مجله مهندسی مکانیک مدرس، اردیبهشت 1395، دوره 16، شماره 2، ص‌ص 309-318

ماهنامه علمی پژوهشی

مهندسی مکانیک مدرس

mme.modares.ac.ir

ارائه الگوریتم جدید توماس شطرنجی برای حل دستگاه معادلات سه‌قطری روی پردازنده‌گرافیکی

سید علیرضا ذوالفقاری[1*]، علی فؤادالدینی[2]

1- استادیار، گروه مهندسی مکانیک و مدیر گروه پژوهشی انرژی در ساختمان و آسایش حرارتی، دانشگاه بیرجند، بیرجند

2- دانشجوی کارشناسی ارشد، مهندسی مکانیک، دانشگاه بیرجند، بیرجند

* بیرجند، صندوق پستی 97175/376، zolfaghari@birjand.ac.ir

اطلاعات مقاله

مقاله پژوهشی کامل

دریافت: 10 آذر 1394

پذیرش: 18 دی 1394

ارائه در سایت: 26 بهمن 1394

کلید واژگان:

روش توماس شطرنجی

پردازنده گرافیکی همه منظوره

دستگاه معادلات سه‌قطری

الگوریتم کاهش متناوب موازی

چکیده

پردازنده گرافیکی همه منظوره کاربر را قادر می‌سازد تا از پردازنده گرافیکی برای مقاصد محاسباتی عمومی بهره بگیرد. استفاده از این نوع پردازنده‌ها موجب افزایش قابل توجهی در سرعت محاسبات عددی می‌شود. تحقیقات متعددی جهت بررسی مزیت استفاده از پردازنده گرافیکی در محاسبات از جمله بکارگیری آن برای حل دستگاه معادلات سه‌قطری صورت گرفته است. تمرکز اصلی تحقیقات مذکور، روی ارتقاء شیوه‌های بهره‌گیری از الگوریتم‌های موازی، نظیر کاهش متناوب و کاهش متناوب موازی بوده است. این الگوریتم‌ها با معماری پردازنده گرافیکی سازگارند، با این وجود پیچیدگی محاسباتی بالایی نسبت به الگوریتم توماس سری دارند و دارای محدودیت‌هایی در خصوص ابعاد دستگاه معادلات می‌باشند. بنابراین در تحقیق حاضر با توجه به مزایای الگوریتم توماس نسبت به الگوریتم‌های موازی، شیوه‌ای نوین با عنوان توماس شطرنجی جهت سازگار کردن الگوریتم توماس برای اجرا روی پردازنده گرافیکی ارائه شده است. این روش برای حل مسئله هدایت پایای دوبعدی استفاده شده و نتایج نشان‌دهنده افزایش دقت پاسخ نسبت به دو الگوریتم توماس و کاهش متناوب موازی می‌باشد. همچنین نتایج حاکی از آن است که روش جدید می‌تواند نسبت به الگوریتم توماس، بین 5.7 تا 22.2 افزایش سرعت محاسباتی را در پی داشته باشد. بعلاوه نتایج نشان‌می‌دهد که سرعت این روش به طور میانگین در حدود 2 برابر الگوریتم کاهش متناوب موازی می‌باشد. همچنین مشاهده شد که دسترسی غیرهم‌مکان به حافظه سراسری موجب حداقل و حداکثر کاهش سرعت 42.7 و 81.9 درصد به ترتیب برای اندازه شبکه 128×128 و 1024×1024 می‌شود.

Developing new Checkerboard Thomas algorithm for solving tridiagonal set of equations on GPU

Alireza Zolfaghari*, Ali Foadaddini

Department of Mechanical Engineering, University of Birjand, Birjand, Iran
* P.O.B. 97175/376, Birjand, Iran, zolfaghari@birjand.ac.ir

ARTICLE INFORMATION

Original Research Paper
Received 01 December 2015
Accepted 08 January 2015
Available Online 15 February 2016

Keywords:
Checkerboard Thomas method
GPGPU
tridiagonal set of equations
PCR method

ABSTRACT

General Purpose Graphics Processing Unite (GPGPU) allows the user to utilize GPU for general computing purposes. Using these processors can cause a great speedup in numerical calculations. Several studies have been performed to investigate the advantages of using the GPGPU in numerical calculations including solving tridiagonal set of equations. The main focus of the mentioned studies was on improving parallel methods, for example, CR and PCR algorithms. Although these algorithms are consistent with GPU architecture, they have higher arithmetic complexity compared with serial Thomas algorithm, they also have limitations in dimensions of the equations' set. Therefore, in the present study, according to the advantages of Thomas algorithm compared with the parallel algorithms, a novel method entitled checkerboard Thomas has been developed to accommodate Thomas algorithm for running on GPU. This method has been used for solving 2D steady heat conduction problem and the results show an increase in the solution precision compared to Thomas and PCR algorithms. Also, the results indicate that the new algorithm can cause computing to increase in speedup between 5.7 to 22.2x, compared with Thomas algorithm. Furthermore, results show that the new method is about 2x faster than PCR algorithm. It has also been seen that speed decrement for uncoalesced access to global memory is 42.7% minimum and 81.9% maximum for 128×128 and 1024×1024 grid size, respectively.

1- مقدمه

واحد پردازش گرافیکی[1] ابزاری تخصصی است که برای ارائه ماشینی تصاویر اجسام سه‌بعدی به صورت دو بعدی در رایانه‌های شخصی، کنسول‌های بازی و ... مورد استفاده قرار می‌گیرد. پردازنده گرافیکی همه منظوره[2]، نوعی

1- GPU(Graphics Processing Unit)

2- GPGPU(General Purpose Graphics Processing Unit)

برای ارجاع به این مقاله از عبارت ذیل استفاده نمایید:

Please cite this article using:

A. Zolfaghari, A. Foadaddini, Developing new Checkerboard Thomas algorithm for solving tridiagonal set of equations on GPU, *Modares Mechanical Engineering*, Vol. 16, No. 2, pp. 309-318, 2016 (in Persian)

پردازنده است که کاربر را قادر می‌سازد تا از پردازنده گرافیکی برای مقاصد محاسباتی عمومی نیز بهره بگیرد، معماری ویژه پردازنده گرافیکی به دلیل داشتن پردازنده‌های متعدد، امکان انجام تعداد زیادی محاسبه را به صورت موازی ممکن می‌کند و علاوه بر آن تاخیرات حافظه را پنهان می‌سازد.

استفاده از پردازنده گرافیکی همه منظوره برای انجام محاسبات عددی از سال 2001 آغاز شد و برای اولین بار مزیت بکارگیری آن در انجام عملیات فاکتورگیری ال- یو[1]در سال 2007 واضح گردید [1]، پس از آن تحقیقات متعددی جهت بررسی عملکرد پردازنده گرافیکی در افزایش سرعت[2] محاسبات صورت گرفت، در این میان روش‌های حل موازی دستگاه معادلات سه قطری در بسیاری از تحقیقات مورد توجه و تحلیل قرار گرفت، سنگوپتا و همکاران [2] در سال 2007 برای اولین بار الگوریتم کاهش متناوب[3] را روی پردازنده گرافیکی جهت مدل‌سازی جریان بکار گرفتند، ساخارنیخ [3] در سال 2009 الگوریتم توماس موازی[4] را جهت مدل‌سازی جریان سیالات بکار برد، وی حداکثر افزایش سرعت 11 برابری را در مقایسه با حل پردازنده مرکزی گزارش کرد، همچنین ژانگ و همکاران [4] روشی ترکیبی در کنار الگوریتم‌های توماس، کاهش متناوب، کاهش متناوب موازی[5] و دوبرابر سازی معکوس[6] پیشنهاد کردند، تحقیقات ایشان بر نقش عملیات زمانبر انتقال اطلاعات بین حافظه سراسری و حافظه میزبان در کاهش سرعت محاسبات صحه گذاشت، همچنین در تحقیقات ایشان وجود تداخل در دسترسی به حافظه اشتراکی[7] در الگوریتم کاهش متناوب و مزیت بهره‌گیری از الگوریتم‌های ترکیبی مورد بررسی قرار گرفت، گودک و همکاران [5] در سال 2011 یک الگوریتم کاهش متناوب که در آن تداخل دسترسی به حافظه اشتراکی[8] اتفاق نمی‌افتد ارائه کردند، الگوریتم کاهش متناوب در سال 2011 توسط دیویدسون و همکاران [6] به روش فشرده‌سازی حافظه ثبات[9] بهینه‌سازی شد، در همان سال ساخارنیخ [7] ترکیبی از الگوریتم کاهش متناوب موازی و توماس را ارائه کرد، روش مذکور توسط ژانگ و همکاران [8] مورد بررسی قرار گرفت و افزایش سرعت حدودا 2 برابری نسبت به الگوریتم کاهش متناوب گزارش شد، اگلوف [9] نمونه‌ای از الگوریتم کاهش متناوب موازی را برای حلگرهای تفاضل محدود جهت حل دستگاه معادلات سه قطری در ابعاد بزرگ ارائه نمود، وی و همکاران [10] با ارائه روشی جدید جهت سازگاری الگوریتم کاهش متناوب موازی با دستگاه معادلات بزرگ و ایجاد امکان دسترسی هم مکان[10] به حافظه سراسری[11] عملکرد حلگر ضمنی جهت متغیر[12] را برای حل معادلات هدایت حرارتی گذرا، بهبود بخشیدند، کیم و همکاران [11] با ارائه روشی مرکب از الگوریتم کاهش متناوب موازی موزاییکی[13] و توماس موازی جهت حل دستگاه معادلات بزرگ به افزایش سرعت 8 و 49 برابری نسبت به روش چند نخی[14] و سری با استفاده از حلگر ام کا ال[15] دست یافت، در سال 2012 اصفهانیان و همکاران [12] پیاده‌سازی طرح‌های ونو[16] را روی پردازنده گرافیکی مورد مطالعه قرار دادند، همچنین در سال 2013 اصفهانیان و همکاران [13] تاثیر نحوه دسترسی به حافظه سراسری را روی الگوریتم کاهش متناوب مورد مطالعه قرار دادند، ایشان به افزایش سرعت حل 1.9 تا 15.2 برابر در دو بعد و 6.4 تا 20.3 برابر در سه بعد دست یافتند، همچنین داریان و اصفهانیان [14] در سال 2014 افزایش سرعت 105 برابر را در حل معادلات اویلر به کمک پردازنده گرافیکی گزارش کردند.

همان‌گونه که ذکر شد، در تحقیقات پیشین تمرکز اصلی روی ارتقاء شیوه‌های بهره‌گیری از الگوریتم‌های موازی، جهت حل دستگاه معادلات سه قطری بوده است، الگوریتم‌های مذکور با وجود سازگاری بالا با ساختار پردازنده گرافیکی که موجب امکان بهره‌گیری مناسب از مزایای پردازنده گرافیکی می‌گردد، دارای پیچیدگی محاسباتی بسیار بیشتری نسبت به الگوریتم توماس هستند، بعلاوه در الگوریتم‌هایی نظیر کاهش متناوب و کاهش متناوب موازی محدودیت قابل توجهی پیرامون ابعاد دستگاه معادلات وجود دارد، در تحقیق حاضر شیوه‌ای نوین با عنوان توماس شطرنجی جهت بهره‌گیری از الگوریتم توماس در حلگر ADI برای اجرا روی پردازنده گرافیکی ارائه شده است، در این روش ضمن بهره‌گیری از مزایای الگوریتم توماس نسبت به الگوریتم‌های موازی از جمله سادگی محاسباتی، دفعات پایین دسترسی به حافظه و عدم وجود محدودیت در ابعاد دستگاه معادلات امکان فعال‌سازی نخ‌های محاسباتی کافی جهت بهره‌گیری از تمام ظرفیت پردازنده گرافیکی و پنهان ساختن تاخیرات حافظه فراهم شده است.

2- پردازنده گرافیکی همه منظوره

تفاوت اصلی پردازنده گرافیکی و واحد پردازش مرکزی[17] در تعداد هسته‌های پردازشگری است که عملیات محاسباتی و منطقی را در رایانه‌ها به عهده می‌گیرند، این تفاوت موجب شده که در هر یک از این دو نوع پردازنده، راهکارهای متفاوتی برای سرعت بخشیدن به انجام محاسبات بکار گرفته شود، در پردازنده مرکزی افزایش سرعت حافظه و پردازنده در کنار بکارگیری سازوکارهایی نظیر انجام محاسبات به صورت خط‌لوله‌ای[18] و یا استفاده از اجرای حدسی[19] بهره گرفته می‌شود، اما با توجه به ساختار متفاوت پردازنده گرافیکی و حضور تعداد زیادی پردازنده موسوم به هسته کودا[20] و امکان بکارگیری نخ‌های محاسباتی متعدد، می‌توان با استفاده از سازوکارهای متفاوتی به محاسبات سرعت بخشید، انجام محاسبات مستقل به صورت همزمان روی پردازنده‌های متعدد موجب افزایش حجم محاسبات در هر لحظه شده و استفاده از تعداد بالایی نخ‌های پردازشی امکان پنهان کردن تاخیرات حافظه[21] را فراهم ساخته است.

هر پردازنده گرافیکی از چندین مجتمع‌پردازشی[22] تشکیل شده است که مشخصات و امکانات آن وابسته به معماری و توانایی محاسباتی[23] (نسخه) آن پردازنده می‌باشد، هر مجتمع پردازشی به صورت عمده دارای چندین هسته کودا، انواع متفاوتی حافظه (سراسری، اشتراکی، بافتی[24]، ثبات و ...) و واحدی نظیر زمانبند[25] است که دستورالعمل‌ها را پیش از پردازش آماده‌سازی می‌کند، شکل1 نمایی کلی از معماری پردازنده گرافیکی را نشان می‌دهد، فلش‌های نشان داده شده در شکل، نحوه ارتباط میان میزبان و پردازنده‌های

1- LU factorization
2- Speed up
3- Cyclic Reduction(CR)
4- Parallel Thomas
5- Parallel Cyclic Reduction(PCR)
6- Recursive Doubling(RD)
7- Shared memory
8- Bank conflict
9- Register packing
10- Coalesced access
11- Global memory
12- Alternating Direction Implicit solver(ADI)
13- Tiled PCR
14- Multithreaded
15- MKL(Math Kernel Library)
16- WENO

17- CPU(Central processing unit)
18- Pipelining
19- Speculative execution
20- CUDA cores
21- Hiding memory latency
22- Multiprocessor
23- Compute capability
24- Texture
25- Schedulers

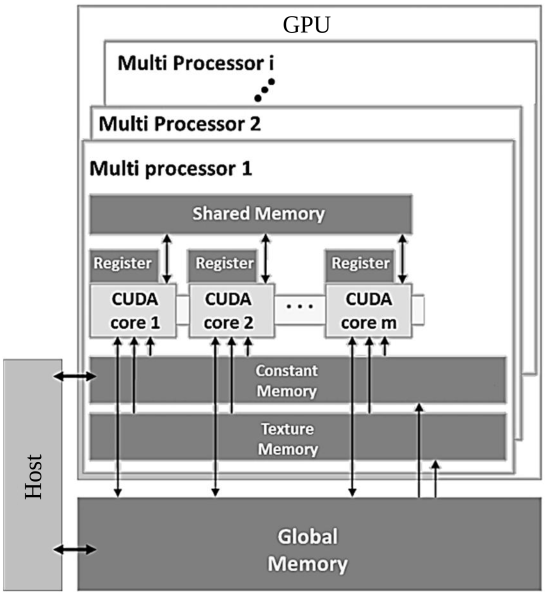

Fig. 1 GPU architecture

شکل 1 معماری پردازنده گرافیکی

کودا را با حافظه‌ها مشخص می‌کند. فلش یک طرفه به این معناست که پردازنده تنها قادر به خواندن حافظه است و فلش‌های دوطرفه به این معناست که حافظه قادر است هم حافظه را خوانده و هم بنویسد. همچنین حافظه‌های مختلف به لحاظ حجم، سرعت دسترسی، دامنه دسترسی و مدت زمان حفظ اطلاعات متفاوت می‌باشند.

پردازنده گرافیکی جهت انجام محاسبات موازی بسیار کارآمد است. این در حالی است که انجام محاسبات سری روی پردازنده مرکزی با سرعت بیشتری انجام می‌شود. بنابراین باید دستورالعمل‌ها را به دو بخش سری و موازی تقسیم کرد. بخش سری توسط واحد پردازش مرکزی (میزبان[1]) و بخش موازی (توابع کرنل[2]) توسط پردازنده گرافیکی (دستگاه[3]) به انجام می‌رسد. به این نحوه از انجام محاسبات، محاسبه ناهمگن[4] می‌گویند (شکل2). توابع مربوط به بخش موازی برنامه توسط میزبان فراخوان شده و روی دستگاه اجرا می‌گردند. جهت اجرای بخش موازی محاسبات، میان‌افزار کودا یک شبکه محاسباتی را به کاربر عرضه می‌کند که در آن مجتمع‌های پردازشی در قالب بلاک‌ها[5] و هسته‌های کودا در قالب نخ‌های محاسباتی[6] قرار دارند و وی را قادر می‌سازد تا بدون توجه به تعداد واقعی مجتمع‌های پردازشی و هسته‌های کودا، عملیات محاسباتی را روی هزاران بلاک و نخ محاسباتی توزیع نماید. در حقیقت هر نخ محاسباتی جزئی از یک پردازش است که امکان اجرای مستقل را دارا می‌باشد. نخ‌های محاسباتی در قالب بلاک‌ها دسته بندی می‌شوند. ابعاد بلاک‌ها، تعداد و آرایش آن‌ها در شبکه محاسباتی هنگام فراخوان تابع کرنل مشخص می‌شود. در هنگام اجرای دستورالعمل‌ها در پردازنده گرافیکی، در هر مرتبه تعداد مشخصی از بلاک‌ها جهت انجام محاسبات مربوطه در هر یک از مجتمع‌های پردازشی قرار می‌گیرند. نخ‌های محاسباتی در هر بلاک پس از قرارگیری در مجتمع‌های پردازشی به دسته‌های 32 تایی با نام وارپ[7] تقسیم می‌شوند. ویژگی اساسی

1- Host
2- Kernel
3- Device
4- Heterogeneous Computation
5- Block
6- Thread
7- Warp

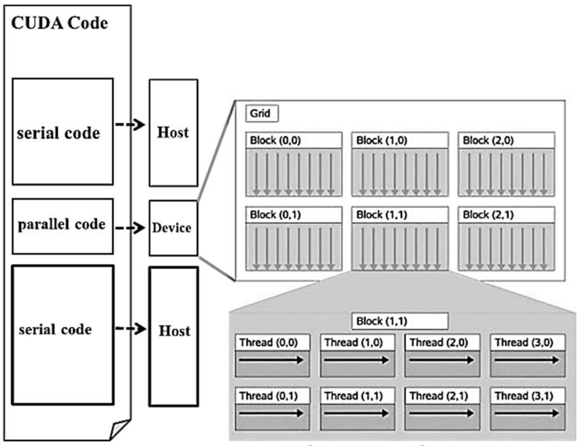

Fig. 2 Heterogeneous computing by CPU and GPU

شکل 2 محاسبات ناهمگن به وسیله پردازنده مرکزی و پردازنده گرافیکی

یک وارپ این است که تمام نخ‌های موجود در آن دستورالعملی یکسان با داده‌هایی متفاوت را انجام می‌دهند. برای مثال عملیات جمع برداری را در نظر بگیرید که در آن عملیات جمع برای درایه‌های مختلف دو بردار صورت می‌گیرد. دستورالعمل‌های هر وارپ توسط زمانبند آماده اجرا می‌شود. زمانبندی وارپ برای انجام عملیات جدید تنها در صورتی انجام می‌پذیرد که عملیات قبل توسط نخ‌های محاسباتی موجود در آن به پایان رسیده باشد. در این مدت، زمانبند به آماده‌سازی وارپ‌های دیگر می‌پردازد. این عمل موجب پنهان شدن تاخیرات مربوط به عملیات زمانبری نظیر دسترسی به حافظه سراسری می‌گردد.

از مسائل مهمی که می‌بایست در بهره‌گیری از پردازنده گرافیکی به آن توجه شود این است که میزان مزیت بهره‌گیری از پردازنده گرافیکی علاوه بر مشخصات فنی پردازنده تحت تاثیر عوامل فراوانی از جمله روش محاسبات، بکارگیری حافظه‌های مختلف، نحوه دسترسی به حافظه و ... قرار دارد. با رعایت ملاحظاتی از این دست می‌توان محاسبات را به شکلی بهینه و با سرعت بالا روی پردازنده گرافیکی همه منظوره به انجام رساند.

3- بررسی الگوریتم‌های رایج حل دستگاه معادلات سه‌قطری

3-1- الگوریتم توماس

الگوریتم توماس از سریع‌ترین روش‌های حل سری دستگاه معادلات سه قطری می‌باشد که از روش حذفی گوس گرفته شده است. دستگاه معادلات (1) را در نظر بگیرید:

$$\begin{bmatrix} b_1 & c_1 & & & & \\ a_2 & b_2 & c_2 & & 0 & \\ & a_3 & b_3 & c_3 & & \\ & & . & . & . & \\ & & . & . & . & \\ & 0 & & a_{n-1} & b_{n-1} & c_{n-1} \\ & & & & a_n & b_n \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ . \\ . \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ . \\ . \\ d_{n-1} \\ d_n \end{bmatrix} \quad (1)$$

که شامل ماتریس ضرایب، ماتریس مجهولات و ماتریس دست راست معادلات می‌باشد. الگوریتم توماس این دستگاه معادلات سه قطری را در دو مرحله حل می‌کند: کاهش پیشرو و جای‌گذاری پسرو. در مرحله کاهش پیشرو، قطر پایین ماتریس سه قطری حذف شده و مقادیر متناظر برای قطر میانی و بالا محاسبه می‌گردند. محاسبه مقادیر مذکور از طریق روابط (2) و (3) صورت می‌گیرد:

$$\acute{c}_1 = \frac{c_1}{b_1}\,, \acute{c}_l = \frac{c_l}{b_l - c_{l-1}^{\prime} a_l},\ \ l = 2,3,\dots,n-1 \tag{2}$$

$$\acute{d}_1 = \frac{d_1}{b_1}\,, \acute{d}_l = \frac{d_l - \acute{d}_l a_l}{b_l - c_{l-1}^{\prime} a_l},\ \ l = 2,3,\dots,n-1 \tag{3}$$

در مرحله جایگذاری پسرو با به‌دست آمدن آخرین مجهول و جای‌گذاری آن در معادلات بالاتر به صورت پی در پی مجهولات مسئله از رابطه (4) به دست می‌آید:

$$x_n = \acute{d}_n\,, x_l = \acute{d}_l - c_{l-1}^{\prime} x_{l+1} \tag{4}$$

الگوریتم توماس یک الگوریتم کاملا سری است چراکه مقدار $\acute{c}_l$ ، $\acute{d}_l$ و x_l به ترتیب به مقادیر محاسبه شده $c_{l-1}^{\prime}$ ، $d_{l-1}^{\prime}$ و x_{l+1} بستگی دارد. در این الگوریتم به ازای به دست آمدن هر مجهول 8 محاسبه صورت می‌گیرد و به تعداد دو برابر مجهولات، مراحل محاسباتی طی می‌شود.

3-2- الگوریتم کاهش متناوب

الگوریتم کاهش متناوب یکی از الگوریتم‌های حل موازی دستگاه معادلات سه قطری است که امکان بهره‌گیری از مزایای پردازنده گرافیکی همه منظوره را فراهم می‌کند. الگوریتم کاهش متناوب از دو بخش تشکیل شده است: کاهش پیشرو و جایگذاری پسرو. در بخش کاهش پیشرو در هر مرحله دستگاه معادلات موجود به دستگاهی با نیمی از مجهولات و معادلات مرحله قبل کاسته می‌شود و این روند تا زمان به دست آوردن دستگاهی با دو معادله و دو مجهول ادامه می‌یابد. در بخش جای‌گذاری پسرو در هر مرحله با قرار دادن مجهولات به‌دست آمده از مرحله قبل در دستگاه معادلات، سایر مجهولات به دست می‌آیند. در شکل 3 روند حل یک دستگاه معادله 8 مجهولی به وسیله الگوریتم کاهش متناوب نشان داده شده است.

در هر مرحله از بخش کاهش پیشرو ضرایب معادله جدید از رابطه (5) به دست می‌آیند:

$$\begin{aligned} &\acute{a}_l = -a_{l-1}k_1\,, \acute{b}_l = b_l - c_{l-1}k_1 - a_{l+1}k_2 \\ &\acute{c}_l = -c_{l+1}k_2\,, \acute{d}_l = d_l - d_{l-1}k_1 - d_{l+1}k_2 \\ &k_1 = \frac{a_l}{b_{l-1}}, \qquad k_2 = \frac{c_l}{b_{l+1}} \end{aligned} \tag{5}$$

در هر مرحله از جای‌گذاری پسرو نیز مجهول مدنظر به وسیله رابطه (6) از معلومات مرحله قبل خود به دست می‌آید:

$$x_l = \frac{\acute{d}_l - \acute{a}_l x_{l-1} - \acute{c}_l x_{l+1}}{\acute{b}_l} \tag{6}$$

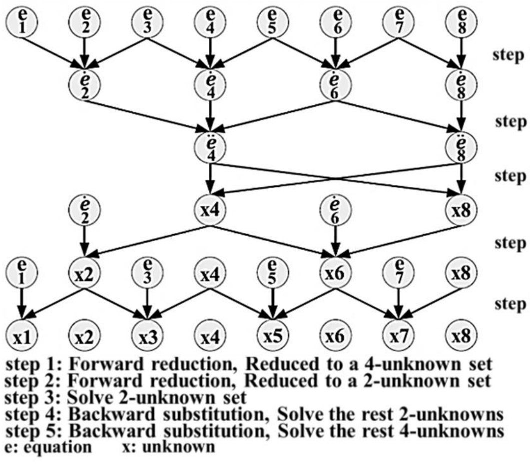

Fig. 3 Solution steps for a 8-unknown set of equations by CR algorithm

شکل 3 مراحل حل یک دستگاه معادله 8 مجهولی به وسیله الگوریتم کاهش متناوب (CR)

همان‌طور که مشخص است، برخلاف الگوریتم توماس، در الگوریتم کاهش متناوب در هر مرحله مقدار ضرایب یک معادله جدید به معادلات قبل و بعد آن وابستگی ندارد، به این ترتیب می‌توان محاسبه ضرایب برای هر معادله را به صورت مستقل و موازی با سایر معادلات انجام داد.

3-3- الگوریتم کاهش متناوب موازی

الگوریتم کاهش متناوب موازی الگوریتمی نظیر کاهش متناوب است، با این تفاوت که تنها دارای بخش کاهش پیشرو می‌باشد. همچنین معادلات مرتبط با بخش کاهش پیشرو در الگوریتم کاهش متناوب موازی کاملا مطابق با الگوریتم کاهش متناوب می‌باشد. در این الگوریتم در هر مرحله از کاهش پیشرو، هر دستگاه معادله موجود به دو دستگاه با اندازه‌ای نصف اندازه دستگاه قبلی تقسیم می‌شود. نهایتا با ایجاد چند دستگاه معادله دو مجهولی پاسخ دستگاه معادله به دست می‌آید. در شکل 4 روند حل یک دستگاه معادله (8) مجهولی به وسیله الگوریتم کاهش متناوب موازی نشان داده شده است.

3-4- مقایسه الگوریتم‌های سری و موازی

به طورکلی جهت بهره‌گیری مناسب از مزایای پردازنده گرافیکی در محاسبات عددی می‌بایست روش‌هایی بکارگیری شوند که قابلیت انجام عملیات محاسباتی به صورت موازی را فراهم نمایند. به عبارت دیگر باید امکان انجام چندین عملیات محاسباتی به صورت مستقل روی پردازنده‌های متعدد فراهم باشد. در الگوریتم توماس به دلایلی که پیشتر بیان شد، امکان انجام محاسبات به صورت موازی و مستقل از هم وجود ندارد. در حل دستگاه معادلات سه قطری به وسیله پردازنده گرافیکی، دو روش کاهش متناوب و کاهش متناوب موازی نسبت به الگوریتم توماس مزیت دارند. در این دو روش امکان انجام محاسبات مستقل از هم فراهم شده که می‌توان آنها را به صورتی موازی و روی نخ‌های پردازشی متعدد پیگیری کرد. در جدول 1 سه روش حل مذکور به لحاظ مراحل، تعداد عملیات محاسباتی و دفعات دسترسی به حافظه مورد مقایسه قرار گرفته‌اند.

همان‌طور که در جدول 1 ارائه شده است دو الگوریتم کاهش متناوب و کاهش متناوب موازی، در مقایسه با الگوریتم توماس دارای تعداد مراحل

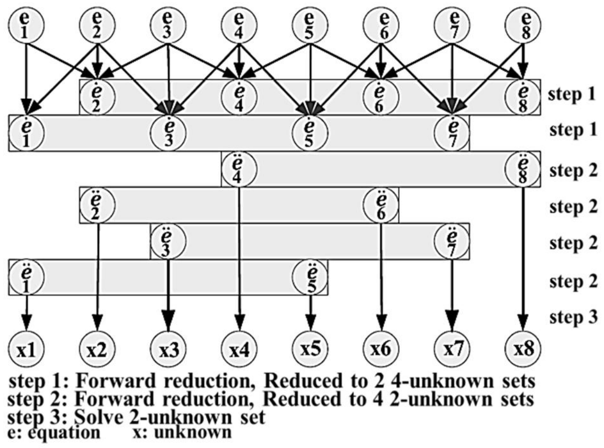

Fig.4 solution steps for a 8-unknown set of equations by PCR algorithm

شکل 4 مراحل حل یک دستگاه معادله 8 مجهولی به وسیله الگوریتم کاهش متناوب موازی (PCR)

جدول 1 مقایسه الگوریتم‌های رایج حل دستگاه معادلات سه قطری

Table 1 comparison of conventional algorithms for solving thridiagonal system of equations

نام الگوریتم	تعداد مراحل	تعداد عملیات محاسباتی	دفعات دسترسی به حافظه
توماس	$2n$	$8n$	$12n$
کاهش متناوب	$2\log_2 n - 1$	$17n$	$23n$
کاهش متناوب موازی	$\log_2 n$	$12\log_2 n$	$16n\log_2 n$

سری کمتری هستند، این ویژگی روش‌های مذکور را جهت انجام محاسبات روی پردازنده گرافیکی مناسب می‌سازد، با این حال تعداد عملیات محاسباتی و دفعات دسترسی به حافظه در این روش‌ها برای بدست آوردن مجهولات بسیار بیشتر از الگوریتم توماس می‌باشد. بعلاوه، الگوریتم CR و PCR تنها قادر به حل دستگاه معادلاتی با 2^n مجهول می‌باشد حال آن‌که الگوریتم توماس محدودیتی از جهت اندازه دستگاه معادلات مدنظر ندارد. در تحقیق حاضر با توجه به مزیت های ذکر شده برای الگوریتم توماس نسبت به دو الگوریتم دیگر، سعی شده تا با ارائه روشی جدید امکان بهره‌گیری از الگوریتم مذکور در حلگر ADI بگونه‌ای سازگار با معماری پردازنده گرافیکی فراهم گردد.

4- ارائه روش جدید توماس شطرنجی[1]

همان‌گونه که در بخش قبل ذکر شد، سری بودن الگوریتم توماس مهمترین مانع بهره‌گیری از این الگوریتم روی پردازنده گرافیکی می‌باشد که استفاده از مزایای آن از جمله سادگی محاسباتی و دفعات پایین دسترسی به حافظه را محدود می‌سازد. جهت رفع این مانع می‌توان با تقسیم دستگاه معادله اولیه به چندین دستگاه کوچک‌تر و حل تکراری آن، امکان مشارکت پردازنده‌های متعدد در روند حل را فراهم کرد. به این ترتیب روشی جدید با عنوان توماس شطرنجی ارائه می‌گردد. در این روش هر دستگاه معادله اولیه به nop دستگاه معادله با اندازه dop تقسیم می‌شود و حل این دستگاه معادلات به صورت تکراری و توسط nop نخ محاسباتی صورت می‌گیرد. برای مثال مطابق شکل 5 شبکه محاسباتی یک بعدی، متشکل از n مجهول مدنظر است (n مضرب صحیحی از 4 می‌باشد). به جای حل مجهولات از طریق حل یک دستگاه معادله n مجهولی، می‌توان از طریق حل تکراری $nop = n/4$ دستگاه معادله چهار مجهولی به پاسخ رسید. در این حل تکراری، برای شروع، مقداری اولیه به همه مجهولات داده شده و در هر تکرار دستگاه معادلات همواره براساس مقادیر معلوم شبکه و معادله انفصال حاکم تشکیل خواهند شد. برای مثال دستگاه معادله اول شامل 4 مجهول مربوط به نقاط 1،2،3 و 4 شبکه است. این دستگاه معادله براساس معادله انفصال حاکم و براساس مقادیر معلوم مربوط به نقطه مرزی و نقطه 5 شبکه محاسباتی به دست آمده است. مقدار معلوم نقطه 5 در حقیقت مقداریست که در تکرار پیشین حل حاصل شده است. به همین ترتیب مقادیر معلوم مورد استفاده در تشکیل دستگاه معادلات دوم نیز مقادیر مربوط به نقاط 4 و 9 شبکه هستند که در تکرار قبل به دست آمده‌اند. حل تکراری به دو صورت قابل انجام است. مطابق حالت (الف) شکل 5 می‌توان در هر تکرار هر nop دستگاه معادله را به صورت همزمان و به وسیله nop نخ محاسباتی حل کرد. به همین شکل در این روش تمام مقادیر جدید بر اساس تکرار قبل به دست می‌آیند. حالت (ب) در شکل5 روشی از حل را نشان می‌دهد که درآن هر تکرار در دو مرحله صورت می‌گیرد، به‌طوری که در هر مرحله دستگاه معادلات به صورت شطرنجی یا یکی در میان حل می‌شوند. در این روش نخ‌های پردازشی در مرحله دوم از نتایج مرحله اول همان تکرار به عنوان مقادیر معلوم جهت تشکیل دستگاه معادلات بهره می‌برند. با توجه به این خاصیت، سرعت همگرایی حالت (ب) از حالت (الف) بیشتر خواهد بود چرا که این مقادیر نسبت به مقادیر تکرار قبل اصلاح شده‌تر می‌باشند و این باعث می‌شود که پاسخ دستگاه معادله به مقادیر حل نهایی شبکه نزدیکتر شود. البته در این صورت n نیز باید به قدر کافی بزرگ باشد تا تعداد دستگاه معادله کافی برای مشغول نگهداشتن تمام هسته‌های پردازنده و پنهان کردن تاخیرات حافظه فراهم شود. تعداد تقسیمات دستگاه معادلات (nop) از جمله پارامترهای است که عملکرد الگوریتم توماس شطرنجی را تحت تاثیر قرار می‌دهد. افزایش nop موجب می‌شود که نخ‌های محاسباتی بیشتری جهت حل دستگاه معادلات فعال شوند و از طرفی تعداد تکرارها جهت رسیدن به همگرایی را افزایش می‌دهد. هر چه تعداد این نخ‌ها بیشتر باشد هسته‌های پردازشی بیشتری بکار گرفته شده و تاخیرات حافظه نیز بهتر پنهان می‌شود. با این حال افزایش بیش از حد تعداد تکرارها، ضمن افزایش nop می‌تواند از سرعت همگرایی الگوریتم بکاهد. فواید و کاربرد الگوریتم توماس شطرنجی در حل تکراری دستگاه معادلات سه قطری زمانی نمود بیشتری خواهد داشت که از آن در یک حلگر ADI بهره گرفته شود. روش توماس شطرنجی نسبت به روش توماس موازی به کار رفته توسط ساخارنیخ [3] سریع تر است چرا که دستگاه معادلات سه‌قطری مستقل بیشتری را جهت حل در نخ‌های محاسباتی ایجاد می‌کند. بعلاوه در این روش برخلاف روش ترکیبی کاهش متناوب موازی موزاییکی و توماس موازی [11] از الگوریتم کاهش متناوب موازی جهت ایجاد دستگاه معادلات سه قطری متعدد استفاده نشده است. این موضوع موجب شده است که روش توماس شطرنجی محدودیت‌های روش کاهش متناوب موازی از جهت اندازه دستگاه معادلات و پیچیدگی محاسباتی را نداشته باشد.

5- حل یک مسئله نمونه با بکارگیری روش توماس شطرنجی

5-1- تعریف یک مسئله نمونه با حلگر ADI

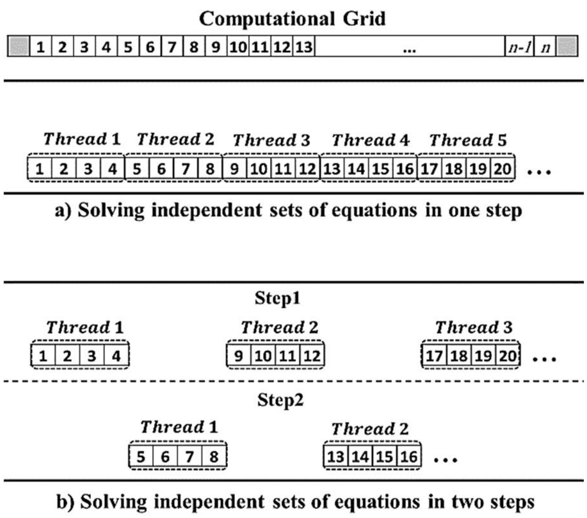

Fig.5 Two different iterative methods for solving a set of equations by Thomas algorithm

شکل 5 دو روش مختلف حل تکراری دستگاه معادلات به وسیله الگوریتم توماس

1- Checkerboard Thomas(CH-Thomas)

حلگر ADI یک روش تحلیل عددی است که برای حل معادلات مشتق جزئی بیضوی، سهموی و هذلولی در فضای چند بعدی به‌کار می‌رود. این موضوع موجب شده است که این روش به طور گسترده در علوم پایه و مهندسی به کار گرفته شود. در روش ADI متناسب با ابعاد مسئله هر مرحله از محاسبات به چند زیر- مرحله تقسیم می‌شود که در آن معادله دیفرانسیل مربوطه در یک راستا به صورت ضمنی حل شده، در حالی که اطلاعات مسئله در سایر جهت‌ها به صورت صریح مورد استفاده قرار می‌گیرد. در نتیجه، این روش بدون پیش شرط پایدار است. یک خاصیت جالب توجه این روش این است که در آن برای حل معادلات در هر زیر- مرحله می‌توان از الگوریتم حل ماتریس سه قطری بهره برد.

حل معادله هدایت حرارتی پایا به عنوان مسئله‌ای ساده، مثال خوبی برای بهره‌گیری از حلگر ADI است. معادله هدایت حرارت پایا در دو بعد به صورت معادله (7) می‌باشد:

$$\frac{\partial}{\partial x}\left(\mathrm{k}\frac{\partial T}{\partial x}\right)+\frac{\partial}{\partial y}\left(\mathrm{k}\frac{\partial T}{\partial y}\right)=0 \qquad (7)$$

استخراج معادله انفصال به روش حجم محدود و روی شبکه نشان داده شده در شکل 6 صورت گرفته و نتیجه به این صورت است:

$$\begin{aligned}&\left(2\frac{\mathrm{k}\Delta y}{\Delta x}+2\frac{\mathrm{k}\Delta y}{\Delta x}\right)T(i,j)\\&=\frac{\mathrm{k}\Delta y}{\Delta x}T(i-1,j)+\frac{\mathrm{k}\Delta y}{\Delta x}T(i+1,j)+\frac{\mathrm{k}\Delta x}{\Delta y}T(i,j+1)\\&+\frac{\mathrm{k}\Delta x}{\Delta y}T(i,j-1)\end{aligned} \qquad (8)$$

در صورتی که معادله انفصال مطابق رابطه (8) جهت حل مقادیر مجهول در یک شبکه دو بعدی مورد استفاده قرار گیرد یک دستگاه معادله 5 قطری تشکیل می‌شود. با حل این دستگاه معادله پاسخ مورد نظر به صورت مستقیم به دست خواهد آمد. اما در حلگر ADI حل معادله انفصالی (8) به صورت تکراری انجام می‌شود. در مرحله اول معادله انفصال به صورت ضمنی در جهت X حل می‌شود. در این مرحله اطلاعات شبکه در جهت Y به صورت صریح مورد استفاده قرار می‌گیرند. رابطه (9) شکل معادله انفصالی را در این وضعیت نشان می‌دهد:

$$\begin{aligned}&\left(2\frac{\mathrm{k}\Delta y}{\Delta x}+2\frac{\mathrm{k}\Delta y}{\Delta x}\right)T^{K}(i,j)\\&=\frac{\mathrm{k}\Delta y}{\Delta x}T^{K}(i-1,j)+\frac{\mathrm{k}\Delta y}{\Delta x}T^{K}(i+1,j)\\&+\frac{\mathrm{k}\Delta x}{\Delta y}T^{K-1}(i,j+1)\\&+\frac{\mathrm{k}\Delta x}{\Delta y}T^{K-1}(i,j-1)\end{aligned} \qquad (9)$$

در رابطه (9) بالا نویس K و K-1 به ترتیب اشاره به تکرار جدید و تکرار قبل دارند. مقادیر تکرار قبل به‌عنوان مقدار معلوم در تشکیل دستگاه معادلات شرکت داده می‌شوند. به این ترتیب m (تعداد تقسیمات شبکه در راستای عرض) دستگاه معادله سه قطری تشکیل می‌شود که شبکه حل را در راستای عرض جاروب می‌کنند. حل این دستگاه معادلات می‌تواند با یکی از روش‌های گفته شده از جمله توماس شطرنجی انجام پذیرد. در مرحله دوم معادله انفصال به صورت ضمنی در جهت Y حل می‌شود. در این مرحله اطلاعات شبکه در جهت X به صورت صریح مورد استفاده قرار می‌گیرند. رابطه (10) شکل معادله انفصالی را در این وضعیت نشان می‌دهد:

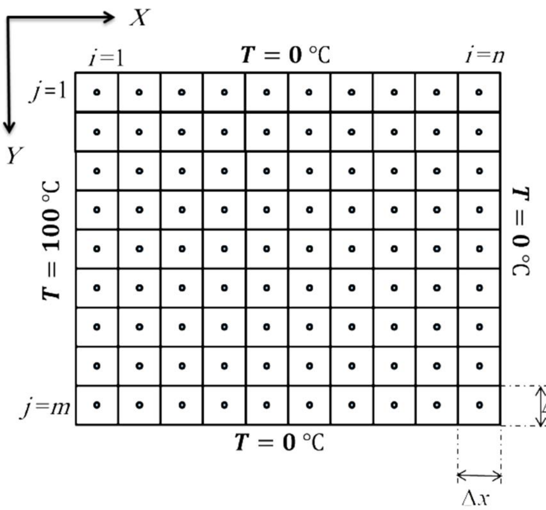

Fig.6 Grid and boundary conditions for the present 2*D* heat conduction problem

شکل 6 شبکه و شرایط مرزی برای مسئله هدایت دو بعدی تحقیق حاضر

$$\begin{aligned}&\left(2\frac{\mathrm{k}\Delta y}{\Delta x}+2\frac{\mathrm{k}\Delta y}{\Delta x}\right)T^{K}(i,j)\\&=\frac{\mathrm{k}\Delta y}{\Delta x}T^{K-1}(i-1,\mathrm{j})+\frac{\mathrm{k}\Delta y}{\Delta x}T^{K-1}(i+1,j)\\&+\frac{\mathrm{k}\Delta x}{\Delta y}T^{K}(i,j+1)\\&+\frac{\mathrm{k}\Delta x}{\Delta y}T^{K}(i,j-1)\end{aligned} \qquad (10)$$

2-5- پیاده سازی حلگر با به‌کارگیری الگوریتم توماس شطرنجی

برای پیاده سازی حلگر ADI در روش توماس شطرنجی کلیه اطلاعات به صورت آرایه‌های یک بعدی (شکل7) در حافظه سراسری ذخیره شده و تا انتها در این حافظه باقی می‌مانند. جهت حل صریح معادله دیفرانسیل در هر جهت میدان حل در آن راستا به *nop* قسمت تقسیم می‌شود. این *nop* قسمت به صورت یکی در میان به وسیله *nop*/2 نخ محاسباتی در دو مرحله حل می‌شوند. سپس تقسیمات شبکه در راستای دیگر صورت گرفته و معادله دیفرانسیل در این جهت به صورت صریح حل می‌گردد. به عنوان مثال در شکل8 مراحل حل در دو جهت X و Y برای شبکه‌ای به اندازه 8×8 و $nop=2$ نشان داده شده است. اعداد داخل سلول‌های شبکه نشان‌دهنده ترتیب ذخیره مقادیر آنها در آرایه یک‌بعدی نشان داده شده در شکل7 می‌باشد. همان‌طور که در شکل مذکور ارائه شده است، حل در هر جهت به صورت دو مرحله‌ای صورت می‌گیرد. در یک مرحله تمام سلول‌های خاکستری و در مرحله بعد تمام سلول‌های سفید به وسیله نخ‌های محاسباتی اختصاصی حل می‌شوند. لازم به ذکر است که هر یک از مراحل مذکور در قالب یک کرنل مستقل انجام شده و پس از هر یک، سمت راست دستگاه معادلات به روز رسانی می‌شود.

دسترسی بهینه به حافظه اهمیت ویژه‌ای در به‌کارگیری مناسب پردازنده گرافیکی دارد. بنابراین در پیاده‌سازی روش توماس شطرنجی، استفاده بهینه از حافظه سراسری می‌بایست مورد توجه قرار گیرد. موضوع مهمی که هنگام بهره‌گیری از حافظه سراسری، سرعت محاسبات را به شدت تحت تاثیر قرار می‌دهد، دسترسی هم مکان و یا غیرمکان به حافظه سراسری است. این مسئله به شیوه دسترسی نخ‌های محاسباتی یک وارپ به حافظه سراسری مرتبط است. در پردازنده‌های گرافیکی نظیر پردازنده بکار رفته در تحقیق

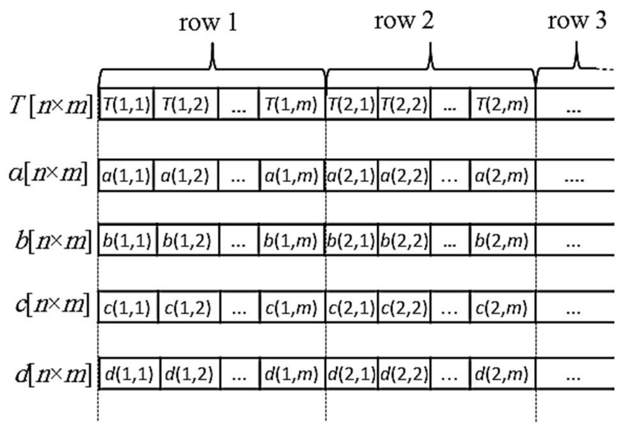

Fig.7 D arrays storage arrangement in 1D arrays

شکل 7 دستورالعمل ذخیره آرایه‌های دو بعدی به صورت آرایه‌های یک بعدی

حاضر، هر 128 بایت پی‌درپی از حافظه سراسری با یک بار مراجعه توسط وارپ در دسترس قرار می‌گیرد. دسترسی به حافظه در صورتی هم‌مکان است که تمام اطلاعات مورد نیاز نخ‌های محاسباتی، در این 128 بایت از حافظه قرار داشته و نیازی به مراجعات متعدد به حافظه وجود نداشته باشد. دسترسی هم‌مکان به حافظه سراسری در صورتی میسر می‌گردد که اطلاعات فراخوانده شده توسط نخ‌های محاسباتی یک وارپ به صورتی نزدیک به هم در حافظه ذخیره شده باشند. هر چه این فاصله بیشتر باشد مدت زمان انجام عملیات افزایش خواهد یافت.

بهینه‌سازی دسترسی به حافظه سراسری در روش توماس شطرنجی نیز می‌تواند کمک قابل توجهی به افزایش سرعت حل نماید. همان‌گونه که در شکل 8 نشان داده شده، در هنگام حل معادله در جهت Y میان مکان‌های دسترسی نخ‌های محاسباتی مجاور به اندازه 1 درایه فاصله وجود دارد. اما در وضعیت حل معادله دیفرانسیل در جهت X بین مکان‌های دسترسی نخ‌های محاسباتی مجاور به اندازه 11 درایه فاصله وجود دارد. طبق مطالب گفته شده، فاصله زیاد بین محل ذخیره اطلاعات در حافظه سراسری، تعداد دفعات مراجعه به این حافظه توسط وارپ را افزایش داده و موجب کاهش سرعت عملیات محاسباتی می‌گردد.

جهت رفع این مشکل می‌توان پس از انجام حل در جهت Y، ماتریس مقادیر شبکه را ترانهاده کرد و معادله دیفرانسیل را مجددا روی شبکه جدید در جهت Y حل نمود. عملیات ترانهاده کردن ماتریس توسط پردازنده گرافیکی عملیاتی زمانبر است با این حال می‌توان با بهره‌گیری از توابع بهینه از زمان محاسبات کاست. مجموعه‌ای از انواع این توابع در بسته توسعه نرم‌افزاری کودا[1] قرار دارد. در تحقیق حاضر از تابعی استفاد شده که در آن دسترسی به حافظه سراسری هم‌مکان می‌باشد.

همچنین لازم به ذکر است که شبکه محاسباتی کودا جهت حل دستگاه معادلات آرایه‌ای یک بعدی متشکل از بلاک‌هایی با 128 نخ محاسباتی است و تعداد بلاک‌ها با توجه به اندازه شبکه حل تعیین شده است.

5-3- معرفی سیستم محاسباتی

در تحقیق حاضر از پردازنده گرافیکی جی- تی- ایکس 750 ام استفاده شده است که دارای قابلیت محاسباتی (نسخه) 3.0 می‌باشد. مشخصات این پردازنده در جدول2 ارائه شده است. نسخه کودای به‌کار رفته نیز 6.5 بوده

1- CUDA Software Development Kit (CUDA SDK)

جدول 2 مشخصات پردازنده به کار رفته در تحقیق حاضر

Table 2 characterstics of the GPU which have been used in the present research

مشخصه	مقدار
تعداد مجتمع پردازشی	2
تعداد هسته کودا در هر مجتمع پردازشی	192
بیشترین میزان حافظه اشتراکی در هر مجتمع پردازشی	48 kB
بیشترین تعداد نخ‌های محاسباتی در هر بلاک	1024
بیشترین تعداد نخ‌های محاسباتی ساکن در هر مجتمع پردازشی	2048
بیشترین تعداد وارپ ساکن در هر مجتمع پردازشی	64
بیشترین تعداد بلاک ساکن در هر مجتمع پردازشی	16

است. همچنین پردازنده مرکزی بکار رفته اینتل کر- آی- سون 4500 یو می‌باشد که در زمره پردازنده‌های هم‌رده با پردازنده گرافیکی به‌کار رفته قرار می‌گیرد.

6- نتایج

در تحقیق حاضر روشی نوین تحت عنوان توماس شطرنجی برای حل معادلات دیفرانسیل به کمک پردازنده گرافیکی ارائه شده است. این روش براساس دستورالعمل بخش 5-2 پیاده‌سازی شده و عملکرد آن از حیث دقت پاسخ، روند تغییرات باقی‌مانده‌ها و سرعت بخشی به انجام محاسبات با الگوریتم PCR و توماس (پردازنده مرکزی) مقایسه شده است. همچنین میزان تاثیر دسترسی غیرهم‌مکان به حافظه سراسری مورد بررسی قرار گرفته است.

لازم به ذکر است که در تحقیق حاضر حلگر PCR مورد استفاده براساس روش وی و همکاران [10] پیاده‌سازی شده است. در این حلگر جهت سرعت بخشی به انجام محاسبات از حافظه اشتراکی بهره گرفته شده و بعلاوه مشکل تداخل دسترسی به حافظه اشتراکی دسترسی غیرهم‌مکان به حافظه سراسری مرتفع شده است.

شکل 9 مقدار دمای به دست آمده در نقطه مرکزی دامنه حل را برحسب تکرار برای حلگرهای مختلف نشان می‌دهد. اندازه شبکه حل به کار رفته در این مقایسه 512×512 می‌باشد. مقدار پاسخ دقیق مسئله نیز روی نمودار مشخص شده است. همان‌گونه که مشاهده می‌شود هنگامی که از روش توماس شطرنجی استفاده شده، حل در تکرار بیشتری نسبت به دو الگوریتم PCR و توماس همگرا می‌شود. مقدار پاسخ به دست آمده در نقطه مرکزی نیز در روش توماس شطرنجی به مقدار حل دقیق نزدیک‌تر می‌باشد. همچنین همان‌طور که مشاهده می‌شود با افزایش مقدار *nop* حل نهایی به حل دقیق نزدیک‌تر شده و تعداد تکرارها تا رسیدن به همگرایی، افزایش می‌یابد.

شکل 10 روند تغییرات باقی‌مانده‌ها را در شبکه‌ای به اندازه 512×512 برای حلگرهای مختلف نشان می‌دهد. همان‌گونه که مشاهده می‌شود روند تغییرات باقی‌مانده‌ها در روش توماس شطرنجی نسبت به دو الگوریتم توماس و PCR بسیار یکنواخت‌تر است. همچنین، در شکل11 افزایش سرعت ناشی از بهره‌گیری روش توماس شطرنجی (برای *nop* متفاوت) و الگوریتم PCR نسبت به الگوریتم توماس نمایش داده شده است. افزایش سرعت معیاریست برای سنجش میزان بهبود روش‌های مذکور به لحاظ زمانی نسبت به یک روش مبنا (که در این‌جا الگوریتم توماس (پردازنده مرکزی) می‌باشد و مقدار آن از رابطه (11) به دست می‌آید:

$$\text{Speed up} = \frac{t_{\text{CPU}}}{t} \qquad (11)$$

که در آن t_{CPU} مدت زمان حل مسئله توسط الگوریتم توماس (پردازنده

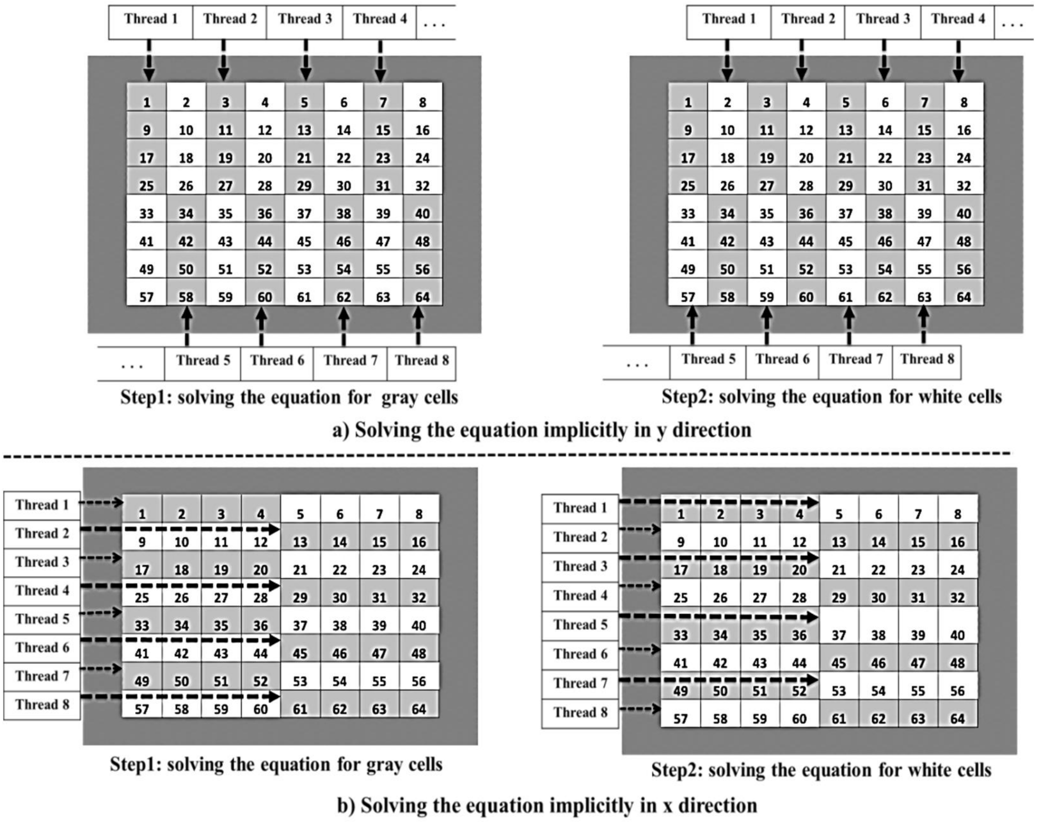

Fig.8 Solving differential equation implicitly in x and y direction by ADI solver

شکل 8 حل معادله دیفرانسیل به صورت صریح در جهت X و Y به وسیله حلگر ADI

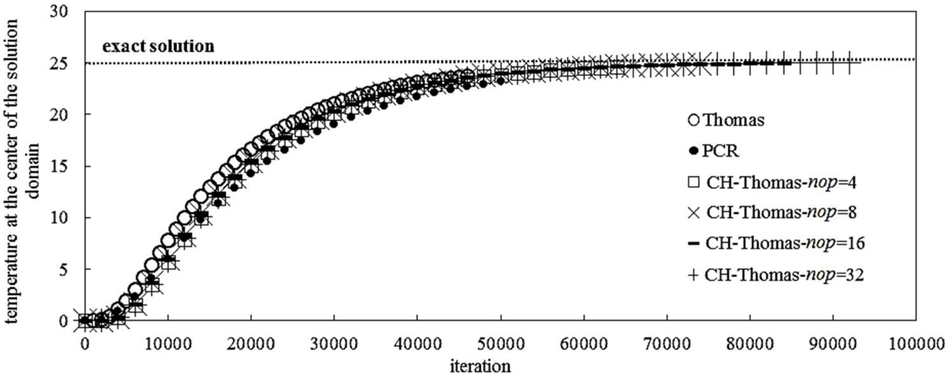

Fig.9 temperature at the center of the solution domain VS iteration number for Thomas, PCR and CH-Thomas (Gridsize:512×512)

شکل 9 تغییرات دما در مرکز دامنه حل برحسب تکرار برای الگوریتم توماس، PCR و روش توماس شطرنجی (اندازه شبکه حل: 512×512)

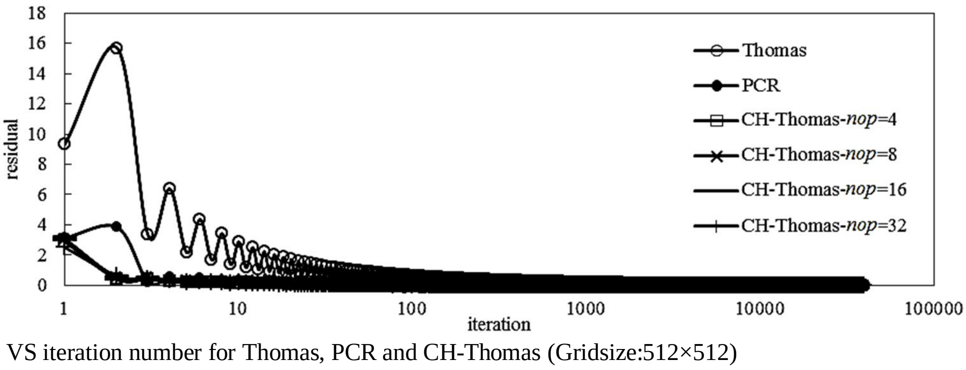

Fig. 10 Residual VS iteration number for Thomas, PCR and CH-Thomas (Gridsize:512×512)

شکل 10 باقی‌مانده برحسب تکرار برای الگوریتم توماس، PCR و روش توماس شطرنجی (اندازه شبکه حل: 512×512)

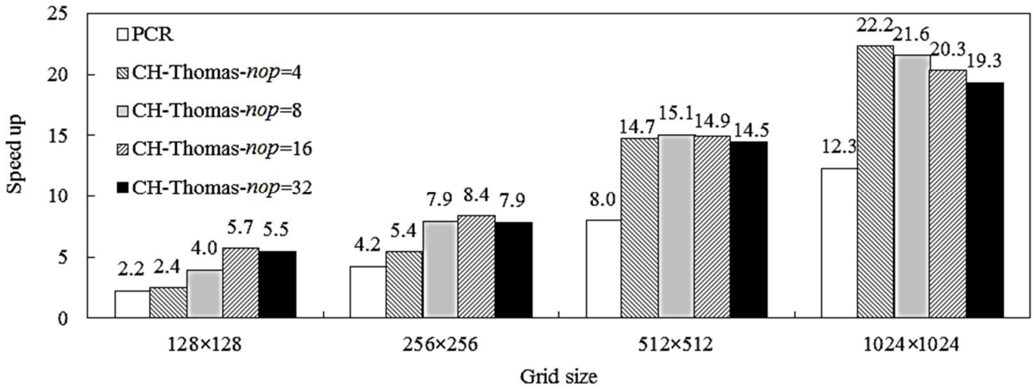

Fig. **11** Speed up for PCR and CH-Thomas (with respect to Thomas algorithm)

شکل 11 افزایش سرعت برای الگوریتم PCR و روش توماس شطرنجی (نسبت به الگوریتم توماس)

مرکزی) و t مدت زمان حل مسئله به وسیله روش مدنظر می‌باشد. همان‌گونه که در شکل11 مشاهده می‌شود پارامتر *nop* تاثیر قابل ملاحظه‌ای بر عملکرد روش توماس شطرنجی داشته است. برای مثال در شبکه‌ای به اندازه 128×128 با افزایش *nop* از 4 به 16، سرعت حلگر به طور قابل ملاحظه‌ای افزایش یافته است. همان‌گونه که در بخش 4 گفته شد، بهبود به وجود آمده در سرعت به این دلیل است که با افزایش *nop* تعداد دستگاه معادلات و در نتیجه تعداد نخ‌های محاسباتی فعال افزایش می‌یابد. افزایش نخ‌های محاسباتی امکان درگیر کردن تمام هسته‌های پردازنده، پنهان ساختن تاخیرات حافظه و در نتیجه افزایش سرعت را فراهم می‌سازد. با این حال افزایش *nop* از 16 به 32 به دلیل افزایش قابل ملاحظه تعداد تکرارهای مورد نیاز جهت همگرا شدن حل، موجب کاهش سرعت همگرایی می‌شود. نکته مهم دیگر این است که در روش توماس شطرنجی، در اندازه شبکه بزرگتر سرعت بیشینه در *nop* کمتری نسبت به اندازه شبکه کوچکتر اتفاق می‌افتد. به‌عنوان مثال بیشینه سرعت در شبکه محاسباتی با اندازه 128×128 در *nop*=16 اتفاق می‌افتد، در حالی که بیشینه سرعت در شبکه محاسباتی با اندازه 1024×1024 در *nop*=4 اتفاق افتاده است. مشاهده مذکور به این صورت قابل توضیح است که در شبکه‌های با اندازه بزرگتر در *nop* کمتری نخ محاسباتی لازم جهت بکارگیری تمام هسته‌های کودا و پنهان ساختن تاخیرات حافظه فراهم می‌شود. همچنین همان‌گونه که مشاهده می‌شود میزان افزایش سرعت در روش توماس شطرنجی به مراتب بیشتر از روش PCR است. همان‌گونه که ارائه شده است در *nop* مناسب نسبت سرعت روش توماس شطرنجی به الگوریتم PCR حدود 2 برابر می‌باشد.

در شکل 12 کاهش سرعت ناشی از دسترسی غیر هم‌مکان به حافظه سراسری در روش توماس شطرنجی برای *nop*=8 نشان داده شده است. همان‌گونه که مشاهده می‌شود با افزایش اندازه شبکه درصد کاهش سرعت، افزایش یافته است و از حدود 43 درصد در اندازه شبکه 128×128 به حدود 82 درصد در اندازه شبکه 1024×1024 رسیده است.

7- نتیجه‌گیری

در تحقیق حاضر با توجه به مزایای الگوریتم توماس نسبت به الگوریتم‌های ویژه حل موازی نظیر CR و PCR، مزیت بهره‌گیری از این الگوریتم جهت استفاده در حلگر ADI برای اجرا روی پردازنده گرافیکی مورد بررسی قرار گرفت. برای این منظور روشی جدید با عنوان توماس شطرنجی ارائه شده و مورد مطالعه قرار گرفت. روش مذکور برای حل مسئله هدایت پایای دوبعدی

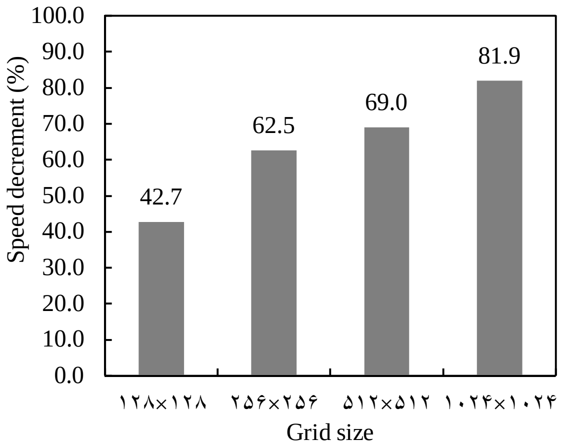

Fig. 12 Speed decrement caused by uncoalesced global memory access (nop=8)

شکل 12 کاهش سرعت ناشی از دسترسی غیرهم‌مکان به حافظه سراسری (nop=8)

به‌کار رفته و به لحاظ افزایش سرعت با الگوریتم توماس (پردازنده مرکزی) و PCR مورد مقایسه قرار گرفت. نتایج نشان می‌دهد که استفاده از روش، توماس شطرنجی موجب افزایش دقت پاسخ نهایی نسبت به استفاده از دو الگوریتم توماس و PCR می‌گردد و دقت پاسخ با افزایش پارامتر *nop* افزایش می‌یابد. بعلاوه مشخص گردید که در این روش روند تغییرات باقی‌مانده‌ها برحسب تکرار نسبت به دو الگوریتم توماس و PCR یکنواخت‌تر است. نتایج حل نشان داد که مزیت استفاده از این روش به میزان قابل توجهی به پارامتر *nop* وابسته است. با توجه به معماری پردازنده گرافیکی در اندازه شبکه‌های مختلف، بیشینه سرعت در *nop*های متفاوتی اتفاق می‌افتد. برای مثال در شبکه‌ای به اندازه 128×128 بیشینه سرعت در *nop*=16 و در شبکه‌ای به اندازه 1024×1024 بیشینه سرعت در *nop*=4 اتفاق افتاده است. همچنین مشاهده شد که بکارگیری روش توماس شطرنجی در حلگر ADI افزایش سرعت قابل توجهی را نسبت به الگوریتم توماس ایجاد می‌کند. این افزایش سرعت با بالا رفتن اندازه شبکه محاسباتی بیشتر شده است. در شبکه‌ای به اندازه 128×128 روش توماس شطرنجی نسبت به الگوریتم توماس 5.7 برابر افزایش سرعت ایجاد کرده است. این در حالی است که استفاده از روش توماس شطرنجی در شبکه ای به اندازه 1024×1024 منجر به افزایش سرعت 22.2 برابری نسبت به الگوریتم توماس شده است. بعلاوه سرعت روش توماس

شطرنجی در حدود 2 برابر الگوریتم PCR می‌باشد. همچنین تاثیر دسترسی غیرهم‌مکان به حافظه سراسری مورد بررسی قرار گرفت. نتایج نشان داد که دسترسی غیرهم‌مکان منجر به کاهش سرعت قابل توجهی در روش توماس شطرنجی می‌گردد. این کاهش سرعت با افزایش اندازه شبکه حل رابطه مستقیم دارد. در *nop*=8 دسترسی غیرهم‌مکان به حافظه سراسری منجر به کاهش سرعت 42.7 درصدی در شبکه‌ای به اندازه 128×128 می‌شود و با افزایش اندازه شبکه مسئله به 1024×1024 این مقدار به 81.9 درصد افزایش می‌یابد.

نهایتا می‌توان نتیجه گرفت که بکارگیری روش توماس شطرنجی در صورت تنظیم درست پارامتر *nop* می‌تواند نقش موثری در بهره‌گیری مناسب از مزایای پردازنده گرافیکی جهت حل معادلات دیفرانسیل به وسیله حلگر ADI داشته باشد. این روش نسبت به الگوریتم PCR عملکرد بهتری به لحاظ افزایش سرعت از خود نشان داده است در حالی که محدودیت‌های این الگوریتم از حیث اندازه دستگاه معادلات را دارا نمی‌باشد. به‌علاوه ایده توماس شطرنجی امکان این را فراهم آورده که بتوان با افزایش *nop* و کوچکتر کردن اندازه دستگاه معادلات امکان استفاده از حافظه اشتراکی را فراهم آورد با این حال سودمندی این روش از حیث افزایش سرعت حل می‌بایست مورد بررسی دقیق‌تر قرار بگیرد.

8- فهرست علائم

a	قطر پایین در ماتریس سه قطری
b	قطر اصلی در ماتریس سه قطری
c	قطر بالا در ماتریس سه قطری
d	بردار مقادیر معلوم
dop	اندازه دستگاه معادلات پس از انجام تقسیمات در روش توماس شطرنجی (بدون بعد)
dx	طول سلول‌های شبکه (m)
dy	عرض سلول‌های شبکه (m)
i	شمارنده سلول شبکه در راستای محور طول
j	شمارنده سلول شبکه در راستای محور عرض
k	ضریب هدایت حرارت ($Wm^{\circ}C^{-1}$)
n	تعداد تقسیمات شبکه در راستای محور طول
nop	تعداد تقسیمات دستگاه معادله در روش توماس شطرنجی (بدون بعد)
m	تعداد تقسیمات شبکه در راستای محور عرض
T	دما (°C)
t	مدت زمان انجام حل مسئله
t_{CPU}	مدت زمان انجام حل مسئله به وسیله پردازنده مرکزی
X	محور طول (بدون بعد)
Y	محور عرض (بدون بعد)

علائم یونانی

Δx	طول سلول‌های شبکه (m)
Δy	عرض سلول‌های شبکه (m)

9- مراجع

[1] P. Du, R. Weber, P. Luszczek, S. Tomov, G. Peterson, J. Dongarra, From CUDA to OpenCL: Towards a performance-portable solution for multi-platform GPU programming, *Parallel Computing*, Vol. 38, No. 8, pp. 391-407, 2012.

[2] S. Sengupta, M. Harris, Y. Zhang, J. D. Owens, Scan primitives for GPU computing, *Proceedings of the 22nd ACM SIGGRAPH/EUROGRAPHICS symposium on Graphics hardware*, San Diego, California, 2007.

[3] N. Sakharnykh, Tridiagonal solvers on the GPU and applications to fluid simulation, *NVIDIA GPU Technology Conference*, San Jose, California, USA, 2009.

[4] Y. Zhang, J. Cohen, J. D. Owens, Fast tridiagonal solvers on the GPU, *ACM Sigplan Notices*, Vol. 45, No. 5, pp. 127-136, 2010 .

[5] D. Göddeke, R. Strzodka, Cyclic reduction tridiagonal solvers on GPUs applied to mixed-precision multigrid, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 22, No. 1, pp. 22-32, 2011.

[6] A. Davidson, J. D. Owens, Register packing for cyclic reduction: a case study, *Proceedings of the fourth workshop on general purpose processing on graphics processing units*, Newport Beach, California, USA, 2011.

[7] N. Sakharnykh, Efficient tridiagonal solvers for ADI methods and fluid simulation, *NVIDIA GPU Technology Conference*, San Jose, California, USA, 2010.

[8] Daniel Egloff, *Pricing financial derivatives with high performance finite difference solvers on GPUs*, Wen-mei W. Hwu (Eds.), *GPU Computing Gems Jade Edition*, Burlington: Morgan Kaufmann, 2011.

[9] Yao Zhang, Jonathan Cohen, Andrew A. Davidson, and John D. Owens, *A Hybrid Method for Solving Tridiagonal Systems on the GPU*, Wen-mei W. Hwu (Eds.), *GPU Computing Gems Jade Edition*, Burlington: Morgan Kaufmann, 2011.

[10] Z. Wei, B. Jang, Y. Zhang, Y. Jia, Parallelizing alternating direction implicit solver on GPUs, *Procedia Computer Science*, Vol. 18, pp. 389-398, 2013.

[11] H.-S. Kim, S. Wu, L.-w. Chang, W.-m. W. Hwu ,A scalable tridiagonal solver for gpus, *International Conference on Parallel Processing (ICPP)*, Taipei, Taiwan, 2011.

[12] V. Esfahanian, H. M. Darian, S. I. Gohari, Assessment of WENO schemes for numerical simulation of some hyperbolic equations using GPU, *Computers & Fluids*, Vol. 80, pp. 260-268, 2013.

[13] V. Esfahanian, B. Baghapour, M. Torabzadeh, H. Chizari, An efficient GPU implementation of cyclic reduction solver for high-order compressible viscous flow simulations, *Computers & Fluids*, Vol. 92, pp. 160-171, 2014.

[14] H. M. Darian ,V. Esfahanian, Assessment of WENO schemes for multi-dimensional Euler equations using GPU, *International Journal for Numerical Methods in Fluids*, Vol. 76, No. 12, pp. 961-981, 2014 .

[15] J. V. Beck, N. Wright, A. Haji-Sheikh, K. D. Cole, D. Amos, Conduction in rectangular plates with boundary temperatures specified, *Heat Mass Transfer*, Vol. 51, pp. 4676-4690, 2008.