що надає технологія CUDA, було проведено серію експериментів.

Як апаратну платформу використано ноутбук Fujitsu Lifebook AH532/G21 на базі центрального процесора Intel Pentium B960 з оперативною пам'яттю DDR3 обсягом 4 Гб. Заявлена ємність акумулятора – 4400 мАг.

Оскільки безпосередні вимірювання і фіксація потужності, споживаної під час експерименту, вимагали б наявності вимірювальної апаратури, то було прийнято рішення оцінювати енергоспоживання непрямими методами. У цьому випадку було зроблено припущення, що зарядна ємність акумулятора істотно не змінюється протягом усього експерименту. Тоді у режимі роботи від акумулятора час, протягом якого акумулятор перейде зі стану повністю зарядженого у стан максимально розрядженого, буде прямо пропорційним потужності, спожитої протягом цього інтервалу часу.

Для грубого оцінювання абсолютних значень енергоспоживання можна використовувати значення зарядної ємності акумулятора, заявлене виробником. Однак зрозуміло, що реальне значення ємності може бути іншим і відрізнятися, як правило, у бік менших значень. Але в цьому випадку це не принципово, оскільки інтерес представляє не абсолютне енергоспоживання, а його зміна при задіянні обчислювальних ресурсів, що надаються технологією CUDA.

З програмного погляду сучасні операційні системи надають набір API-функцій для отримання інформації про електроживлення. Серед них для цього експерименту найпридатніші такі показники, як поточний режим живлення і актуальний рівень заряду акумулятора. Перший показник дає змогу відстежити моменти під’єднання і від’єднання живлення від електричної мережі і за необхідності автоматизувати експеримент, прив'язавши до моментів перемикання режимів живлення запуск і зупинку обчислень, що досліджуються. Другий показник дає змогу не тільки фіксувати загальний час роботи в автономному режимі, але й точніше оцінювати процес споживання енергії, періодично зіставляючи момент часу з зарядом акумулятора, що залишився.

За таким підходом до вимірювання енергоспоживання необхідно забезпечити рівномірне, заздалегідь відоме навантаження на процесорні елементи. Для цього слід максимально відключити периферійні апаратні засоби, а також зупинити некритичні системні служби, антивірусні програми та інше сервісне програмне забезпечення. Центральний процесор та/або графічне ядро слід навантажити нескінченно повторюваним виконанням тих самих математичних операцій над тими самими даними протягом усього циклу розряджання акумулятора. Варіюючи набір виконуваних операцій та їхні кількісні характеристики, а також задіяну апаратну і програмну платформу, можна з'ясувати їх вплив на енергоспоживання.

Отже, кожен цикл розряджання акумулятора являє собою один захід експерименту, протягом якого усі вхідні параметри і алгоритми обчислень залишаються незмінними. Базовими результуючими даними до кожного такого заходу є набір значень

$$D_k = (t_k, Q_k),$$

де k – порядковий номер вимірювання у діапазоні 1…K; t_k – момент часу, коли проведено вимірювання; Q_k – заряд акумулятора, що залишився, у момент часу t_k; K – кількість вимірювань.

Для зручності подання результатів експерименту тут і далі передбачається, що заряд акумулятора подано у відсотках від максимально можливого.

Безпосередній аналіз D_i вже дає змогу зробити попередні висновки про енергоспоживання. Наприклад, величина

$$T = t_k - t_1$$

являє собою загальний час роботи від максимального до мінімального заряду. Тоді, знаючи ємність акумулятора, можна розрахувати споживання енергії в одиницю часу.

Але на практиці величину T застосовують тільки як попередню оцінку, оскільки на точність її обчислення впливає низка факторів. По-перше, для того, щоб забезпечити мінімальний вплив на енергоспоживання, запис даних D_i як операція, що вимагає звернення до жорсткого диска, повинна виконуватися відносно рідко. Тому момент часу t_1 як відповідний закінченню одиничного блоку обчислень співвідноситься з рівнем заряду акумулятора меншим, ніж 100 %. Аналогічно момент часу t_k зазвичай відповідає рівню заряду в 5–8 %, а не нульовому заряду, оскільки операційна система, дійшовши до деякого критичного рівня, автоматично переходить у режим гібернації. Конкретне значення величини Q_K багато у чому залежить від тривалості виконання одного блоку обчислень. Тривалість виконання такого блоку може суттєво змінюватися під час кожного заходу експерименту, і точна його настройка, як правило, доволі складна і, як наслідок, недоцільна.

Для точнішого оцінювання часу автономної роботи можна скористатися величиною T_H, що обчислюється за формулою

$$T_H = \frac{100}{Q_1 - Q_K}.$$

Величина T_H – гіпотетично максимально можливий час роботи від акумулятора під час розряджання від 100 % до 0 %.

Додатковий вплив на точність величини T впливає похибка вимірювання заряду акумулятора, тому небажано основувати подальші розрахунки на результатах окремих вимірів.

Експериментально виявлено, що графік розряджання акумулятора має яскраво лінійний характер, отже, залежність між рівнем заряду і часом автономної роботи можна записати у вигляді

$$Q_k = a + bt_k + \varepsilon_Q, \qquad (1)$$

де a, b – коефіцієнти лінійної регресії; ε_Q – похибка зміни заряду.

Для визначення скалярних показників залежності (1) має сенс використовувати метод найменших квадратів. Тоді параметри a і b визначатимуться за формулами

$$b = \frac{\overline{\mathrm{Qt}} - \bar{Q}\bar{t}}{\overline{t^2} - \bar{t}^2};$$

$$a = \bar{Q} - b\bar{t}$$

Смислове наповнення параметра *a* складається на рівні заряду акумулятора, що відповідає часу початку випробування. Значення цього параметра, як правило, знаходиться поблизу 100 %, і для подальших досліджень особливого інтересу не становить.

Найцікавіший з дослідницького погляду параметр *b*, оскільки він визначає, як швидко розряджається акумулятор з часом. Значення коефіцієнта *b* завжди негативне. Більше за модулем значення потребує більш енерговитратного обчислення.

Тестова модель для визначення енергетичної ефективності GPU

З урахуванням особливостей архітектури CUDA як конкретної моделі для проведення експериментальних досліджень обрано нейронну мережу, а саме повнозв'язний персептрон Розенблатта [2] із одним виходом (рис. 1). Такий вибір почасти продиктований максимальною простотою мережі цього виду з погляду алгоритму її роботи, а почасти – значною обчислювальною складністю, пов'язаною як з обчисленням вихідного сигналу, так і з навчанням мережі.

Однією з ключових особливостей вибору виду мережі як найпридатнішої для реалізації на графічному ядрі є її повнозв'язність, тобто наявність зв'язків кожного нейрона *i*-го шару з кожним нейроном (i+1)-го шару для усіх *i*.

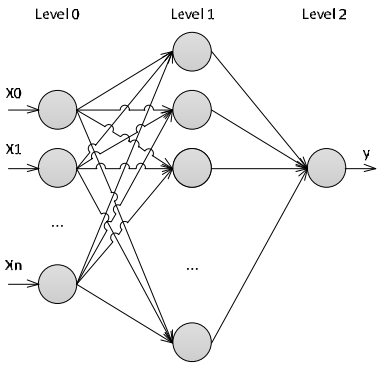

Рис. 1. Модель нейронної мережі, що реалізується

Повнозв'язність забезпечує регулярність структури мережі, за рахунок чого можна обчислювати положення вагових коефіцієнтів у масиві без використання додаткового індексного масиву, що задає зв'язки між нейронами. Це дає змогу знизити кількість звернень до пам'яті, що є одним з вузьких місць у графічному ядрі, а отже, істотно підвищити продуктивність. Повнозв'язність також означає, що буде потрібно виконувати значну кількість обчислень за відносно невеликої кількості вхідних і вихідних даних. Зокрема при прямому поширенні сигналу для кожної пари сусідніх шарів потрібно буде виконати близько до $N_i \cdot N_{i+1}$ обчислювальних операцій, де N_i, N_{i+1} – кількість нейронів в *i*-му та (*i*+1)-му шарах відповідно.

Для навчання мережі використано стандартний метод зворотного поширення помилки [2].

Аналіз енергетичної ефективності

Для порівняльного аналізу енергетичної ефективності було реалізовано моделі нейронної мережі з використанням тільки обчислювальної потужності центрального процесора і з використанням графічного ядра за допомогою технології CUDA. У всіх заходах експерименту представлена нейронна мережа містила три шари. Вихідний шар завжди був представлений одним нейроном. Ряд параметрів, що безпосередньо є кількісними показниками нейронної мережі, змінювався в ході дослідження. До цих параметрів належать:

- *I* – кількість входів нейронної мережі;
- *H* – кількість нейронів у прихованому шарі нейронної мережі.

Наступним змінним параметром є величина *R*, відповідна кількості потоків, що виконуються одночасно. Параметр *R* стосується тільки реалізацій нейронної мережі, що використовують як обчислювач GPU. Реалізація нейронної мережі під CPU завжди використовувала тільки одне процесорне ядро незалежно від фізичних можливостей центрального процесора. Частково це пов'язано із складністю програмної реалізації багатопоточності, частково з істотними витратами на синхронізацію потоків на CPU.

Для врахування впливу особливостей програмної реалізації на енергетичну ефективність, зокрема особливостей взаємодії між CPU і GPU, було задіяно ще два параметри:

C_0 – кількість наборів вхідних значень, що обчислюються у вигляді єдиного пакета даних;

C_1 – кількість повторень обчислень між вимірами рівня заряду акумулятора та відповідного моменту часу.

Введення параметра C_0 знадобилося, оскільки відповідно до принципів технології CUDA в обробці даних у тій або іншій якості бере участь не тільки GPU, але й CPU. CPU при цьому виконує роль пристрою управління, що ініціює процес обчислень. Крім того, джерелом вхідних даних і приймачем результату є ОЗУ, тоді як безпосередньо для обчислень на GPU у цій самій якості використовується глобальна пам'ять відеоадаптера. У зв'язку з цим виникають часові затримки і енергетичні витрати, пов'язані з накладними витратами на пересилання даних між ОЗУ і пам'яттю відеоадаптера, а також на видачу керуючих команд і синхронізацію між CPU і GPU. Одним з найефективніших способів зниження впливу цих факторів є об'єднання декількох послідовних операцій в одну. Параметр C_0 якраз і визначає кількість операцій, інтегрованих в один пакет.

Оскільки при програмної реалізації під CPU вищеописаних проблем не виникає, то параметр C_0 у цих реалізація не задіяний, і у подальших розрахунках приймається як такий, що дорівнює одиниці.

Параметр C_1 задіяний насамперед для того, щоб забезпечити більш-менш однакові інтервали часу між виміром і фіксацією рівня заряду.

Загальна кількість обчислень вихідного значення нейронної мережі в межах одного виміру визначається як

$$C = C_0 C_1 .$$

Значення всіх вищеописаних параметрів, використовувані як найхарактерніший набір значень, перераховано у табл. 1. За кожним заходом експерименту вказують лише значення, які відрізняються від наведених у табл. 1.

Таблиця 1

Базовий набір значень параметрів

	Параметр				
	I	H	R	C_0	C_1
Базове значення	128	1024	512	10000	100

У першій серії експерименту варіювалися параметри C_0 та C_1 у програмної реалізації з використанням технології CUDA і результати зіставлялися з реалізацією під CPU з використанням технології.Net за збереження базових значень параметрів I, H, R. Попередні підсумки цієї серії випробувань наведено у табл. 2.

Таблиця 2

Вплив розміру пакета даних на енергетичну ефективність

Обчислювач	C_0	C_1	T_H , хв.	b, %/хв.
CPU	1	10000	84.2	-1.186
GPU	1	100000	73.2	-1.350
GPU	1000	1000	76.6	-1.298
GPU	10000	100	101.8	-0.972

Результати з табл. 2 добре відображають реальні енерговитрати обчислювальної системи загалом, але для порівняльного аналізу витрат на конкретні обчислення бажано додатково врахувати, що частина енергії, крім програми, що тестується, йшла на роботу фонових програм, на роботу операційної системи і на живлення периферійних пристроїв. Тому було проведено окреме випробування вибраної для цим експериментів програмно-апаратної платформи, і в результаті отримано такі показники

$$T_{H,OS} = 304{,}5 \text{ хв};$$

$$b_{OS} = -0{,}336 \text{ %/хв},$$

де $T_{H,OS}$ – гіпотетичне часу роботи обчислювальної системи у холостому режимі; b_{OS} – кутовий коефіцієнт лінійної регресії роботи обчислювальної системи у холостому режимі.

Тоді енергетична ефективність обчислень визначається за формулою

$$b_p = b - b_{OS}. \quad (2)$$

З урахуванням формули (2) зіставлення енергетичної ефективності різних реалізацій, наведених у табл. 2, зводиться до графіка, представленого на рис. 2.

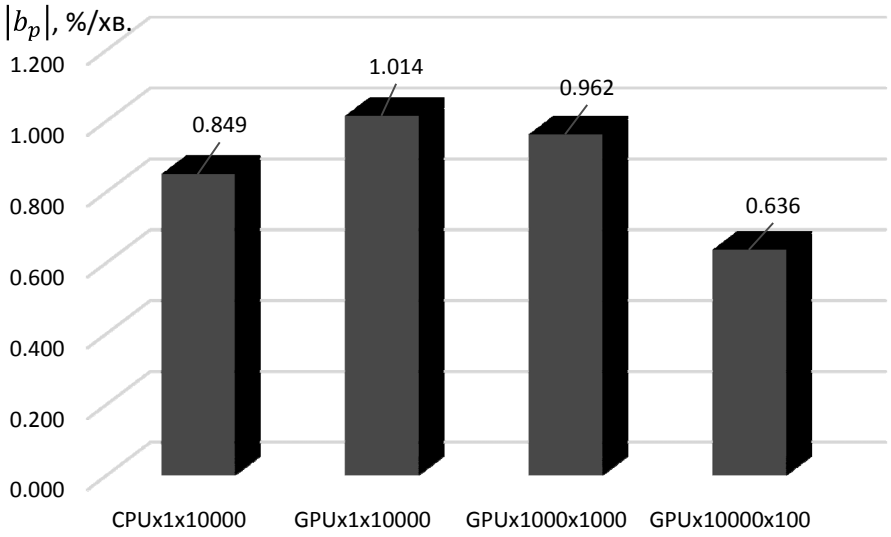

Рис. 2. Абсолютні значення енергетичної ефективності обчислень

Отже, реалізація на GPU з використанням технології CUDA за малих розмірів пакетів даних, таких як один вхідний вектор і навіть 1000 вхідних векторів, програє в енергетичній ефективності аналогічної реалізації на CPU на 16 % і 12 % відповідно.

Із збільшенням розміру пакета даних до 10 тис. енергетична ефективність реалізації на GPU різко зростає і дорівнює за модулем 0.636, що відповідає виграшу в 33 % порівняно з реалізацією аналогічного алгоритму обчислень на CPU.

Такий розподіл енергетичної ефективності пояснюється так. При переході до обчислень на GPU задіюється додаткове обладнання – дискретний графічний адаптер, який у початковому режимі в апаратної платформі, що використовувалася, простоював. Але за малих розмірів пакета даних графічне ядро справляється з розрахунками доволі швидко, і CPU при цьому залишається повністю навантаженим виконанням завдань приймання-передавання даних і видавання керівних команд. Тому очевидно, що енергоспоживання зростає. І тільки за розміру пакетів даних від 10 тис. та більше вузьким місцем з обчислювального погляду стає графічне ядро, а CPU, більш енергоємний порівняно з GPU, виявляється завантаженим не повністю. Внаслідок цього загальне енергоспоживання обчислювальної системи значно знижується.

У цьому випадку не враховувалося, наскільки ефективною з обчислювального погляду була та чи інша реалізація. Розглядалося лише те, яким було енергоспоживання за максимально повного завантаження обраного обчислювача.

У наступній серії експериментів кількість нейронів у прихованому шарі H було зменшено до 512, і результати зіставлялися з даними, отриманими при значеннях параметрів, наведених у табл. 1.

Параметр H у ширшому сенсі є визначальним щодо вимог до обчислювальних ресурсів, оскільки залежність обсягу обчислень із збільшенням значення H є квадратичною і немає необхідності завантажувати додаткові дані з ОЗУ. Цей факт має сенс враховувати, порівнюючи результати описаних експериментів з очікуваною енергетичною ефективністю у разі реалізації за технологією CUDA ширшого класу обчислювальних задач. На рис. 3 наведено результати порівняння енергетичної ефективності обчислень на CPU і GPU за різних значень H.

Як видно з рис. 3, у разі зменшення кількості нейронів у прихованому шарі до 512 використання GPU залишається більш енергоефективним порівняно з CPU, однак виграш в цьому випадку складає лише 21 %, тоді як при H =1024 вдавалося знизити енерговитрати на 33 %. І такі результати узгоджуються з теоретичними припущеннями, оскільки за більшого значення параметра H досягають повнішого завантаження обчислювальних елементів графічного ядра, внаслідок чого менша частина енергії у процентному відношенні йде на пересилання даних між ОЗП і відеоадаптером, а також на управління процесом з боку CPU.

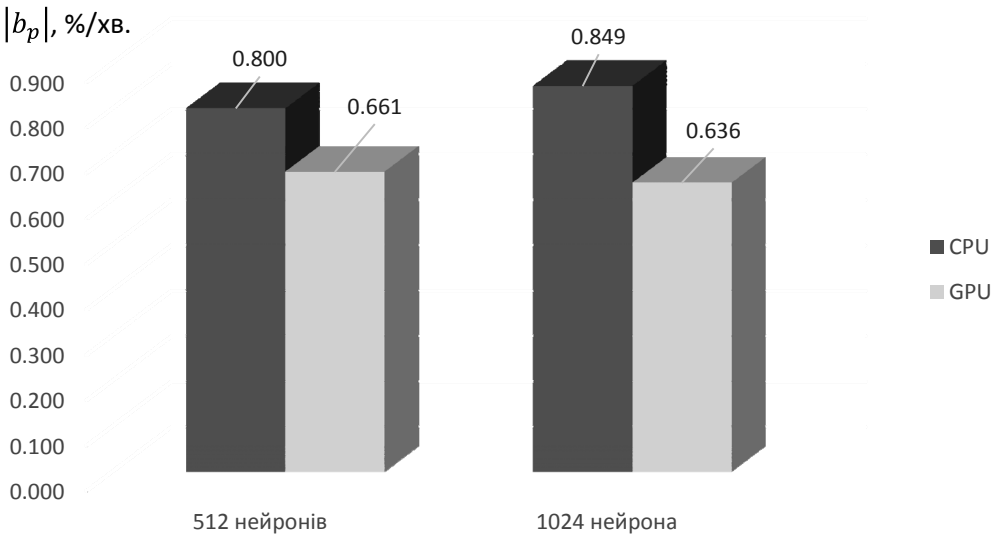

Рис. 3. Залежність енергетичної ефективності від кількості нейронів у прихованому шарі

У наступних серіях експериментів варіювалися параметри I та T_H . У всіх випадках GPU демонструвала стабільно високу енергетичну ефективність.

Висновок

Порівнюючи реалізації на CPU і на GPU, виявлено, що технологія CUDA демонструє вищу енергетичну ефективність навіть за повного навантаження GPU. Так, при 512-ти потоках економія енергії порівняно з реалізацією на CPU 33 %.

Аналогічні результати очікуються при реалізації інших видів нейронних мереж, таких як мережі Кохонена, з використанням технології CUDA [5].

1. CUDA Zone. http://www.nvidia.ru/object/cuda_home_new_ru.html (Last access: 08.02.2014). 2. Каллан Р. Основные концепции нейронных сетей.: Пер. с англ. – М.: Издательский дом "Вильямс", 2001, 288 с., ISBN 5-8459-0210-X. 3. Изопов П. Ю., Суханов С. В., Головашкин Д. Л. Технология реализации нейросетевого алгоритма в среде CUDA на примере распознавания рукописных цифр. – М.: Институт систем обработки изображений РАН "Компьютерная оптика", Т. 34. – №2. – 2010. – С. 243–251, ISSN 0134-2452. 4. Олещук О. В., Попель О. Є., Копитчук М. Б. Моделювання повнозв'язної нейронної мережі з використанням технології CUDA. – Л.: Видавництво Національного університету "Львівська політехніка" (Вісник / Національний університет "Львівська політехніка"), № 747, 2012, с. 131–139. 5. Oleshchuk O., Popel O., Kopytchuk M. Application of Kohonen neural networks for problems of on-line diagnostics. – 13th International Conference "Research and Development in Mechanical Industry" RaDMI, 2013, Vrnjacka Banja, Serbia, p. 705–710.