Parallel Gaussian Elimination Method

Muhammad W. Muhammad Ali
College of Education for Girls
University of Mosul, Iraq

Received on: 17/10/2007 **Accepted on: 04/03/2008**

ABSTRACT

The aim of the project is to develop parallel approaches for Gaussian Elimination Methods that are used in linear programming to solve linear module systems.

Most of these models are time-consuming when executed and processed in the sequential microprocessor computers. During the project, we try to decrease this time and increase the efficiency of the algorithm for the Gaussian Elimination Method, through developing parallel methods appropriate to be executed on MIMD type computers.

In this paper, three algorithms were suggested for paralleling a developed algorithm of Gaussian Elimination Method and a comparison was made between the three algorithms and the original.

As we have been able to accelerate the three parallel methods and the speedup was one of the following:

Speedup = $S_p = \frac{t_s}{t_p} = \frac{2.43}{1.47} = 1.6531$, no. of processor is (50)

In general, the practical results and the suggested programs for these new algorithms proved to be better in performance than their analogues that are executed in computers of sequential processor in view of the two elements of execution time and algorithm time.

Keywords: linear programming, Gaussian Elimination Method.

طريقة كاوس للحذف المتوازية

محمد واجد محمد علي

كلية التربية للبنات، جامعة الموصل

تاريخ استلام البحث: 2007/10/17 **تاريخ قبول البحث: 2008/03/04**

الملخص

هدف البحث هو تطوير طرائق متوازية لطريقة كاوس للحذف Gaussian Elimination (GE) والتي تستعمل في البرمجة الخطية لحل منظومات من النماذج الخطية.

ان معظم هذه النماذج تتطلب وقتاً كبيراً للتنفيذ عند المعالجة باستخدام حاسبات ذات معالج تتابعي، ونحاول في هذا البحث تقليل هذا الوقت وزيادة كفاءة الخوارزميات لطريقة كاوس للحذف من خلال تطوير ثلاث طرائق متوازية ملائمة للتنفيذ على حاسبات نوع MIMD.

في هذا البحث تم اقتراح ثلاث خوارزميات جديدة للتوازي مطورة نسبة الى طريقة كاوس للحذف Gaussian Elimination (GE) كما تمت المقارنة بين هذه الخوارزميات المقترحة مع الخوارزمية الاصلية. إذ تمكنا من تسريع الطرائق الثلاث باستخدام التوازي وكان عامل التسريع لاحد الطرائق كالاتي:

$$S_p = \frac{t_s}{t_p} = \frac{2.43}{1.47} = 1.6531$$

إذ أن عدد المعالجات المستخدمة في هذه الطريقة هو (50) معالجاً.

وعلى العموم أظهرت النتائج العملية والبرمجيات الحاسوبية المقترحة لهذه الخوارزميات الجديدة بانها افضل من مثيلاتها التي تنفذ على حسابات ذات معالج تتابعي نسبة الى عنصري زمن التنفيذ وسرعة الخوارزمية.

الكلمات المفتاحية: البرمجة الخطية، طريقة الحذف لكاوس.

1. مقدمة:

ان نظام المعادلات الجبرية الخطية هو عبارة عن عدد من المعادلات الجبرية الخطية في عدد من المجاهيل ، ويكون حل النظام هو ايجاد قيم هذه المجاهيل، ومن الممكن التعرف على الشكل العام لمنظومة المعادلات الجبرية الخطية التي تتكون من n من المعادلات الخطية و n من المجاهيل بالصيغة:

$$\begin{array}{l} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_1 + \cdots + a_{2n}x_n = b_2 \\ \quad\vdots \qquad\quad \vdots \qquad \ddots \qquad\quad \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n \end{array} \quad \ldots (1)$$

اذ ان المعاملات (Coefficients) a_{ij} , i,j = 1..n وكذلك الحدود المطلقة (Free terms) b_i , i = 1..n تكون معطاة اما x_i, i = 1..n فهي المجاهيل مطلوب الحصول على قيمها [2].

وبالإمكان كتابة هذه المنظومة بصيغة موجزة على النحو الآتي [6]، [14]:

$$AX = B$$

اذ ان :

$$A=\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix},\ X=\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix},\ B=\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

هذا النوع من المعادلات مهم جداً وبرز بشكل مكرر في المواد العلمية والطبية والتطبيقات الاقتصادية، العديد من طرائق التحليلات العددية تستخدم نظام المعادلات الخطية لتقريب اكثر المسائل الرياضية غير الخطية، فعلى سبيل المثال تنبؤ حالة الطقس فرع من العلم الذي يجب ان يحل انظمة معادلات كبيرة جداً لضبط احوال الطقس المستقبلية. وطريقة كاوس للحذف هي الطريقة التي تستطيع ان تؤدي هذه المهمة بطريقة مضبوطة وسليمة [14].

ومن المعلوم ان معظم هذه الانظمة تأخذ وقتاً طويلاً للتنفيذ عند المعالجة باستخدام حاسبات ذات معالج تتابعي، فلهذا السبب كان هدف البحث هو ايجاد طرائق جديدة متوازية لتقليل هذا الوقت وزيادة كفاءة الطريقة.

2. طريقة كاوس للحذف:

وهي من الطرائق المباشرة الشائعة وابسطها لحل منظومة المعدلات الخطية، نبدأ مع نظام المعادلات (1) في البداية ننشئ المصفوفة الموسعة Argument Matrix:

$$\left[\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & b_n \end{array}\right]\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad \ldots (2)$$

سنقوم بتحويل هذه المصفوفة الى مصفوفة مثلثية عليا [14]:

$$\left[\begin{array}{cccc|c} a_{11}^{(n-1)} & a_{12}^{(n-1)} & \cdots & a_{1n}^{(n-1)} & b_1^{(n-1)} \\ 0 & a_{22}^{(n-1)} & \cdots & a_{2n}^{(n-1)} & b_2^{(n-1)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & a_{nn}^{(n-1)} & b_n^{(n-1)} \end{array}\right]\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad \ldots(3)$$

ومنها نحصل على ما يأتي:

$$x_n = \frac{b_n^{(n-1)}}{a_n^{(n-1)}}$$

ومن ثم باستخدام طريقة التعويض التراجعي لنحصل على قيم x_i , i = 1..n-1 وذلك عن طريق المعادلة الآتية [14]:

$$x_i = \frac{1}{a_{ii}}(b_i - \sum_{j=i+1}^{n} a_{ij}x_j)\ ;\ \ i=1..n-1$$

2.1 خطوات حل طريقة كاوس للحذف [6]،[8]:

تنفذ خطوات حل طريقة كاوس للحذف عن طريق خطوتين اساسيتين:

2.1.1 الحذف الامامي:

أولا: ازالة x_1 من المعادلة الثانية باستخدام المعادلة الاولى، وذلك بضرب المعادلة الاولى بـ $\left(\frac{a_{21}}{a_{11}}\right)$ ومن ثم طرحها من المعادلة الثانية لنحصل على المعادلة الثانية الجديدة والتي تكون خالية من x_1:

$$E'q2 \Rightarrow \left(a_{21} - \frac{a_{21}}{a_{11}}a_{11}\right)x_1 + \left(a_{22} - \frac{a_{21}}{a_{11}}a_{12}\right)x_2 + \cdots + \left(a_{2n} - \frac{a_{21}}{a_{11}}a_{1n}\right)x_n = b_2 - \frac{b_1 a_{21}}{a_{11}}$$

أو بالإمكان كتابتها بالشكل الآتي:

$$E'q2 \Rightarrow a'_{22}x_2 + \cdots + a'_{2n}x_n = b'_2$$

اذ ان :

$$a'_{22} = a_{22} - \frac{a_{21}}{a_{11}}a_{12}, \cdots, a'_{2n} = a_{2n} - \frac{a_{21}}{a_{11}}a_{1n}\ ,\ b'_2 = b_2 - \frac{b_1 a_{21}}{a_{11}}$$

وبالطريقة نفسها يتم حذف x_1 من المعادلة الثالثة والرابعة و... الى المعادلة n وذلك من حساب الاتي:

$$\left.\begin{aligned} E'q3 &= Eq3 - \left(\frac{Eq1}{a_{11}}\right)\times a_{31} \\ E'q4 &= Eq4 - \left(\frac{Eq1}{a_{11}}\right)\times a_{41} \\ &\vdots \\ E'qn &= Eqn - \left(\frac{Eq1}{a_{11}}\right)\times a_{n1} \end{aligned}\right\} \quad \ldots(4)$$

فنحصل على النظام الآتي:

$$\begin{array}{ccccccc} a_{11}x_1 & + & a_{12}x_2 & + & \cdots & + a_{1n}x_n & = b_1 \\ & & a'_{22}x_2 & + & \cdots & + a'_{2n}x_n & = b'_2 \\ & & a'_{32}x_2 & + & \cdots & + a'_{3n}x_n & = b'_3 \\ & & \vdots & & \ddots & \vdots & \vdots \\ & & a'_{n2}x_2 & + & \cdots & + a'_{nn}x_n & = b'_n \end{array}$$

ثانيا: ازالة x_2 من المعادلتين الثالثة والرابعة ... والى المعادلة n وبالطريقة نفسها التي تمت ازالة x_1 من باقي المعادلات، أي انه:

$$E''q3 = Eq'3 - \left(\frac{E'q2}{a_{22}}\right) \times a_{32}$$

وهذا الإجراء يتكرر على المعادلات الباقية لنحصل على ما يأتي:

$$\begin{array}{lllll} a_{11}x_1 + & a_{12}x_2 + & a_{13}x_3 + \cdots & + a_{1n}x_n & = b_1 \\ & a'_{22}x_1 + & a'_{23}x_3 + \cdots & + a'_{2n}x_n & = b_2 \\ & & a''_{33}x_3 + \cdots & + a''_{3n}x_n & = b_3 \\ & & \ddots & \vdots & \vdots \\ & & a''_{2n3}x_3 + \cdots & + a''_{nn}x_n & = b_n \end{array}$$

ونحذف x_3 من المعادلتين الرابعة والخامسة... إلى المعادلة n ، ونحذف x_4 من المعادلة الخامسة ... الى المعادلة n، وهكذا الى ان نحصل على النظام الآتي:

$$\begin{array}{lllll} a_{11}x_1 + & a_{12}x_2 + & a_{13}x_3 + \cdots & + a_{1n}x_n & = b_1 \\ & a'_{22}x_1 + & a'_{23}x_3 + \cdots & + a'_{2n}x_n & = b'_2 \\ & & a''_{33}x_3 + \cdots & + a''_{3n}x_n & = b''_3 \\ & & \ddots & \vdots & \vdots \\ & & \cdots & + a_{nn}^{(n-1)}x_n & = b_n \end{array}$$

2.1.2 التعويض التراجعي:

نبدأ من المعادلة n سنحصل على اول العناصر وهو x_n :

$$x_n = \frac{b_n^{(n-1)}}{a_{nn}^{(n-1)}}$$

ونحصل على بقية العناصر من المعادلة الآتية:

$$x_i = \frac{b_i^{(i-1)} - \sum_{j=i+1}^{n} a_{ij}^{(i-1)} x_j}{a_{ii}^{(i-1)}} \quad , \; i = n-1, n-2, \ldots 1$$

2.2 خوارزمية طريقة كاوس للحذف [12]:

الخطوة 1) الحذف الامامي:

```
Do k = 1; N - 1
    Do i = k + 1; N
```

$b_i = b_i - (a_{ik}/a_{kk})\, b_k$
Do j = k + 1;N
$a_{ij} = a_{ij} - (a_{ik}/a_{kk})\, a_{kj}$
Repeat j
Repeat i
Repeat k

الخطوة 2) التعويض التراجعي

$x_N = b_N / a_{NN}$
Do i = N - 1; 1
$y_i = 0$
Do j = i + 1;N
$y_i = y_i + a_{ij}x_j$
Repeat j
$x_i = (b_i - y_i)/a_{ii}$
Repeat i

2.3 مثال تطبيقي:

باستخدام طريقة كاوس للحذف جد حلاً لمنظومة المعادلات الآتية:

$$\begin{array}{lllllllll} 11 & x_1 + & 6 & x_2 + & 10 & x_3 + & 12 & x_4 + & 9 & x_5 = & 273 \\ 7 & x_1 + & 3 & x_2 + & 4 & x_3 + & 3 & x_4 + & 5 & x_5 = & 112 \\ 10 & x_1 + & 7 & x_2 + & 12 & x_3 + & 8 & x_4 + & 8 & x_5 = & 266 \\ 2 & x_1 + & 9 & x_2 + & 5 & x_3 + & 12 & x_4 + & 10 & x_5 = & 227 \\ 5 & x_1 + & 4 & x_2 + & 11 & x_3 + & 9 & x_4 + & 2 & x_5 = & 213 \end{array}$$

وبعد اجراء خطوات الحل يكون الحل كالاتي:

$$x_1 = 3\,,\; x_2 = 8\,,\; x_3 = 9\,,\; x_4 = 7\,,\; x_5 = 2$$

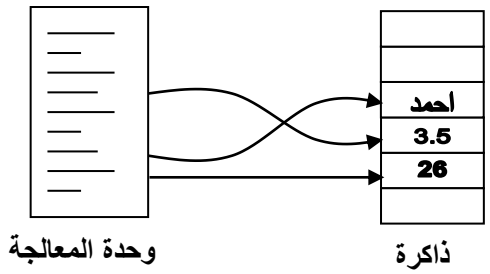
3. التوازي في طريقة كاوس للحذف:

سنناقش في هذا البند ثلاث طرائق متوازية جديدة تم التوصل إليها من خلال دراستنا لموضوع التوازي لطريقة كاوس للحذف، علماً ان هناك بحوثاً أجريت لإيجاد حل نظام المعادلات الخطية بطريقة الحذف لكاوس، مثل بحث Khalil K. Abbo [13] الذي اقترح خوارزميتين متوازيتين، اعتمد في الأولى على توازي العمليات في الصفوف Parallel Gaussian Elimination Based on Rows (PGER)، واعتمد على الثانية على توازي الأعمدة Parallel Gaussian Elimination Based on Columns (PGEC).

وبما إننا نتكلم على الحاسبات المتوازية فلا بد من التطرق إليها:

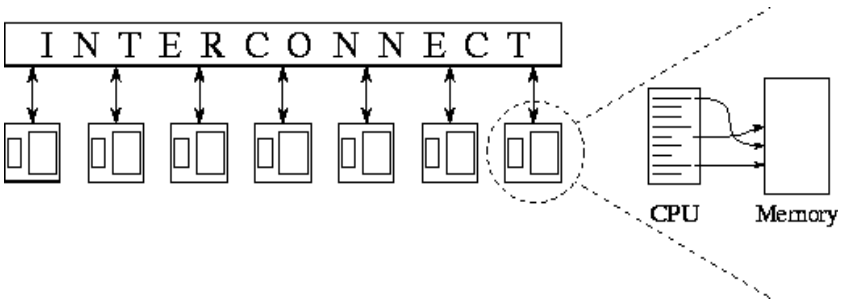
3.1 الحاسبات المتوازية:

يعود الفضل للتقدم السريع في صناعة الحاسبات ودخولها مجالات عدة مثل التجارة والعلوم والتعليم لنموذج الحاسبة الأولى (Single Machine Model) لحاسبات Von Neuman، وكما هو معروف فان هذه الحاسبات تمتلك وحدة معالجة مركزية واحدة (CPU) مرتبطة بوحدة خزن (Memory) (الشكل(1))، وان المعالج يقوم بتنفيذ برنامج مخزون والذي يحدد بواسطة مجموعة متسلسلة من علميات القراءة والكتابة على الذاكرة [10].



الشكل (1) حاسبة Von Neuman، تقوم وحدة المعالجة المركزية (CPU) بانجاز البرنامج الذي ينفذ بشكل تسلسلي لعمليات القراءة والكتابة على الذاكرة المحجوزة [10].

بينما في الحاسبات المتوازية فهو ايجاد نموذج متوازٍ يتكون من عدد من حاسبات Von Neuman (الشكل(2)) يكون عاماً لكثير من التطبيقات وان تمتلك الحاسبة المتوازية ميزتين اساسيتين هما (البساطة والواقعية)، والبساطة تعني امكانية فهم الحاسبة والبرمجة لها، والواقعية هي ضمان تنفيذ النماذج البرمجية لها بكفاءة معقولة [9].



الشكل (2) : نموذج حاسبة متوازية مثالية، تحتوي كل نقطة ارتباط على حاسبة من نوع Von Neuman، ويمكن لكل نقطة الاتصال بالنقاط الاخرى عن طريق ارسال الرسائل واستقبالها عبر الشبكة المربوطة. [10]

3.2 تعاريف:

3.2.1 الحاسوب المتوازي:

هو مجموعة من المعالجات التي بامكانها العمل بصورة متوازية لحل مسألة حسابية ما، وهذا التعريف يشمل الحاسبات المتوازية التي تمتلك عشرات او مئات من المعالجات (Processors) [9].

أو هو حاسوب يستخدم عدة معالجات تعمل بشكل متزامن (أي تعمل في الوقت نفسه) لحل مسألة او لأداء وظيفة معينة[5].

3.2.2 الخوارزمية المتوازية:

الخوارزمية: هي مجموعة من التوجيهات لتنفيذ عمليات حسابية مصممة بشكل يؤدي الى حل المسألة المعطاة [1]، [3].

وعليه فمن الممكن تعريف الخوارزمية المتوازية: بأنها تجزئة مسألة واحدة الى عدة مسائل جزئية مستقلة حتى يمكن حل كل مسألة جزئية في معالج مستقل ثم جمع الحلول لتركيب حل المسألة الاصلية [9].

3.2.3 قياس الوقت وحسابه:

تقاس القدرة الحسابية للحاسوب بعدد العمليات التي ينفذها بالثانية على الاعداد الحقيقية Mega FLOPS (Floating Point Operation Per Second) ، وتتطلب التطبيقات الحسابية قدرة حسابية من درجة الف مليون عملية في الثانية، لقد وصلت الحاسبات الفائقة في عام 2003 الى 35TEFLOPS وهو ما يقارب 36.5 تريليون عملية في كل ثانية، ومن اجل تحسين الاداء يجب تقصير زمن دورة المعالج للعملية الواحدة وزيادة العمليات التي تنفذ على التوازي. ولكن تسريع المعالجات لا يكفي اذا لم يصاحب القدرة على نقل المعلومات من الذاكرة الى وحدة المعالجة بالسرعة الكافية وتقاس هذه القدرة بعدد البايتات Bytes التي يمكن نقلها من الذاكرة الى وحدة المعالجة في الثانية [5].

ملاحظة مهمة:

ان حاسبات من نوع MIMD غير متوافرة في بلادنا، فلذلك تم تنفيذ كل إجراء على حده وحساب الوقت، ويكون الوقت الأكبر هو الوقت المعتمد في حساب تنفيذ الإجراءات المتوازية، وهذا

ممكـــــــن اذ ان المعـــــــالج المســـــــتخدم فـــــــي الحاســـــــبات المتوازيـــــــة هـــو المعـــالج نفســـه المســـتخدم فـــي الحاســـبات الشخصـــية، كمـــا توصـــل اليـــه الباحـــث Y.F. Fung, et al في دراسته اذ اثبت انه بامكان الطلاب استخدام حاسباتهم الشخصية لتنفيذ البرامج والخورازميات المتوازية ومعالجتها وذلك بسبب ارتفاع كلفة الحاسبات ذات المعالجات المتعددة [15].

3.2.4 التسريع (S_p): [5]

هو حاصل قسمة الزمن التسلسلي (t_s) على الزمن المتوازي (t_p).

اي ان : $S_p = \frac{t_s}{t_p}$

3.3 تصنيف الحاسبات المتوازية:

يعمل أي نوع من أنواع الحاسبات (حاسبات تسلسلية كانت أم متوازية) على تنفيذ مجموعة من الايعازات (Instructions) على مجموعة من البيانات (Data) ومن ثم تتم معالجتها وإخراجها على شكل معلومات[9].

وهناك تصانيف عدة لمعمارية الحاسبات، من اشهر هذه التصانيف تصنيف الباحث Flynn [10]، فقد صنف انظمة بحسب نوع التحكم (Type of Control) والخوارزميات (Algorithms) والمعالجات المنطقية (Logical Processors) الى اربعة اصناف رئيسة [4]، [9] هي:

1- معالجات ذات ايعاز فردي وبيانات جارية مفردة

SISD: Single Instruction Stream, Single Data Stream.

2- معالجات ذات ايعازات متعددة وبيانات جارية مفردة.

MISD: Multiple Instruction Stream, Single Data Stream.

3- معالجات ذات ايعاز فردي وبيانات جارية متعددة.

SIMD: Single Instruction Stream, Multiple Data Stream.

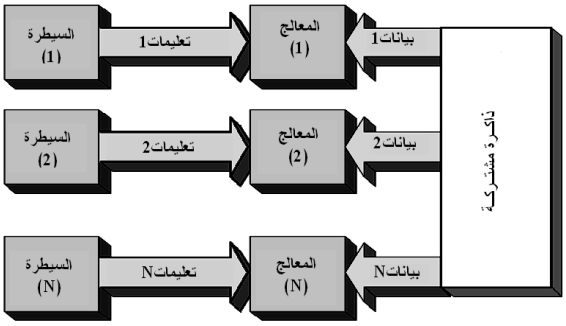
4- معالجات ذات ايعازات متعددة وبيانات جارية متعددة.

MIMD: Multiple Instruction Stream, Multiple Data Stream.

وهدفنا من البحث هو تطوير طرائق متوازية ملائمة للتنفيذ على حسابات من نوع MIMD ولهذا سوف نقتصر الحديث عنها:

3.4 حاسبات من نوع MIMD: [9]

يعد هذا النوع من الحاسبات الاكثر شيوعاً وكفاءة وقوة في تنفيذ التطبيقات العامة في موضوع الحاسبات المتوازية اذ تكون فيها مجموعة الايعازات ومجموعة البيانات مختلفة.
يحتوي هذا النوع من الحاسبات على (N) من المعالجات وعلى (N) من الايعازات وعلى (N) من البيانات (الشكل (3)). ويمتلك كل معالج ذاكرة خاصة به فضلاً عن وحدة الحساب والمنطق ووحدة السيطرة الخاصة به، وله القدرة على العمل بصورة ليست متزامنة ومستقلة على بياناته الجارية الخاصة، وفي اي وقت من الممكن ان تقوم معالجات مختلفة بتنفيذ ايعازات مختلفة وعلى بيانات مختلفة [4].



الشكل (3): مخطط توزيع الايعازات والبيانات في حاسبات من نوع MIMD [15].

ونأتي الان إلى شرح الطرائق التي تم التوصل اليها:

3.5 الطريقة المتوازية الجديدة الأولى:

3.5.1 التوازي في الحذف الامامي:

كما لاحظنا خطوات حل طريقة كاوس للحذف؛ فإن هناك خطوات مستقلة بعضها عن البعض الآخر أي أنه من الممكن ان ننجزها في الوقت نفسه؛ لأنها لا تعتمد على بعضها البعض، وهذه الخطوات المستقلة هي خطوات الحذف الامامي، فإذا اخذنا المعادلات (4) المستخدمة لحذف x_1 من جميع المعادلات الا المعادلة الاولى:

$$E'q2 = Eq2 - \left(\frac{Eq1}{a_{11}}\right) \times a_{21}$$

$$E'q3 = Eq3 - \left(\frac{Eq1}{a_{11}}\right) \times a_{31}$$

$$E'q4 = Eq4 - \left(\frac{Eq1}{a_{11}}\right) \times a_{41}$$

.
.

.

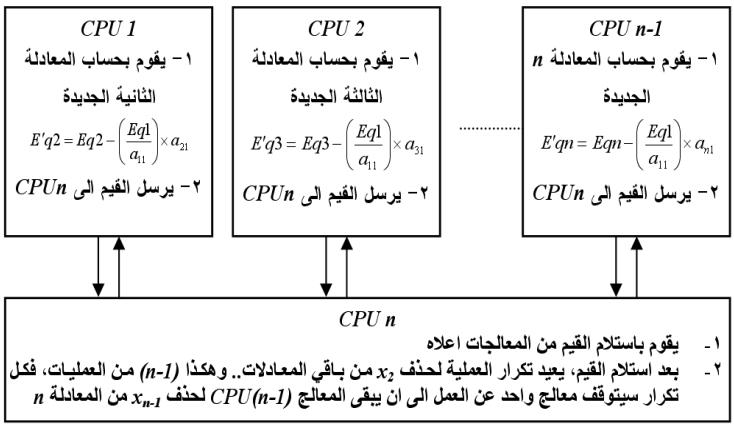
$$E'qn = Eqn - \left(\frac{Eq1}{a_{11}}\right) \times a_{n1}$$

فمن الممكن تنفيذ كل معادلة على معالج مستقل، ومن ثم يكون الزمن المحسوب لتنفيذ هذه الخطوات هو زمن تنفيذ معادلة واحدة.

والكلام نفسه يتكرر بالنسبة إلى حذف x_2 من المعادلتين الثالثة والرابعة الى المعادلة n، وهكذا بالنسبة إلى حذف x_i من المعادلة j ، اذ ان i=1..n-1 و j = i+1..n .

من الواضح اننا نحتاج في هذه الطريقة الى n من المعالجات، لان العمليات تتكرر (n-1) من المرات ويبقى معالج واحد للجمع بين المعالجات (الشكل (4)).

ويمكن تمثيل الطريقة المتوازية بالشكل الآتي:



الشكل (4): توزيع الأوامر على المعالجات عند مرحلة الحذف الامامي لطريقة كاوس للحذف.

3.5.2 توزيع الأوامر على المعالجات عند مرحلة الحذف الامامي:

الـCPU1 يقوم بحساب المعادلة الثانية الجديدة بعد حذف x_1 منها حسب المعادلة الآتية:

$$E'q2 = Eq2 - \left(\frac{Eq1}{a_{11}}\right) \times a_{21}$$

الـCPU2 يقوم بحساب المعادلة الثالثة الجديدة بعد حذف x_1 منها حسب المعادلة الآتية:

$$E'q3 = Eq3 - \left(\frac{Eq1}{a_{11}}\right) \times a_{31}$$

الـCPUn-1 يقوم بحساب المعادلة n الجديدة بعد حذف x_1 منها حسب المعادلة الآتية:

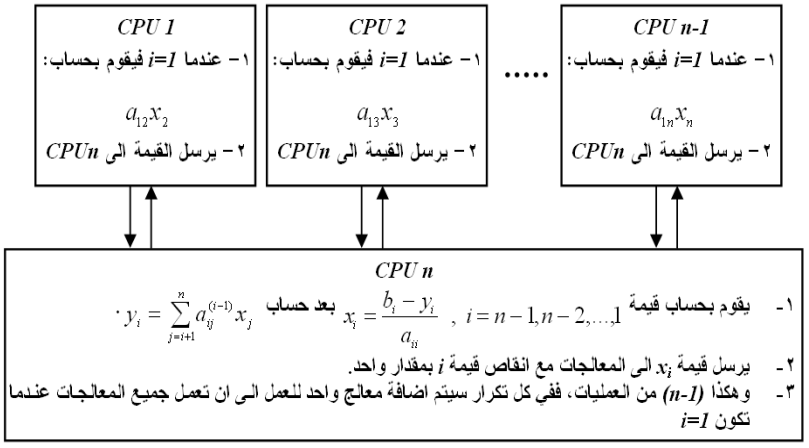
$$E'qn = Eqn - \left(\frac{Eq1}{a_{11}}\right) \times a_{n1}$$

الـCPUn يقوم باستلام القيم من المعالجات ومن ثم يرسل المعادلات الجديدة لحذف x_i من المعادلات j اذ ان i=1..n-1 و j=i+1..n .

3.5.3 التوازي في التعويض التراجع:

بعد الانتهاء من الحذف الامامي سيأتي الان دور التعويض التراجعي؛ والذي فيه خطوات مستقلة؛ وهذه الخطوات المستقلة هي ايجاد $y_i = \sum_{j=i+1}^{n} a_{ij}^{(i-1)} x_j$ لكل i=n-1,n-2 ..,1 وبعدها يتم الحصول على x_i من المعادلة: $x_i = \frac{b_i^{(i-1)} - y_i}{a_{ii}^{(i-1)}} \quad , \; i = n-1, n-2, \ldots, 1$

فسيكون لدينا n من المعالجات، اذ ان المعالج الاول سيكون للجمع بين المعالجات واستلام القيم وارسالها، وبقية المعالجات ستعمل بشكل متوازٍ وتبدأ بمعالج واحد وتنتهي بـn-1 من المعالجات (الشكل (5))، وكما موضح في الشكل ادناه:



الشكل(5): توزيع الأوامر على المعالجات عند مرحلة التعويض المتراجع لطريقة كاوس للحذف.

3.5.4 توزيع الأوامر على المعالجات عند مرحلة الحذف الامامي:

الـCPU1 يقوم بحساب $a_{12}x_2$ عندما i=1.

الـCPU2 يقوم بحساب $a_{13}x_3$ عندما i=1.

.
.
.

الـCPUn-1 يقوم بحساب $a_{1n}x_n$ عندما i=1.

الـCPUn يقوم بحساب $x_i = \frac{b_i^{(i-1)} - y_i}{a_{ii}^{(i-1)}}$, $i = n-1, n-2, \ldots, 1$ بعد ايجاد $y_i = \sum_{j=i+1}^{n} a_{ij}^{(i-1)} x_j$.

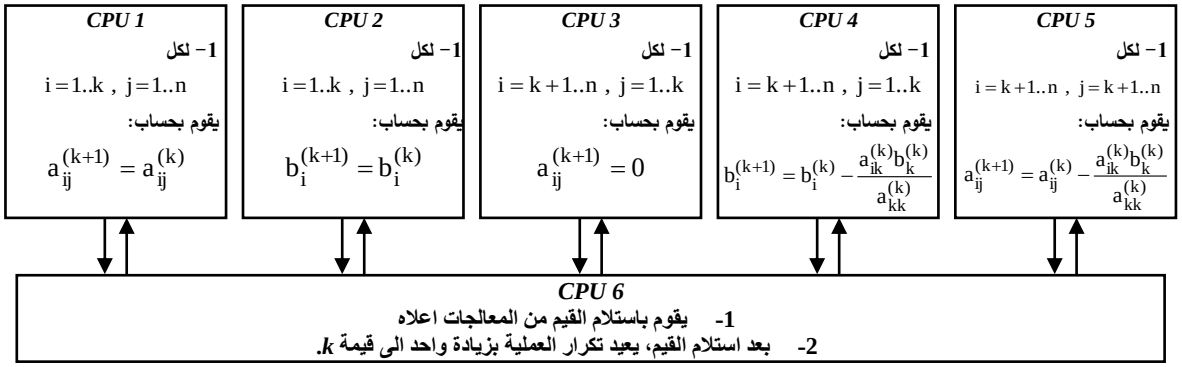
3.6 الطريقة المتوازية الجديدة الثانية:

من الملاحظ ان خطوات حل طريقة كاوس للحذف يمكن كتابتها بالشكل الاتي:

$$k = 1..n-1 \left\{ \begin{array}{l} \left. \begin{array}{l} a_{ij}^{(k+1)} = a_{ij}^{(k)} \\ b_i^{(k+1)} = b_i^{(k)} \end{array} \right\} ; i = 1..k \ , \ j = 1..n \quad (a) \\ \left. \begin{array}{l} a_{ij}^{(k+1)} = 0 \\ b_i^{(k+1)} = b_i^{(k)} - \frac{a_{ik}^{(k)} b_k^{(k)}}{a_{kk}^{(k)}} \end{array} \right\} ; i = k+1..n \ , \ j = 1..k \quad (b) \\ \left. a_{ij}^{(k+1)} = a_{ij}^{(k)} - \frac{a_{ik}^{(k)} b_k^{(k)}}{a_{kk}^{(k)}} \right\} ; i = k+1..n \ , \ j = k+1..n \quad (c) \end{array} \right\} \quad \ldots (5)$$

كما نلاحظ ان هذه المعادلات مستقلة فيما بينها أي أنه من الممكن ان ننجزها في الوقت نفسه؛ لأنها لا تعتمد على بعضها البعض، وهذه الخطوات المستقلة هي خطوات الحذف الامامي. ففي هذه الطريقة نحتاج الى 6 من المعالجات، لان المعادلات عددها 5 ويبقى معالج واحد للجمع بين المعالجات (الشكل (6)).

ويمكن تمثيل الطريقة المتوازية بالشكل الآتي:



الشكل (6): توزيع الأوامر على المعالجات في الطريقة الثانية للتوازي لطريقة كاوس للحذف.

3.6.1 توزيع الأوامر على المعالجات للطريقة المتوازية الثانية:

الـCPU1 يقوم بحساب المعادلة $a_{ij}^{(k+1)} = a_{ij}^{(k)}$ اذ ان $i=1..k\ ,\ j=1..n$

الـCPU2 يقوم بحساب المعادلة $b_{i}^{(k+1)} = b_{i}^{(k)}$ اذ ان $i=1..k\ ,\ j=1..n$

الـCPU3 يقوم بحساب المعادلة $a_{ij}^{(k+1)} = 0$ اذ ان $i=k+1..n\ ,\ j=1..k$

الـCPU4 يقوم بحساب المعادلة $b_{i}^{(k+1)} = b_{i}^{(k)} - \frac{a_{ik}^{(k)} b_{k}^{(k)}}{a_{kk}^{(k)}}$ اذ ان $i=k+1..n\ ,\ j=1..k$

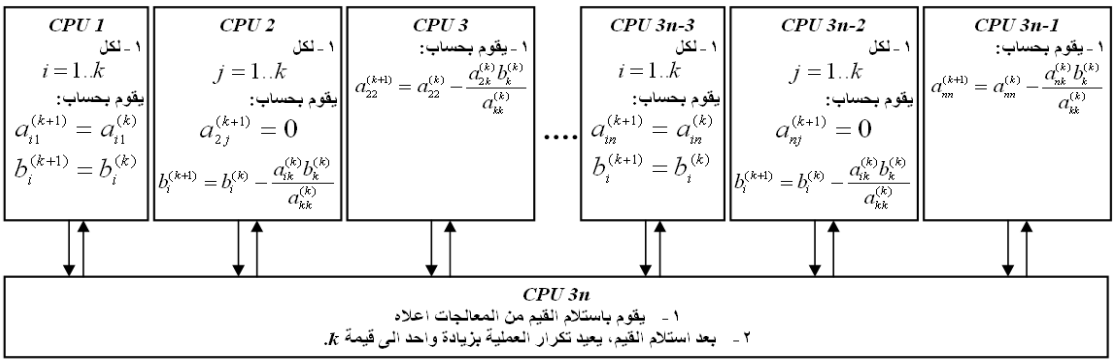
الـCPU5 يقوم بحساب المعادلة $a_{ij}^{(k+1)} = a_{ij}^{(k)} - \frac{a_{ik}^{(k)} b_{k}^{(k)}}{a_{kk}^{(k)}}$ اذ ان $i=k+1..n\ ,\ j=k+1..n$

الـCPU6 يقوم باستلام القيم من المعالجات ويعيد تكرار العمليات بعد زيادة واحد الى قيمة k.

3.7 الطريقة المتوازية الجديدة الثالثة:

من الملاحظ ان خطوات حل طريقة كاوس للحذف المعادلات (5a), (5b) & (5c) والخاصة بالحذف الأمامي يمكن توزيعها على معالجات لانها معادلات مستقلة لا يعتمد أي منها على الآخر (الشكل (7)).

في هذه الطريقة سنحتاج الى 3n من المعالجات، وكما موضح في الشكل ادناه:



الشكل (7): توزيع الأوامر على المعالجات في الطريقة الثالثة للتوازي لطريقة كاوس للحذف.

وبالامكان استخدام الطريقة المتوازية في حالة التعويض المتراجع نفسها المذكورة في الطريقة الاولى لزيادة التسريع.

3.7.1 توزيع الاوامر على المعالجات للطريقة المتوازية الثالثة:

الـCPU1 يقوم بحساب $a_{i1}^{(k+1)} = a_{i1}^{(k)}$ ، $b_{i}^{(k+1)} = b_{i}^{(k)}$ اذ ان $i=1..k$

الـCPU2 يقوم بحساب $a_{2j}^{(k+1)} = 0$ ، $b_{2}^{(k+1)} = b_{2}^{(k)} - \frac{a_{2k}^{(k)} b_{k}^{(k)}}{a_{kk}^{(k)}}$ اذ ان $j=1..k$

الـCPU3 يقوم بحساب المعادلة $a_{2j}^{(k+1)} = a_{2j}^{(k)} - \frac{a_{2k}^{(k)} b_k^{(k)}}{a_{kk}^{(k)}}$ اذ ان $j = k+1..n$

الـCPU4 يقوم بحساب $a_{i2}^{(k+1)} = a_{i2}^{(k)}$ ، $b_i^{(k+1)} = b_i^{(k)}$ اذ ان $i = 1..k$

الـCPU5 يقوم بحساب $a_{3j}^{(k+1)} = 0$ ، $b_3^{(k+1)} = b_3^{(k)} - \frac{a_{3k}^{(k)} b_k^{(k)}}{a_{kk}^{(k)}}$ اذ ان $j = 1..k$

الـCPU6 يقوم بحساب المعادلة $a_{3j}^{(k+1)} = a_{3j}^{(k)} - \frac{a_{3k}^{(k)} b_k^{(k)}}{a_{kk}^{(k)}}$ اذ ان $j = k+1..n$

.
.
.

الـCPU3n-3 يقوم بحساب $a_{in}^{(k+1)} = a_{in}^{(k)}$ ، $b_i^{(k+1)} = b_i^{(k)}$ اذ ان $i = 1..k$

الـCPU3n-2 يقوم بحساب $a_{nj}^{(k+1)} = 0$ ، $b_n^{(k+1)} = b_n^{(k)} - \frac{a_{nk}^{(k)} b_k^{(k)}}{a_{kk}^{(k)}}$ اذ ان $j = 1..k$

الـCPU3n-2 يقوم بحساب المعادلة $a_{nj}^{(k+1)} = a_{nj}^{(k)} - \frac{a_{nk}^{(k)} b_k^{(k)}}{a_{kk}^{(k)}}$ اذ ان $j = k+1..n$

الـCPU3n يقوم باستلام القيم من المعالجات ويعيد تكرار العمليات بعد زيادة واحد الى قيمة k.

4. مناقشة النتائج:

4.1 برمجة الطرق المتوازية:

كما ذكرنا سابقاً ان حاسبات من نوع MIMD غير متوفرة في بلادنا، فلذلك تم تنفيذ إجراء على حدى وحساب الوقت، ويكون الوقت الأكبر هو الوقت المعتمد في حساب تنفيذ الإجراءات المتوازية، وهذا ممكن اذ ان المعالج المستخدم في الحاسبات المتوازية هو المعالج نفسه المستخدم في الحاسبات الشخصية، كما توصل اليها الباحث Y.F. Fung et al في دراسته[15]، اذ اثبت انه بإمكان الطلاب استخدام حاسباتهم الشخصية لتنفيذ ومعالجة البرامج والخوارزميات المتوازية بسبب ارتفاع كلفة الحاسبات ذات المعالجات المتعددة [15].

وقد تمت برمجة الطرق المتوازية الثلاث وتنفيذها باستخدام لغة C++ على حاسبة PIV بمعالج 2GHz وباستخدام دالة Delay (120000)[1] لعدة أنظمة وكانت النتائج المستحصلة كالاتي:

الجدول (1): يمثل الوقت المحسوب في تنفيذ الطريقة العادية والطرائق المتوازية.

(1) تم استخدام هذه الدالة بسبب سرعة تنفيذ البرنامج اذ يكون الوقت المحسوب باجزاء الثواني او اقل، فتم استخدامها لتأخير عمل المعالج ليظهر الوقت المحسوب بالثواني.

سعة المصفوفة	الوقت المحسوب (s)			
	الطريقة العادية	الطريقة الاولى	الطريقة الثانية	الطريقة الثالثة
50×50	2.43	1.47	2.094	1.125
100×100	4.94	2.83	4.065	1.934
200×200	19.86	10.54	14.59	4.67

فعند حساب عامل التسريع Speed up بين الطريقة العادية والطرائق المتوازية الثلاث نحصل على ما يأتي:

$$\text{التسريع Sp} = \frac{\text{وقت التنفيذ لـM1 باستخدام معالج واحد (Ts)}}{\text{وقت التنفيذ لـM باستخدام n من المعالجات (Tp)}}$$

اذ ان (M) هي تطوير عن خوارزمية متسلسلة (M1) باستخدام (P) من المعالجات.

الجدول (2): يمثل عامل التسريع للطرائق المتوازية.

سعة المصفوفة	عامل التسريع		
	الطريقة الأولى	الطريقة الثانية	الطريقة الثالثة
50×50	1.6531	1.1605	2.1594
100×100	1.7456	1.2153	2.5543
200×200	1.8843	1.3612	4.2526

إذن كما نلاحظ ان عامل تسريع الطريقة الثانية اقل من الطريقتين الاولى والثالثة وان عامل التسريع في الطريقة الثالثة هو افضل من الطريقة الاولى، فضلاً عن ان عامل التسريع في الطريقتين الاولى والثالثة يزداد بنسبة ملحوظة عن الطريقة الثانية اذا زادت سعة المصفوفة.

5. مقارنة بين الطرائق المتوازية الثلاث:

من خلال النتائج التي ظهرت لنا يمكن أن نلاحظ فروقاً بين الطرائق يمكن تلخيصها بالجدول الآتي:

الجدول (3): الفروق بين الطرائق الثلاث المتوازية المقترحة لطريقة كاوس للحذف

	الطريقة الأولى	الطريقة الثانية	الطريقة الثالثة

.1	نحتاج إلى n من المعالجات اذ ان n هو عدد المعادلات	نحتاج إلى 6 من المعالجات مهما كان عدد المعادلات	نحتاج إلى 3n من المعالجات اذ ان n هو عدد المعادلات
.2	عندما تكون المسألة كبيرة يكون وقت التنفيذ أقل من الطريقة الثانية ولكن بعدد معالجات أكثر.	عندما تكون المسألة كبيرة تأخذ وقتاً كبيراً بالنسبة الى لطريقة الاولى والثالثة.	عندما تكون المسألة كبيرة يكون وقت التنفيذ أقل من الطريقة الاولى ولكن بعدد معالجات أكثر.
.3	تعد مكلفة للمسألة الكبيرة ولهذا يفضل استخدامها في المسائل الصغيرة.	غير مكلفة بالنسبة الى لمسائل الكبيرة لأن المسائل كافة تحتاج إلى 6 معالجات.	تعد مكلفة جداً للمسألة الكبيرة ولهذا يفضل استخدامها في المسائل الصغيرة.
.4	الحمل يكون على المعالجات بشكل متساوٍ تقريباً.	الحمل على المعالجات الثلاث الأولى متساوٍ والرابعة والخامسة متساويتان ايضاً.	الحمل يكون على المعالجات بشكل متساوٍ تقريباً.

المصادر

[1] الالوسي، د.أحمد صالح وعادل زينل البياتي (1989): "مقدمة في التحليل العددي"، مطبعة التعليم العالي في الموصل.

[2] اميل شكر الله، (2003)، "التحليل العددي التطبيقي، النظرية التطبيقات والطرق التقريبية"، جامعة المنوفية، مصر ، الطبعة الثانية.

[3] سيفي، د. علي محمد صادق و د. ابتسام كمال الدين (1986): "مباديء التحليل العددي"، مديرية دار الكتب للطباعة والنشر ،جامعة الموصل.

[4] عبد الحبيب عبدالله أحمد مرشد (2000): "تقصي خوارزميات عددية لحل المعادلات التفاضلية الاعتيادية الصلبة الملائمة للحاسبات المتوازية"، أطروحة دكتوراه، كلية العلوم، جامعة الموصل ، وزارة التعليم العالي والبحث العلمي.

[5] عماد فتاش، (2004) "الحاسبات المتوازية والخوارزميات المتوازية (دراسة)"، كلية العلوم، جامعة الملك سعود، المملكة العربية السعودية.

[6] عمر محمد التومي، احمد الشوشة، (2005) "التحليل العددي"، كتاب الكتروني، منتديات التقنية، www.tkne.net .

[7] يحيى قاسم إبراهيم القاضلي (2002م): "حل مسألة التخصيص باستخدام خوارزمية متوازية"، مشروع ماجستير ، كلية علوم الحاسبات والرياضيات، جامعة الموصل، وزارة التعليم العالي والبحث العلمي.

[8] Autar Kaw, (2007) "Introduction of Matrix Algebra", http://numericalmethods.eng.usf.edu.

[9] Bashir M. S., Khalaf and Khlil K.Abbo (2001): "Parallel Revised Simplex Method", Raf. Jour. Sci., Vol. 13, No. 1, pp. 51-60.

[10] Flynn M. J. (1972): "Some Computer Organization and their Effectiveness". IEEE Transaction on /computers, Vol. C-21, pp. 948-960.

[11] Ian Foster, (2002) "Designing and building Parallel Program", www.unix.ms.anl.gov/dbpp/text/node8.html.

[12] McDonough J. M., (2004) "Lectures in Basic Computational Numerical Analysis", University of Kentucky.

[13] Khalil, K. Aboo, (2006) "Parallel Newtonian Optimization without Hessian Approximation", Raf. J. of Comp. & Math's., Vol.3, No. 2, pp. 69-82

[14] Michael, Wasilewski (2004), "Parallel Gaussian Elimination", University of Waterloo.

[15] Y. F. Fung, et al; "Teaching Parallel Computing Concepts with A Desktop Computer" International Journal of Electrical Engineering Education Journal, Vol 41 Issue 2, April (2004), pp. 113-125.