Amirkabir Journal of Mechanical Engineering

Amirkabir J. Mech. Eng., 53(special issue 3) (2021) 461-464
DOI: 10.22060/mej.2019.16178.6295

Developing an alternating direction explicit-implicit domain-decomposition approach to solve heat transfer equation on graphics processing unit

A. Foadaddini[1], S. A. Zolfaghari1[*], H. Mahmoodi Darian[2]

[1] Department of Mechanical Engineering, University of Birjand, Birjand, Iran
[2] School of Engineering Science, College of Engineering, University of Tehran, Tehran, Iran

ABSTRACT: In the present study, a new alternating direction explicit-implicit domain decomposition approach is proposed by combining the alternating direction implicit method with the explicit-implicit domain decomposition method. The method is used for solving the two-dimensional conduction heat transfer equation on a graphics processing unit. In this method, an explicit numerical scheme is used to predict values at the inner boundaries, and an implicit scheme based on the alternating direction implicit method is used to solve the sub-domains. Then, an implicit scheme is used to correct the values on the inner boundaries. Numerical experiments are done to investigate the accuracy and speed of the method. The results show that the present method can achieve a speedup of 1.3 to 2.6 times compared to the alternating direction implicit method. Increasing the number of subdomains increases the speed and decreases the accuracy of the method. Although numerical experiments show high stability of the present method, its error is higher than the alternating direction implicit method. Furthermore, the results show that the present method is more advantageous to problems with coarse grids, such that by increasing the grid size from 256 × 256 to 512 × 512, the speedup decreases from 2.4 to 1.7.

Review History:

Received: Apr. 21,2019
Revised: Jul. 20,2019
Accepted: Sep. 02, 2019
Available Online: Oct. 09, 2019

Keywords:

Computational Fluid Dynamics
Parallel Processing
Graphics Processing Unit
Alternating Direction Implicit Method
Corrected Explicit-Implicit Domain Decomposition Algorithm

1- Introduction

The two-dimensional heat conduction equation is one of the most commonly used equations in the field of mechanical engineering. Due to the high computational complexity and huge computational cost in long-term and real-scale problems, there is a pressing need to develop fast and accurate GPU-accelerated solvers for this equation. Due to their special architecture, GPUs become more effective when the given problem can be decomposed to many tasks performing the same operation on multiple data simultaneously. Regarding the methods of time advancement, such a condition occurs when explicit numerical schemes are used for solving Partial Differential Equations (PDEs). So, many researchers have developed GPU-accelerated solvers based on explicit numerical schemes [1, 2]. The Alternating Direction Implicit (ADI) method is a simple method for dealing with this problem. In ADI method, solving the tridiagonal matrix equation is the main building block of the algorithm. So, for the efficient implementation of the ADI solver on GPU, the solution method for the tridiagonal matrix equation is the key part.

For solving tri-diagonal systems on GPU, serial algorithms like Thomas [3, 4] and parallel algorithms like CR and PCR [5] have been proposed. The Thomas algorithm has low computational complexity and it is easy to implement. But because of the serial nature of the algorithm, computation related to each tri-diagonal system is mapped to one thread which leads to a low occupancy, especially in small-scale problems. Unlike the Thomas algorithm, in CR and PCR algorithms, each equation is mapped to one thread. So, the number of resident warp is increased and this leads to higher occupancy and efficient use of resources. But parallel algorithms suffer from high computational complexity because of the high number of mathematical operations per equation and communications between threads.

In the present study, a Corrected Explicit-Implicit Domain Decomposition (CEIDD) approach is proposed to reduce the size of the tridiagonal system of equations in the ADI method. In this method in each time step values at some points in the domain are predicted using an explicit scheme. By doing this, each tridiagonal system is partitioned into many smaller systems and this allows for partitioning the workloads. Explicit-implicit domain decomposition is a non-iterative and non-overlapping domain decomposition method so it is computationally and computationally efficient. In 2002. Du et

*Corresponding author's email: alireza.zolfaghari@yahoo.com

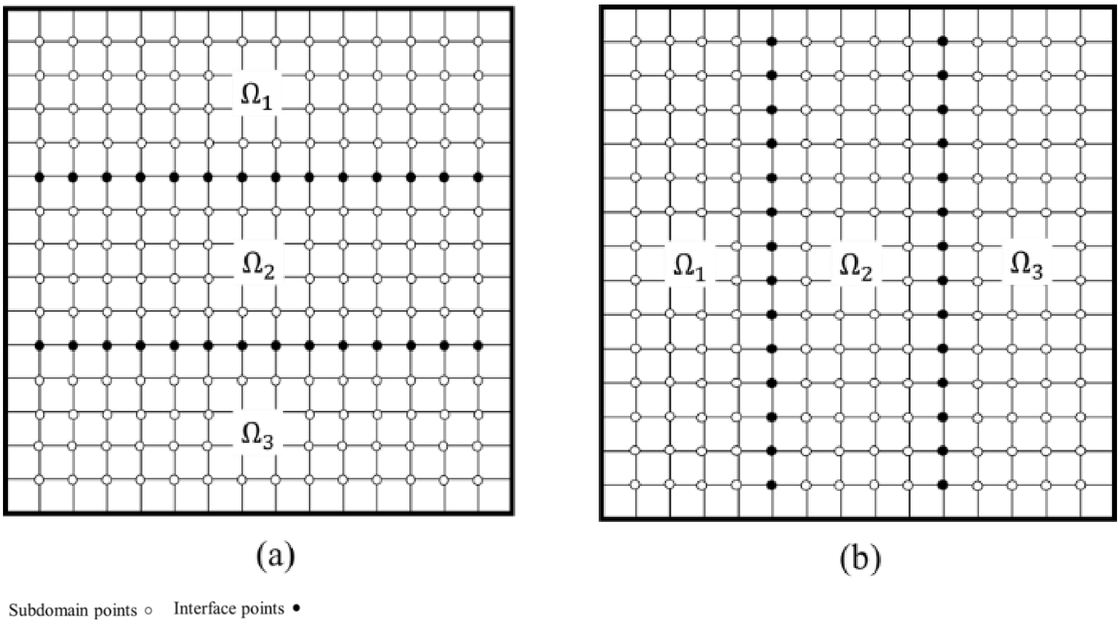


Fig. 1. Domain decomposition in X and Y directions in ADI-CEIDD

al. [6] proposed an EIDD method that can compute the values on the inner interfaces by either a high-order explicit scheme or multistep explicit scheme. The method is conditionally stable. Sun and Zhuang [7] added a further step to EIDD method to replace the values from the explicit scheme on interfaces with values computed by an implicit scheme. This method is unconditionally stable. More related to the present study, Du and Liang [8] combined EIDD algorithm with the splitting technique and proposed S-DDM method. In this method, the interface values are computed using local multilevel schemes and the splitting implicit scheme is utilized to compute the interior values of the subdomains. S-DDM is conditionally stable but using an efficient local multilevel scheme at interface points relax the stability condition [9].

In the present study, a new alternating direction explicit-implicit domain decomposition approach is proposed by combining the ADI method with the CEIDD domain decomposition method.

2- Methodology

The ADI-CEIDD method consists of two steps: *y*-sweep and *x*-sweep. In the *y*-sweep step, the domain is decomposed to *nos* subdomains in *Y* direction (Fig. 1(a)). Then the solution for interface points is predicted using an explicit scheme and a combination of values from two previous time levels. Next, the heat conduction equation is solved implicitly in *Y* direction and explicitly in *X* direction in the subdomains. Finally, the values in boundary points are corrected using an implicit scheme. In *x*-sweep, the domain is decomposed to *nos* subdomains in *X* direction (Fig. 1(b)) and the equation is solved implicitly in *X* direction and explicitly in *Y* direction. In ADI-CEIDD the number of independent systems of equations is several times the ADI method. So, the number of active threads is increased and this can lead to efficient use of GPU resources.

3- Results

Fig. 2 shows the L2-norm error for ADI-CEIDD method. The results show that decomposing the domain in ADI-CEIDD has increased the error compared to the ADI method. The minimal effect of the parameter *nos* has occurred in. As an overall trend, the error increases by increasing the λ. But in *nos*=4, 8, 16 the error for λ=0.5 is greater than the error for λ=1. Also, the results indicate the high stability of ADI-CEIDD method. For further optimization of the algorithm, coalesced and uncoalesced versions are investigated. Results show that the coalesced version is more efficient especially for big-scale problems.

Speedup of the coalesced version of ADI-CEIDD method versus the ADI method is presented in Fig. 3.

As the results show, the ADI-CEIDD can achieve a speedup of 1.3 to 2.6 times compared to the ADI method by increasing the occupancy. By increasing the number of subdomains from 2 to 32, the speed of the proposed method is increased up to 1.6 times. Furthermore, the results show that the ADI-CEIDD method is more advantageous to problems with coarse grids, such that by increasing the grid size from 256 × 256 to 512 × 512, the speedup decreases from 2.4 to 1.7.

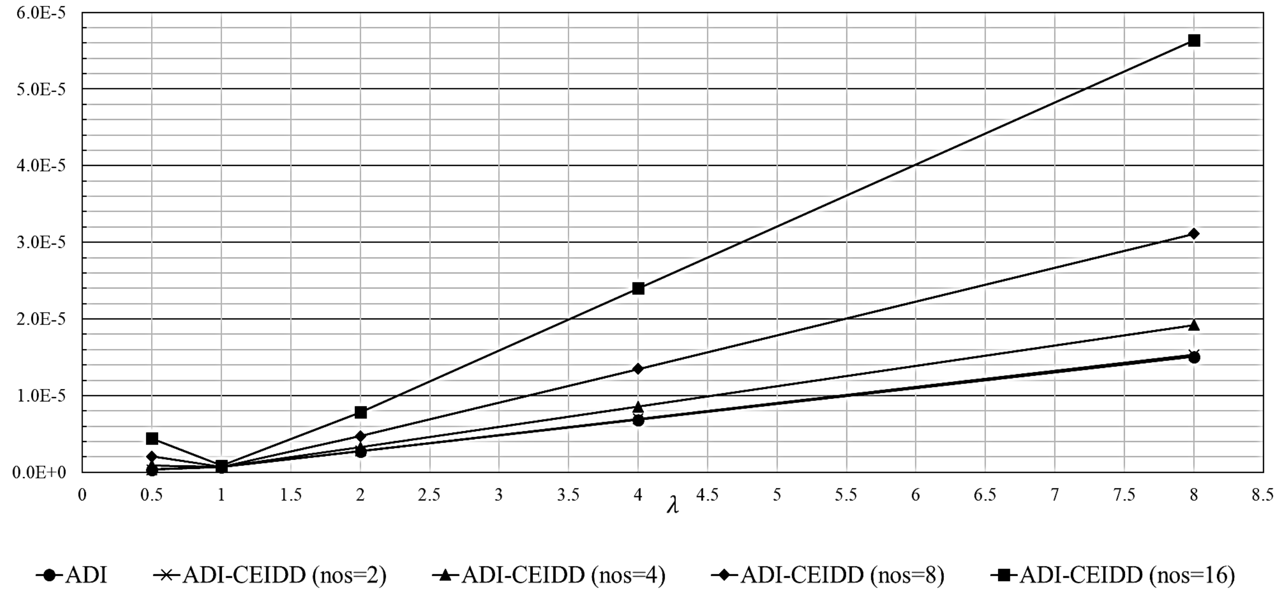


Fig. 2. L_2-norm error for a 256×256 grid

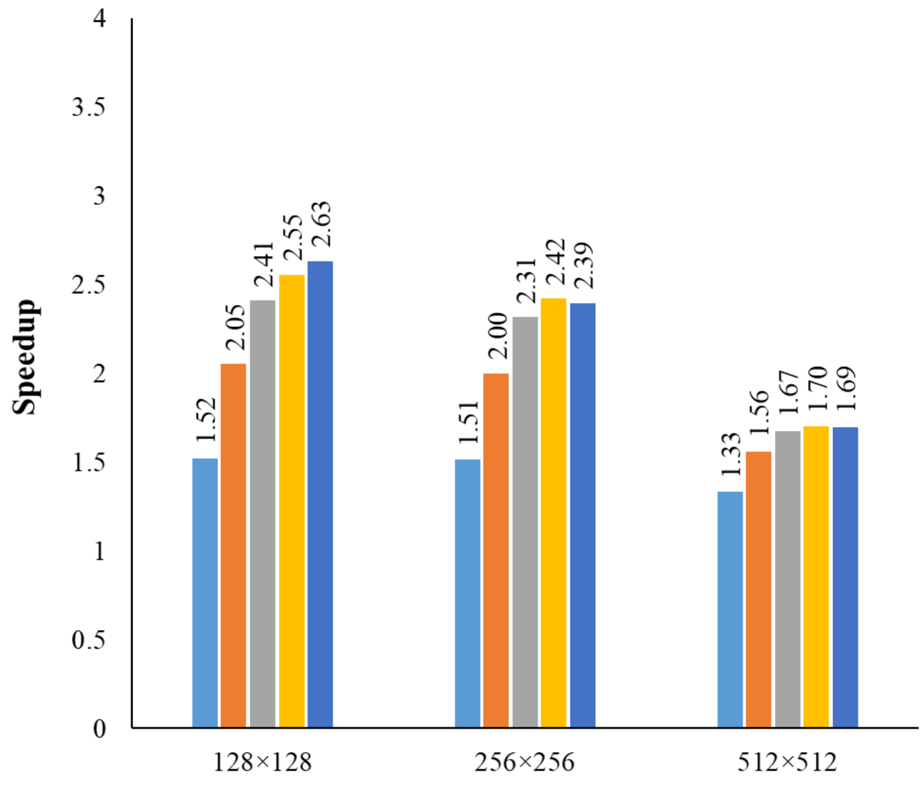


Fig. 3. Speedup of the ADI-CEIDD algorithm vs ADI

4- Conclusions

In the present study, a new ADI-CEIDD approach is proposed by combining the ADI method with the explicit-implicit domain decomposition method. ADI-CEIDD can achieve a speedup of 1.3 to 2.6 times compared with the ADI method by increasing the occupancy. Increasing the number of subdomains improves the performance of the ADI-CEIDD. Although numerical experiments show high stability of the ADI-CEIDD, its error is higher than the ADI. Furthermore, the results show that the ADI-CEIDD method is more advantageous to problems with coarse grids

References

[1] F. Salvadore, M. Bernardini, M. Botti, GPU accelerated flow solver for direct numerical simulation of turbulent flows, Journal of Computational Physics, 235 (2013) 129-142.

[2] S. Vanka, A.F. Shinn, K.C. Sahu, Computational Fluid Dynamics Using Graphics Processing Units: Challenges and Opportunities, Fluids and Thermal Systems; Advances for Process Industries, (2011) 429-437.

[3] G. Alfonsi, S.A. Ciliberti, M. Mancini, L. Primavera, GPGPU implementation of mixed spectral-finite difference computational code for the numerical integration of the three-dimensional time-dependent incompressible Navier–Stokes equations, Computers & Fluids, 102 (2014) 237-249.

[4] N. Sakharnykh, Tridiagonal solvers on the GPU and applications to fluid simulation, in: NVIDIA GPU Technology Conference, San Jose, California, USA, 2009, pp. 17-19.

[5] S. Ha, J. Park, D. You, A GPU-accelerated semi-implicit fractional-step method for numerical solutions of incompressible Navier–Stokes equations, Journal of Computational Physics, 352 (2018) 246-264.

[6] Q. Du, Mu, M, Wu, Z N, Efficient parallel algorithms for parabolic problems, SIAM Journal on Numerical Analysis, 39(5) (2002) 1469-1487.

[7] X.-h. Sun, Y. Zhuang, stablized explici-implicit domain decomposition method for the numerical solution of finit difference equation, SIAM Journal on Numerical Analysis, 24(1) (2002) 335-358.

[8] C. Du, D. Liang, An efficient S-DDM iterative approach for compressible contamination fluid flows in porous media, Journal of Computational Physics, 229(12) (2010) 4501-4521.

[9] Z. Zhou, D. Liang, Y. Wong, The new mass-conserving S-DDM scheme for two-dimensional parabolic equations with variable coefficients, Applied Mathematics and

HOW TO CITE THIS ARTICLE

A. Foadaddini, S. A. Zolfaghari, H. Mahmoodi Darian, Developing an alternating direction explicit-implicit domain-decomposition approach to solve heat transfer equation on graphics processing unit. Amirkabir J. Mech. Eng., 53(special issue 3) (2021). 461-464.

DOI: 10.22060/mej.2019.16178.6295

نشریه مهندسی مکانیک امیرکبیر

نشریه مهندسی مکانیک امیرکبیر، دوره ۵۳ شماره ویژه ۳، سال ۱۴۰۰، صفحات ۱۹۱۵ تا ۱۹۳۶
DOI: 10.22060/mej.2019.16178.6295

ارائه رویکرد تقسیم دامنه صریح- ضمنی جهت‌متغیر برای حل معادله انتقال حرارت روی پردازنده گرافیکی

علی فوادالدینی[۱]، سید علیرضا ذوالفقاری[۲،*]، حسین محمودی داریان[۳]

[۱] گروه مهندسی مکانیک، دانشگاه بیرجند، بیرجند، ایران
[۲] گروه مهندسی مکانیک، دانشگاه بیرجند، بیرجند، ایران
[۳] دانشکده علوم مهندسی، پردیس دانشکده‌های فنی، دانشگاه تهران، تهران، ایران

تاریخچه داوری:
دریافت: ۱۳۹۸/۰۲/۰۱
بازنگری: ۱۳۹۸/۰۴/۲۹
پذیرش: ۱۳۹۸/۰۶/۱۱
ارائه آنلاین: ۱۳۹۸/۰۷/۱۷

کلمات کلیدی:
دینامیک سیالات محاسباتی
پردازش موازی
پردازنده گرافیکی
حلگر ضمنی جهت‌متغیر
تقسیم دامنه صریح- ضمنی

خلاصه: در تحقیق حاضر، رویکرد جدید تقسیم دامنه صریح- ضمنی جهت‌متغیر با ترکیب روش ضمنی جهت‌متغیر و روش تقسیم دامنه صریح- ضمنی برای حل معادله انتقال حرارت هدایت دو بعدی روی پردازنده گرافیکی ارائه شده است. در این روش تخمین مقادیر مرزی با یک طرح عددی صریح صورت گرفته و برای حل درون زیردامنه‌ها از روش ضمنی جهت‌متغیر استفاده می‌شود. سپس از یک طرح ضمنی برای تصحیح مقادیر روی مرز استفاده می‌شود. همچنین، آزمایش عددی برای تحلیل دقت و سرعت روش به انجام رسیده است. نتایج تحقیق نشان می‌دهد که در روش ارائه‌شده با افزایش مشغولیت پردازنده گرافیکی سرعت حل بین ۱/۳ تا ۲/۶ برابر نسبت به روش ضمنی جهت‌متغیر افزایش می‌یابد. در روش ارائه‌شده با افزایش تعداد تقسیمات، سرعت محاسبات افزایش و دقت پاسخ کاهش می‌یابد. خطای روش ارائه‌شده از روش ضمنی جهت‌متغیر بیشتر است با این حال نتایج نشان‌دهنده پایداری بالای روش ارائه‌شده است. همچنین نتایج نشان می‌دهد که مزیت روش ارائه‌شده در شبکه‌های ریز بیشتر از شبکه‌های درشت است بگونه‌ای که با افزایش اندازه شبکه از ۲۵۶×۲۵۶ به ۵۱۲×۵۱۲ مقدار پارامتر افزایش سرعت از ۲/۴ به ۱/۷ کاهش می‌یابد.

۱- مقدمه

معادله انتقال حرارت هدایت دو بعدی جزو معادلات پرکاربرد در حوزه مهندسی مکانیک است. با توجه به پیچیدگی محاسباتی و هزینه محاسباتی بالای حل این قبیل معادلات در مسائل واقعی با بازه زمانی بزرگ و ابعاد دامنه وسیع، نیاز مبرمی به توسعه روش‌های حل سریع و دقیق برای حل آن‌ها وجود دارد که البته ارائه این روش‌ها می‌تواند موجب توسعه روش‌های حل سایر معادلات با ویژگی‌های مشابه شود.

در سال‌های اخیر رشد تکنولوژی‌های مرتبط با پردازش موازی امکان وسیعی را برای تسریع محاسبات ایجاد نموده است. در این میان پردازنده گرافیکی همه‌منظوره یک ابزار قدرتمند در پردازش موازی است که با توزیع پردازش روی تعداد زیادی هسته و پنهان‌سازی تاخیرات حافظه سرعت محاسبات را به نحو قابل ملاحظه‌ای افزایش می‌دهد. این ابزار جایگاه مهمی در دینامیک سیالات محاسباتی پیدا کرده و تحقیقات زیادی پیرامون پیاده‌سازی روش‌های عددی رایج روی پردازنده گرافیکی صورت گرفته است [۱].

با توجه به معماری ویژه پردازنده گرافیکی روش‌های عددی مناسب این سخت‌افزار باید امکان ایجاد تعداد بالایی بخش‌های مستقل محاسباتی را فراهم نمایند. در روش‌های حل صریح، محاسبه مجهولات در هر نقطه از شبکه محاسباتی از سایر نقاط مستقل است. این موضوع امکان فعال‌سازی تعداد زیادی نخ محاسباتی را فراهم

* نویسنده عهده‌دار مکاتبات: zolfaghari@birjand.ac.ir

می‌کند و به همین دلیل، روش‌های حل عددی صریح در بسیاری از تحقیقات مورد توجه قرار گرفتند [۲-۴]. اما با توجه به محدودیت روش‌های حل صریح معادلات در اندازه گام زمانی، روش‌های ضمنی نیز برای حل بسیاری از معادلات پیاده‌سازی شده و مورد تحلیل قرار گرفت[۲, ۵-۸]. در حل ضمنیِ معادلات در شبکه ساختاریافته، استفاده از روش‌های مرحله جزئی[۱] و یا روش ضمنی جهت‌متغیر[۲] منجر به تشکیل یک دستگاه معادله سه قطری به ازای هر خط از شبکه می‌شود. سپس می‌توان حل هر دستگاه معادله را به یک نخ محاسباتی واگذار نمود. تحقیقات نشان می‌دهد که افزایش سرعت ناشی از استفاده پردازنده گرافیکی در این روش به دلیل کم‌بودن تعداد نخ‌های فعال شدیدا محدود می‌شود [۲, ۸].

در تحقیق حاضر برای بهبود عملکرد حلگرهای ضمنی روی پردازنده گرافیکی، یک رویکرد ترکیبی بر مبنای تقسیم دامنه ارائه شده است. تقسیم دامنه روشی برای تقسیم دامنه مکانی حل به تعدادی زیردامنه است که موجب ایجاد بخش‌های مستقل محاسباتی می‌شود. با ایجاد بخش‌های مستقل محاسباتی امکان توزیع آن‌ها روی هسته‌های پردازشی متعدد و حل موازی فراهم می‌شود. روش‌های تقسیم دامنه به دو دسته روش‌های تکراری و غیرتکراری تقسیم می‌شوند. در روش‌های تکراری پس از تقسیم دامنه به چند زیردامنه، در هر گام زمانی حل درون هر زیردامنه چندین بار تکرار می‌شود. اما در دروش‌های غیرتکراری، حل درون زیردامنه‌ها برای هر گام زمانی تنها یکبار به انجام می‌رسد. همچنین، روش‌های تقسیم دامنه به دو دسته متداخل و غیرمتداخل تقسیم می‌شوند. در روش‌های متداخل زیردامنه‌ها دارای نقاط مشترک هستند اما در روش‌های غیرمتداخل زیردامنه‌ها نقطه مشترکی ندارند. روش تقسیم دامنه صریح- ضمنی یکی از روش‌های تقسیم دامنه غیرتکراری و غیرمتداخل است. این روش در مقایسه با سایر روش‌های تقسیم دامنه به لحاظ هزینه محاسباتی و تبادل اطلاعات بین هسته‌های محاسباتی بسیار بهینه است. روش صریح- ضمنی در تحقیقات زیادی مورد بررسی قرار گرفته است. داسون و همکاران [۹] روشی ترکیبی ارائه کردند که در آن حل درون زیردامنه‌ها با استفاده از طرح ضمنی و حل روی مرزهای مشترک زیردامنه‌ها با طرح صریح انجام می‌شود. این روش دارای محدودیت پایداری می‌باشد. دو و همکاران[۱۰] از یک طرح صریح مرتبه بالا و همچنین طرح صریح چند مرحله‌ای برای محاسبه مقادیر مرزهای داخلی استفاده کردند. تحقیقات ایشان نشان داد که استفاده از طرح صریح مرتبه بالا و چندمرحله‌ای، پایداری روش صریح- ضمنی را بهبود می‌دهد. در سال ۲۰۰۲ ژانگ و سان [۱۱] روش صریح- ضمنی تصحیح شده[۳] را ارائه نمودند. در این روش یک مرحله تصحیح ضمنی به الگوریتم صریح- ضمنی جهت تصحیح مقادیر روی مرز زیردامنه‌ها افزوده می‌شود. مرحله تصحیح موجب پایداری بدون قید و شرط حل می‌گردد.

استفاده از روش تقسیم دامنه صریح- ضمنی در مسائل دو بعدی در تحقیقات متعددی مورد بررسی قرار گرفته است. ژانگ و سان [۱۱] در وضعیتی که تقسیم دامنه تنها در یک جهت صورت گرفته است تصحیح روی مرز را با تشکیل یک دستگاه معادله در راستای مرز به انجام رساندند. تشکیل دستگاه معادله در راستای مرزهای مشترک مستلزم انتقال اطلاعات بین هسته‌ها است به همین دلیل روش زیگ- زاگی برای تقسیم دامنه‌ها توسط شی و لیائو [۱۲] در سال ۲۰۰۶ ارائه شد. در این روش برای انجام تصحیح روی مرز نیازی به تشکیل دستگاه معادله وجود ندارد. ژو و همکاران [۱۳] در سال ۲۰۰۹ یک روش تصحیح ۲ مرحله‌ای را برای وضعیتی که مرزهای زیردامنه‌ها یکدیگر را قطع نمایند، ارائه کردند. لیائو و همکاران [۱۴] به جای استفاده از تصحیح ۲ مرحله‌ای برای تصحیح نقاط تقاطع مرزها، از تلفیق روش مرز زیگ- زاگی و غیر زیگ- زاگی استفاده نمودند. دو و لیانگ [۱۵] در سال ۲۰۱۰ الگوریتم تقسیم دامنه صریح- ضمنی را با تکنیک جداسازی[۴] ترکیب نمودند و روش ترکیبی جداسازی- تقسیم دامنه[۵] را ارائه نمودند. در این روش، برای حل جریان سیال آلاینده در محیط متخلخل از یک طرح عددی محلی چند مرحله‌ای برای مرحله پیش‌بینی استفاده شده و از طرح ضمنی برای حل داخل زیردامنه‌ها استفاده می‌شود. در سال ۲۰۱۴ لیانگ و دو [۱۶] روش مذکور را برای حل معادلات دیفرانسیل سهموی توسعه داده و تحلیل نمودند. این روش پایداری بدون قید و شرط ندارد؛ اما استفاده از یک طرح عددی محلی بهینه شرایط پایداری را بهبود می‌بخشد. در سال ۲۰۱۸ ژو و همکاران [۱۷] یک روش ترکیبی جداسازی- تقسیم

1 Fractional steps method
2 Alternating Direction Implicit method
3 Corrected Explicit-Implicit Domain Decomposition Method (CEIDD)
4 Splitting technique
5 splitting domain decomposition method (S-DDM)

دامنه بهینه و جرم- پایسته[1] برای حل معادله سهموی با ضرایب متغیر ارائه نمودند. در این روش از یک طرح عددی صریح وزنی و چندنقطه‌ای برای به‌دست‌آوردن فلاکس‌های مرزی استفاده شده و از طرح ضمنی و جداسازی‌شده برای داخل زیردامنه‌ها استفاده می‌شود. این روش دارای پایداری بدون قید و شرط نیست اما استفاده از طرح چندنقطه‌ای شرایط پایداری را بهبود بخشیده است.

همانگونه که بیان شد، بهره‌گیری بهینه از پردازنده گرافیکی در شرایطی امکان‌پذیر است که بتوان مسئله مورد نظر را به تعداد زیادی بخش‌های مستقل تقسیم نمود و از این طریق تعداد زیادی نخ فعال را ایجاد کرد. یکی از چالش‌های پیاده‌سازی حلگر ضمنی جهت‌متغیر روی پردازنده گرافیکی کم‌بودن تعداد دستگاه معادلات سه‌قطری مستقل و در نتیجه، کم‌بودن تعداد نخ‌های فعال در حل می‌باشد. به همین منظور در تحقیق حاضر روش تقسیم دامنه صریح- ضمنی جهت‌متغیر[2] با ترکیب روش ضمنی جهت‌متغیر و روش تقسیم دامنه صریح- ضمنی جهت حل معادله انتقال حرارت هدایت دوبعدی روی پردازنده گرافیکی ارائه شده است. در این روش به جای تشکیل یک دستگاه معادله مستقل به ازای هر خط از شبکه ساختاریافته، می‌توان چندین دستگاه معادله مستقل در هر خط ایجاد نمود. این موضوع موجب می‌شود که تعداد نخ‌های فعال نسبت به روش ضمنی جهت‌متغیر چندین برابر افزایش یافته و امکان بهره‌گیری از مزایای پردازنده گرافیکی فراهم گردد. در ادامه ابتدا روش عددی مورد استفاده تشریح شده و سپس نحوه پیاده سازی حلگر روی پردازنده گرافیکی مورد بررسی قرار می‌گیرد. در بخش نتایج خطای ناشی از تقسیم دامنه بررسی شده و ضمن بررسی تاثیر پارامترهای مختلف بر عملکرد این روش مزیت بکارگیری آن نسبت به روش ضمنی جهت‌متغیر مقایسه می‌شود.

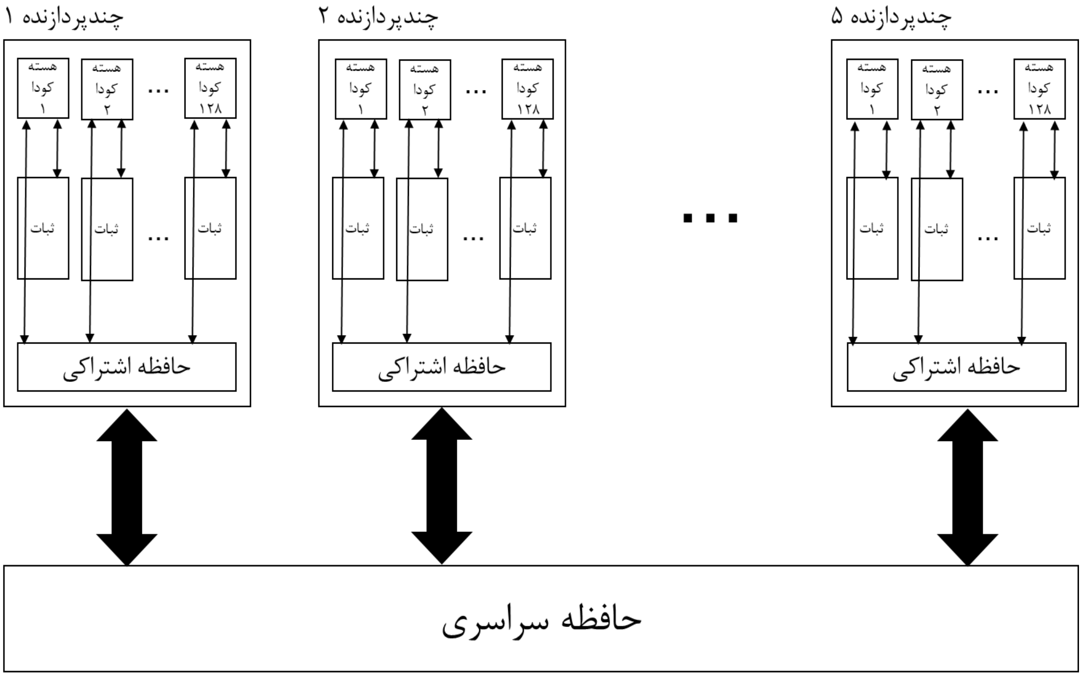
۲- پردازنده گرافیکی همه‌منظوره

هر پردازنده گرافیکی از چندین چندپردازنده[3] تشکیل شده است که مشخصات و امکانات آن وابسته به معماری و توانایی محاسباتی[4](نسخه) آن پردازنده می‌باشد. هر چندپردازنده به صورت عمده دارای منابعی است که برای پردازش از آن بهره می‌گیرد. این منابع شامل صدها هسته کودا[5]، انواع متفاوتی حافظه (سراسری[6]، اشتراکی[7]، بافتی[8]، ثبات و...) و واحدی نظیر زمانبند[9] است که دستورالعمل‌ها را پیش از پردازش آماده‌سازی می‌کند. پردازنده گرافیکی استفاده‌شده در تحقیق حاضر NVIDIA GEFORCE GTX 960M است که دارای معماری ماکسول (نسخه ۵،۰) می‌باشد. شکل ۱ نمایی شماتیک از یک چندپردازنده را در معماری ماکسول نمایش می‌دهد. همانگونه که مشاهده می‌شود این پردازنده گرافیکی دارای ۵ چندپردازنده است. هر چندپردازنده دارای ۱۲۸ هسته کودا می‌باشد. این تعداد هسته محاسباتی در مقایسه با یک پردازنده مرکزی که تعداد محدودی هسته پردازنده دارد بسیار قابل توجه است و تفاوت اساسی ساختار پردازنده گرافیکی و مرکزی نیز از همین جهت است. پردازنده گرافیکی برای مشغول نگه‌داشتن این هسته‌های کودا نیازمند تعداد زیادی پردازش‌های مستقل است تا آن‌ها را بین هسته‌ها توزیع نماید. این پردازش‌ها یا بخش‌های مستقل محاسباتی نخ نامیده می‌شوند.

کودا به عنوان یک رابط برنامه‌ریزی این امکان را به کاربر می‌دهد که شبکه یک، دو و یا سه بعدی از نخ‌ها را ساخته و آن‌ها را در قالب بلاک‌ها دسته‌بندی کند. به این ترتیب کاربر محاسبات مستقل از هم را در این نخ‌ها توزیع می‌نماید. وجود تعداد زیادی نخ امکان افزایش سرعت محاسبات را برای پردازنده گرافیکی فراهم می‌کند. پردازنده گرافیکی از طرفی با تقسیم بار کاری این نخ‌ها بین صدها هسته پردازشگر امکان تقسیم کار و کاهش زمان اجرای محاسبات را فراهم می‌سازد. از طرف دیگر، مدت زمانی را که در آن هسته‌های پردازشگر منتظر رسیدن اطلاعات مربوط به یک دسته از نخ‌ها از حافظه هستند، با پرداختن به دستورالعمل دسته دیگری از نخ‌ها پوشش می‌دهد. بنابراین هرچه بخش‌ها مستقل محاسباتی در روش عددی مدنظر بیشتر باشد امکان فعال‌سازی نخ‌های بیشتر فراهم شده و از این طریق می‌توان هسته‌های کودای بیشتری را در پردازنده گرافیکی مشغول نگه داشت و از طرفی تاخیر حافظه را پنهان نمود.

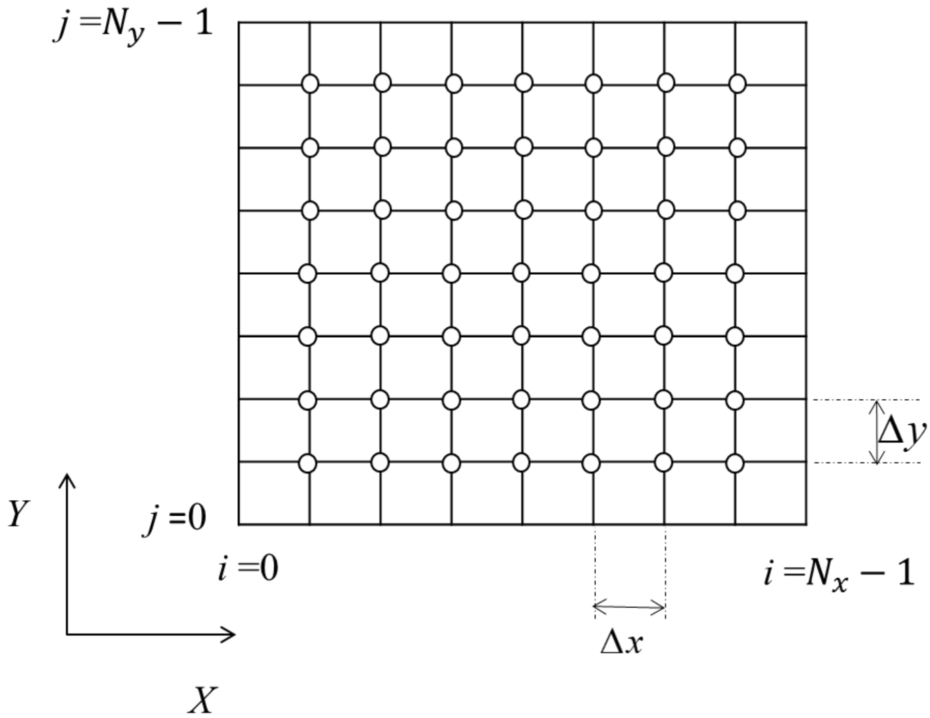
محاسبات در توابعی به نام کرنل بین نخ‌ها توزیع می‌شوند. پس از اجرای کرنل، بلاک‌ها برای انجام محاسبات مربوط به خودشان به

1 mass-conserving
2 Alternating Direction Implicit-Corrected Explicit-Implicit Domain Decomposition (ADI-CEIDD)
3 Multiprocessor
4 Compute capability
5 CUDA core
6 Global memory
7 Shared memory
8 Texture
9 Schedulers



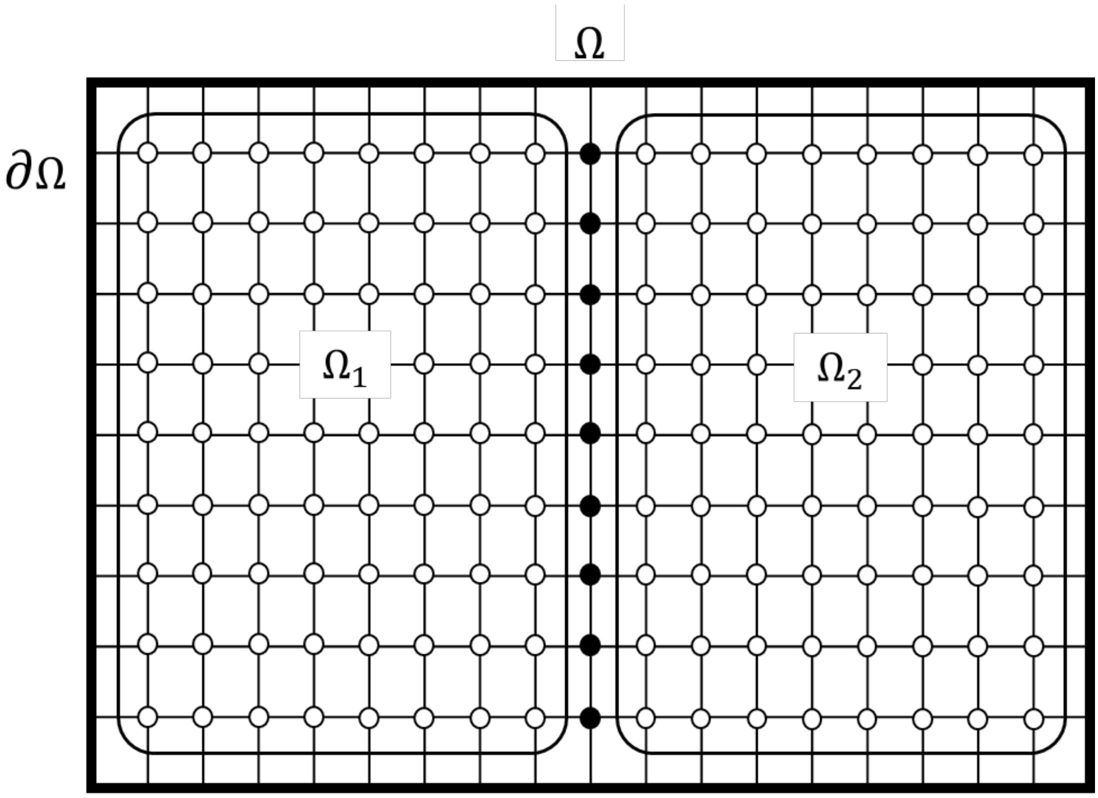
شکل ۱: نمایی ساده شده از پردازنده گرافیکی NVIDIA GEFORCE GTX 960M

Fig. 1. NVIDIA GEFORCE GTX 960M Architecture



شکل ۲: نمایی از شبکه حل تفاضل محدود با اندازه شبکه N_x×N_y

Fig. 2 . Discretisized two-dimensional domain



شکل ۳: تقسیم دامنه دامنه Ω به دو زیردامنه غیر متداخل Ω_1 و Ω_2

Fig. 3 . Decomposing the domain Ω into two non-overlapping subdomains Ω_1 and Ω_2

چندپردازنده‌ها ارسال می‌شوند. اگرچه فراهم‌نمودن تعداد زیادی نخ با استفاده از یک روش عددی مناسب شرط اول استفاده از مزایای پردازنده گرافیکی می‌باشد، با این حال نکاتی اساسی نیز باید در پیاده‌سازی روش‌های عددی روی پردازنده گرافیکی باید رعایت شود تا بیشترین سرعت پردازش حاصل گردد. در پیوست ۱ با اشاره به دو مفهوم مشغولیت پردازنده گرافیکی و دسترسی بهینه به حافظه نکات مهمی پیرامون بهره‌گیری بهینه از ظرفیت این پردازنده گرافیکی بیان شده است.

۳- روش عددی

۳-۱- روش ضمنی جهت‌متغیر

روش ضمنی جهت‌متغیر یکی از روش‌های عددی رایج در دینامیک سیالات محاسباتی است. در این روش این امکان وجود دارد که حل یک معادله دیفرانسیل را به بخش‌های جزئی تقسیم کنیم تا بتوان آن را از طریق تشکیل مجموعه‌ای از دستگاه معادلات سه‌قطری حل نمود.

رابطه زیر معادله انتقال حرارت هدایت دو بعدی مفروض را نشان می‌دهد:

$$x \in [0, L_x] \tag{۱}$$

$$\rho c \frac{\partial T(x,y,t)}{\partial t} = k_x \frac{\partial^2 T(x,y,t)}{\partial x^2} + k_y \frac{\partial^2 T(x,y,t)}{\partial y^2} + f(x,y,t)$$

$$T(L_x, y, t) = g_2(y,t) \quad T(x,0,t) = g_3(x,t)$$

$$T(x,L_y,t)=g_4(x,t) \quad T(x,y,0)=T_0(x,y)$$

$$T(0,y,t)=g_1(y,t)$$

که در آن T دما، k_x و k_y ضرایب هدایت حرارتی، ρ چگالی و c گرمای ویژه است.

در روش ضمنی جهت‌متغیر، معادله انتقال حرارت هدایت دو بعدی برای هر گام زمانی شامل دو مرحله می‌باشد. با گسسته‌سازی معادله روی شبکه دو بعدی نشان‌داده‌شده در شکل ۲، مراحل حل به این صورت است:

۱- حل ضمنی در راستای Y: در مرحله اول ترم مشتق مکانی در راستای X به صورت صریح و در راستای Y به صورت ضمنی گسسته‌سازی می‌شوند:

$$\rho c\frac{T_{i,j}^{n+1/2}-T_{i,j}^{n}}{\Delta t/2}=k_x\frac{T_{i+1,j}^{n}-2T_{i,j}^{n}+T_{i-1,j}^{n}}{\Delta x^2}+ k_y\frac{T_{i,j+1}^{n+1/2}-2T_{i,j}^{n+1/2}+T_{i,j-1}^{n+1/2}}{\Delta y^2}+f_{i,j}^{n+1} \quad (۲)$$

۲- حل ضمنی در راستای X: سپس در مرحله دوم ترم مشتق در راستای Y به صورت صریح و مشتق در راستای X به صورت ضمنی گسسته می‌شود:

$$\rho c\frac{T_{i,j}^{n+1}-T_{i,j}^{n+1/2}}{\Delta t/2}=k_x\frac{T_{i+1,j}^{n+1}-2T_{i,j}^{n+1}+T_{i-1,j}^{n+1}}{\Delta x^2}+ k_y\frac{T_{i,j+1}^{n+1/2}-2T_{i,j}^{n+1/2}+T_{i,j-1}^{n+1/2}}{\Delta y^2}+f_{i,j}^{n+1} \quad (۳)$$

در هر مرحله از الگوریتم ضمنی جهت متغیر مجموعه‌ای از دستگاه معادلات سه‌قطری تشکیل می‌شود که تعداد آن‌ها به ابعاد شبکه حل وابسته است. برای مثال، با گسسته‌سازی معادله در یک شبکه ۸×۸ در هر مرحله از الگوریتم ۸ دستگاه معادله در راستای X و Y خواهیم داشت. در ادامه ضرایب روی قطر، بالای قطر و پایین قطر دستگاه معادلات سه‌قطری به ترتیب با a ،b و c نمایش داده شده و d نشانگر سمت راست دستگاه معادلات است. با پیاده‌سازی این روش روی پردازنده گرافیکی در هر یک از مراحل الگوریتم می‌توان هر دستگاه معادله سه‌قطری را به عنوان یک محاسبه مستقل بر یک نخ محاسباتی منطبق کرد. سپس می‌توان برای حل دستگاه معادله سه‌قطری تشکیل‌شده از الگوریتم توماس استفاده کرد. الگوریتم توماس از سریع‌ترین روش‌های حل سری دستگاه معادلات سه قطری می‌باشد که از روش حذفی گوس گرفته شده است.

۲-۳- روش تقسیم دامنه صریح- ضمنی جهت‌متغیر

همانگونه که بیان شد در روش ضمنی جهت‌متغیر تعداد دستگاه معادلات تشکیل‌شده با توجه به اندازه شبکه تفاضل محدود تعداد مشخصی است. در اغلب موارد حل این دستگاه معادلات به کمک الگوریتم توماس امکان ایجاد تعداد محدودی نخ محاسباتی را فراهم می‌سازد که برای مشغول‌نگه‌داشتن پردازنده گرافیکی و پنهان‌کردن تاخیر حافظه کافی نیست. هدف تحقیق حاضر ارائه یک رویکرد ترکیبی است که در آن از روش‌های تقسیم دامنه برای بهبود عملکرد روش ضمنی جهت‌متغیر روی پردازنده گرافیکی استفاده شده است. در روش‌های مختلف تقسیم دامنه، دامنه مکانی حل به چند زیردامنه تقسیم می‌شود. سپس این امکان وجود دارد که معادلات مشتقات جزئی در هر یک از این زیردامنه‌ها به صورت مستقل از دیگری حل شوند. بنابراین می‌توان مقادیر مربوط به هر زیردامنه را به یک هسته محاسباتی مجزا منتقل نمود و حل معادلات در تمام زیردامنه‌ها را به صورت هم‌زمان و موازی به انجام رساند. برای مثال در شکل ۳ دامنه Ù به دو زیردامنه غیر متداخل $Ù_1$ و $Ù_2$ تقسیم شده است. همانگونه که مشاهده می‌شود با این شیوه دامنه حل به دو ناحیه نقاط مرز داخلی و نقاط زیردامنه‌ها تقسیم می‌شود. به این ترتیب برای حل معادله دیفرانسیل در دامنه Ù روش صریح- ضمنی اصلاح‌شده[1] در هر گام زمانی شامل سه مرحله است:

۱- حل نقاط مرز داخلی با یک طرح عددی صریح

۲- حل نقاط زیردامنه‌ها با استفاده از یک طرح عددی ضمنی و شرط مرزی به‌دست‌آمده از مرحله ۱ برای مرز داخلی

۳- صرف‌نظرکردن از مقادیر به‌دست‌آمده در مرحله ۱ برای نقاط مرز داخلی و جایگزینی مقادیر آن با مقادیر محاسبه‌شده به وسیله یک طرح عددی ضمنی

در روش ترکیبی تقسیم دامنه صریح- ضمنی جهت‌متغیر در هر مرحله از الگوریتم ضمنی جهت‌متغیر دامنه حل در راستای حل ضمنی به زیردامنه‌های متعدد تقسیم می‌شود. سپس از روش صریح-

1 Corrected Explicit-Implicit Domain Decomposition Method (CEIDD)

ضمنی برای برقراری ارتباط بین این زیردامنه‌ها استفاده می‌شود. به این ترتیب هر گام زمانی از الگوریتم ترکیبی تقسیم دامنه صریح-ضمنی جهت‌متغیر شامل مراحل زیر می‌باشد:

۱- حل ضمنی در راستای Y: در این مرحله دامنه حل به nos زیردامنه در راستای Y تقسیم شده و نقاط شبکه به دو دسته نقاط داخل زیردامنه‌ها و نقاط مرزهای داخلی تقسیم می‌شوند. به عنوان مثال در شکل ۴- الف یک دامنه با اندازه ۱۴×۱۴ در راستای Y به سه زیردامنه تقسیم شده است. سپس حل از گام زمانی n تا ۲/n+۱ در سه زیر مرحله انجام می‌شود:

۱-۱- تخمین مقادیر نقاط مرز داخلی با استفاده از برونیابی مرتبه ۱ از دو گام n و ۲/n-۱:

$$T_{i,j}^{n+1/2}=T_{i,j}^{n}+(T_{i,j}^{n}-T_{i,j}^{n-1/2}) \tag{۴}$$

۱-۲- حل ضمنی در راستای Y در داخل زیردامنه‌ها:

$$\begin{aligned}\rho c\frac{T_{i,j}^{n+1/2}-T_{i,j}^{n}}{\Delta t/2}&=k_x\frac{T_{i+1,j}^{n}-2T_{i,j}^{n}+T_{i-1,j}^{n}}{\Delta x^2}+\\ &k_y\frac{T_{i,j+1}^{n+1/2}-2T_{i,j}^{n+1/2}+T_{i,j-1}^{n+1/2}}{\Delta y^2}+f_{i,j}^{n+1}\end{aligned} \tag{۵}$$

در این مرحله در هر زیردامنه تعدادی دستگاه معادله سه‌قطری در راستای Y تشکیل شده و مقادیر تخمینی مرحله ۱-۲ به عنوان مقادیر مرزی برای تشکیل دستگاه معادلات مورد استفاده قرار می‌گیرد.

۱-۳- مقادیر بدست‌آمده برای نقاط مرز داخلی در مرحله ۱-۱ نادیده گرفته شده و از طرح عددی ضمنی برای تصحیح مقادیر مرز داخلی استفاده می‌شود:

$$\begin{aligned}\rho c\frac{T_{i,j}^{n+1/2}-T_{i,j}^{n}}{\Delta t/2}&=k_x\frac{T_{i+1,j}^{n}-2T_{i,j}^{n}+T_{i-1,j}^{n}}{\Delta x^2}+\\ &k_y\frac{T_{i,j+1}^{n+1/2}-2T_{i,j}^{n+1/2}+T_{i,j-1}^{n+1/2}}{\Delta y^2}+f_{i,j}^{n+1}\end{aligned} \tag{۶}$$

در این رابطه مقدار $T_{i,j}^{n+1/2}$ از مقادیر مربوط به گام n و همچنین مقادیر $T_{i,j+1}^{n+1/2}$ و $T_{i,j-1}^{n+1/2}$ که در زیرمرحله ۱- ۲حاصل شده است محاسبه می‌گردد. بنابراین نیاز به تشکیل یک دستگاه معادله نیست و مقادیر از رابطه (۷) و به صورت صریح حاصل می‌شود:

$$T_{i,j}^{n+1/2}=\left(\begin{aligned}&\frac{k_x\Delta t}{2\rho c}\frac{T_{i+1,j}^{n}-2T_{i,j}^{n}+T_{i-1,j}^{n}}{\Delta x^2}\\ &+\frac{k_y\Delta t}{2\rho c}\frac{T_{i,j+1}^{n+1/2}+T_{i,j-1}^{n+1/2}}{\Delta y^2}\\ &+\frac{\Delta t}{2\rho c}f_{i,j}^{n+1}+T_{i,j}^{n}\end{aligned}\right) \Big/\left(1+\frac{k_y\Delta t}{\rho c\Delta y^2}\right) \tag{۷}$$

۲- حل ضمنی در راستای X: در این مرحله دامنه حل به nos زیردامنه در راستای X تقسیم می‌شود. نحوه تقسیم دامنه به سه زیردامنه در راستای X در شکل ۴- ب نمایش داده شده است. سپس حل از گام زمانی ۲/n+۱ تا n+۱ در سه زیر مرحله انجام می‌شود:

۲-۱- تخمین مقادیر نقاط مرز داخلی با استفاده از برونیابی مرتبه ۱ از دو گام ۲/n+۱ و n:

$$T_{i,j}^{n+1}=T_{i,j}^{n+1/2}+(T_{i,j}^{n+1/2}-T_{i,j}^{n}) \tag{۸}$$

۲-۲- حل ضمنی در راستای X در داخل زیردامنه‌ها:

$$\begin{aligned}\rho c\frac{T_{i,j}^{n+1}-T_{i,j}^{n+1/2}}{\Delta t/2}&=k_x\frac{T_{i+1,j}^{n+1}-2T_{i,j}^{n+1}+T_{i-1,j}^{n+1}}{\Delta x^2}+\\ &k_y\frac{T_{i,j+1}^{n+1/2}-2T_{i,j}^{n+1/2}+T_{i,j-1}^{n+1/2}}{\Delta y^2}+f_{i,j}^{n+1}\end{aligned} \tag{۹}$$

در این مرحله در هر زیردامنه تعدادی دستگاه معادله سه‌قطری در راستای X تشکیل شده و مقادیر تخمینی مرحله ۱-۲ به عنوان مقادیر مرزی برای تشکیل دستگاه معادلات مورد استفاده قرار می‌گیرد.

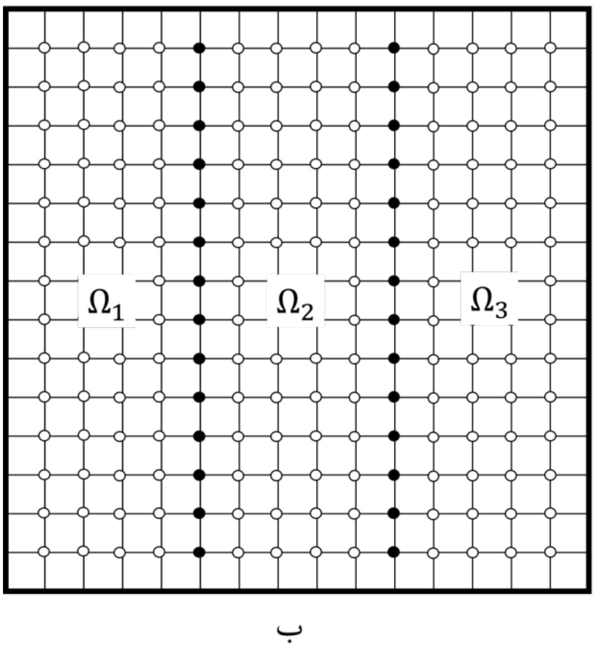
۲-۳- مقادیر بدست‌آمده برای نقاط مرز داخلی در مرحله ۲-۱ نادیده گرفته شده و از یک طرح عددی برای تصحیح مقادیر مرز داخلی استفاده می‌شود:

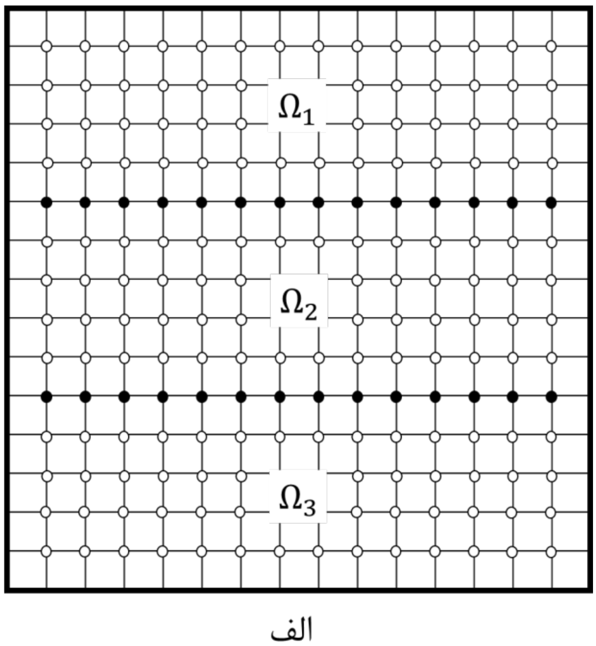
$$T_{i,j}^{n+1}=\left(\begin{aligned}&\frac{k_x\Delta t}{2\rho c}\frac{T_{i+1,j}^{n+1}+T_{i-1,j}^{n+1}}{\Delta x^2}\\ &+\frac{k_y\Delta t}{2\rho c}\frac{T_{i,j+1}^{n+1/2}-2T_{i,j}^{n+1/2}+T_{i,j-1}^{n+1/2}}{\Delta y^2}\\ &+\frac{\Delta t}{2\rho c}f_{i,j}^{n+1}+T_{i,j}^{n+1/2}\end{aligned}\right) \Big/\left(1+\frac{k_x\Delta t}{\rho c\Delta x^2}\right) \tag{۱۰}$$

همانگونه که در شکل ۴ نشان داده شده است در روش ترکیبی تقسیم دامنه صریح- ضمنی جهت‌متغیر به جای تشکیل یک دستگاه معادله به ازای هر خط از شبکه می‌توان ۳ دستگاه معادله تشکیل داد. همانگونه که بیان شد یکی از ویژگی‌های اساسی مورد نیاز برای یک روش عددی سازگار با معماری پردازنده گرافیکی این است که روش عددی مذکور بتواند تعداد زیادی نخ محاسباتی را به صورت همزمان فعال نماید. با ایجاد تعداد زیادی دستگاه معادله مستقل از هم در روش تقسیم دامنه صریح- ضمنی جهت‌متغیر می‌توان به ازای هر دستگاه معادله یک نخ را فعال نمود و تعداد نخ‌های فعال را نسبت به روش ضمنی جهت‌متغیر چندبرابر نمود. این ویژگی موجب می‌شود که روش تقسیم دامنه صریح- ضمنی جهت‌متغیر عملکرد بهتری نسبت به روش ضمنی جهت‌متغیر روی پردازنده گرافیکی داشته باشد. یکی دیگر از مزایای این روش این است که با تعیین تعداد تقسیمات شبکه کاربر می‌تواند تعداد دستگاه معادلات سه‌قطری را متناسب با تعداد نخ محاسباتی مورد نیاز افزایش دهد. این موضوع از آن جهت دارای اهمیت است که تعداد چندپردازنده‌ها و از طرفی تعداد نخ محاسباتی مورد نیاز پردازنده گرافیکی برای مشغول‌نگه‌داشتن چندپردازنده‌ها در مدل‌های مختلف پردازنده گرافیکی متفاوت است. بنابراین در روش تقسیم دامنه صریح- ضمنی جهت‌متغیر می‌توان برای یک پردازنده گرافیکی با ظرفیت بالا تعداد تقسیمات شبکه را افزایش داده و برای یک پردازنده گرافیکی کوچک تعداد تقسیمات شبکه را کاهش داد.

۴- پیاده‌سازی روش تقسیم دامنه صریح-ضمنی جهت‌متغیر روی پردازنده گرافیکی

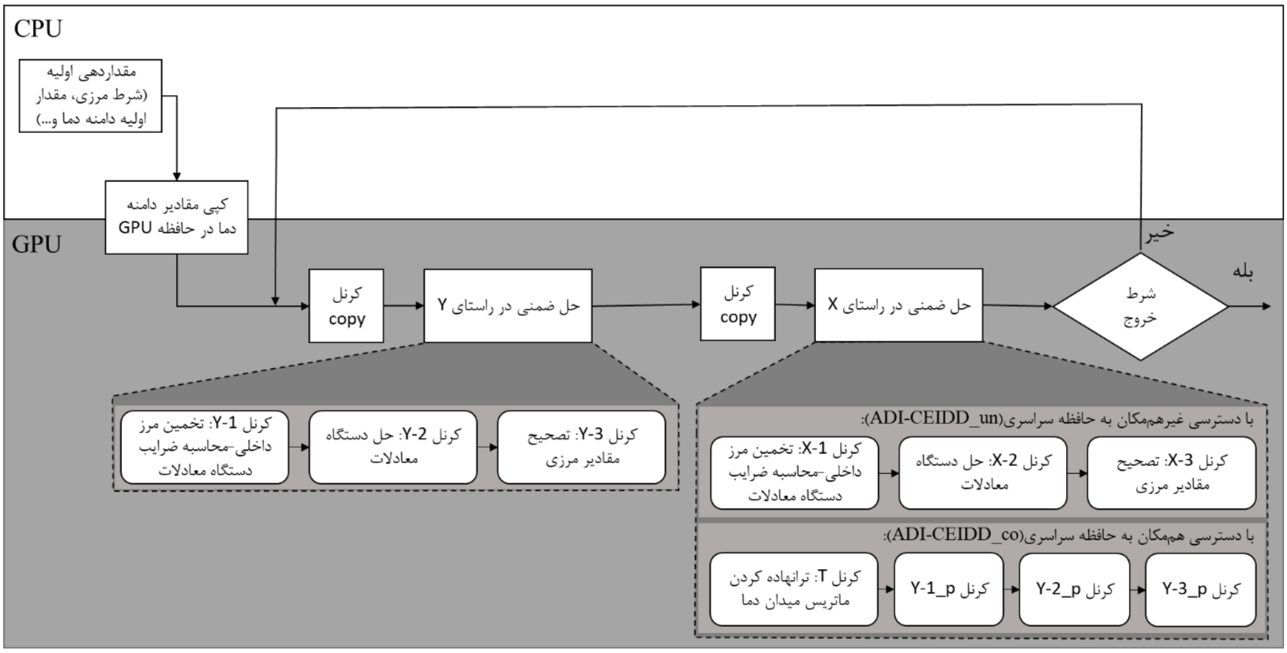
شکل ۵ یک طرح کلی از نحوه پیاده‌سازی الگوریتم روش تقسیم دامنه صریح- ضمنی جهت‌متغیر را نمایش می‌دهد. همانگونه که مشاهده می‌شود تمامی بخش‌های اصلی الگوریتم که در بخش ۴ تشریح شد در دستگاه (پردازنده گرافیکی) به انجام می‌رسد تا از انتقال اطلاعات بین میزبان (پردازنده مرکزی) و دستگاه جلوگیری شود. به این ترتیب آرایه‌های مربوط به ضرایب دستگاه معادلات $(dev\_a,\ dev\_b,\ dev\_c)$، سمت راست دستگاه معادلات (dev_d) و همچنین آرایه دما (dev_T) در حافظه سراسری ایجاد





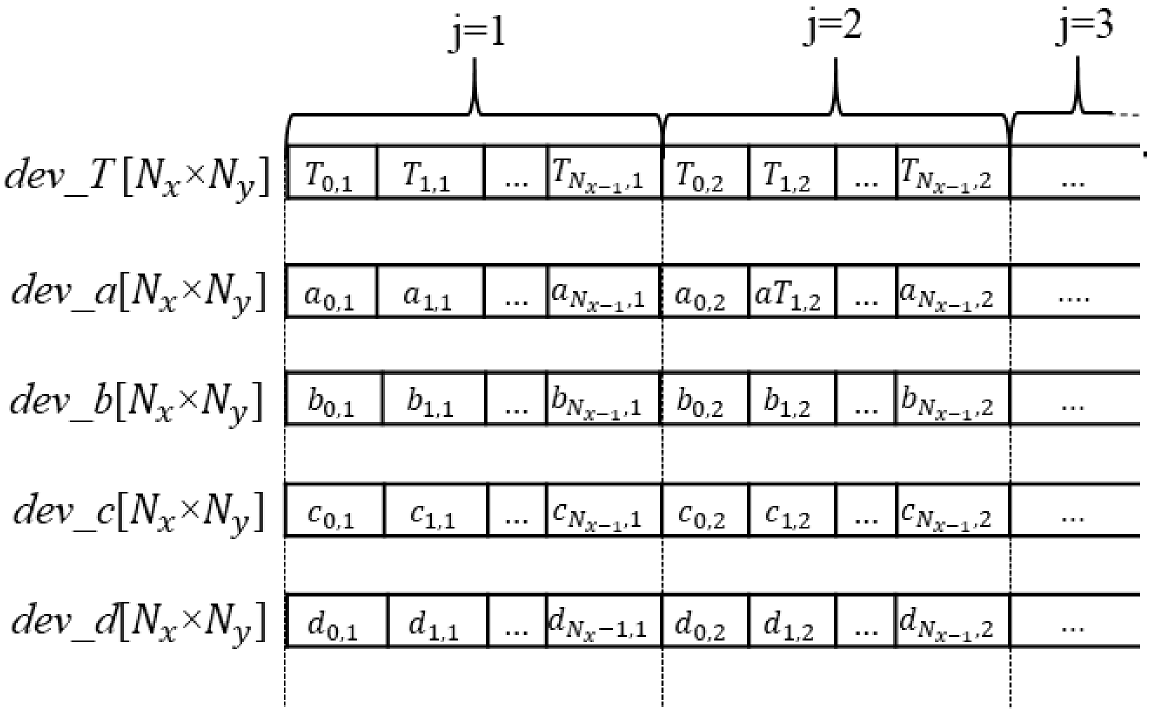
شکل ۴: تقسیم دامنه حل به زیردامنه‌های متعدد برای الف) حل ضمنی در راستای Y. ب)حل ضمنی در راستای X

Fig. 4. Domain decomposition in X direction (left) and Y direction (right) in ADI-CEIDD



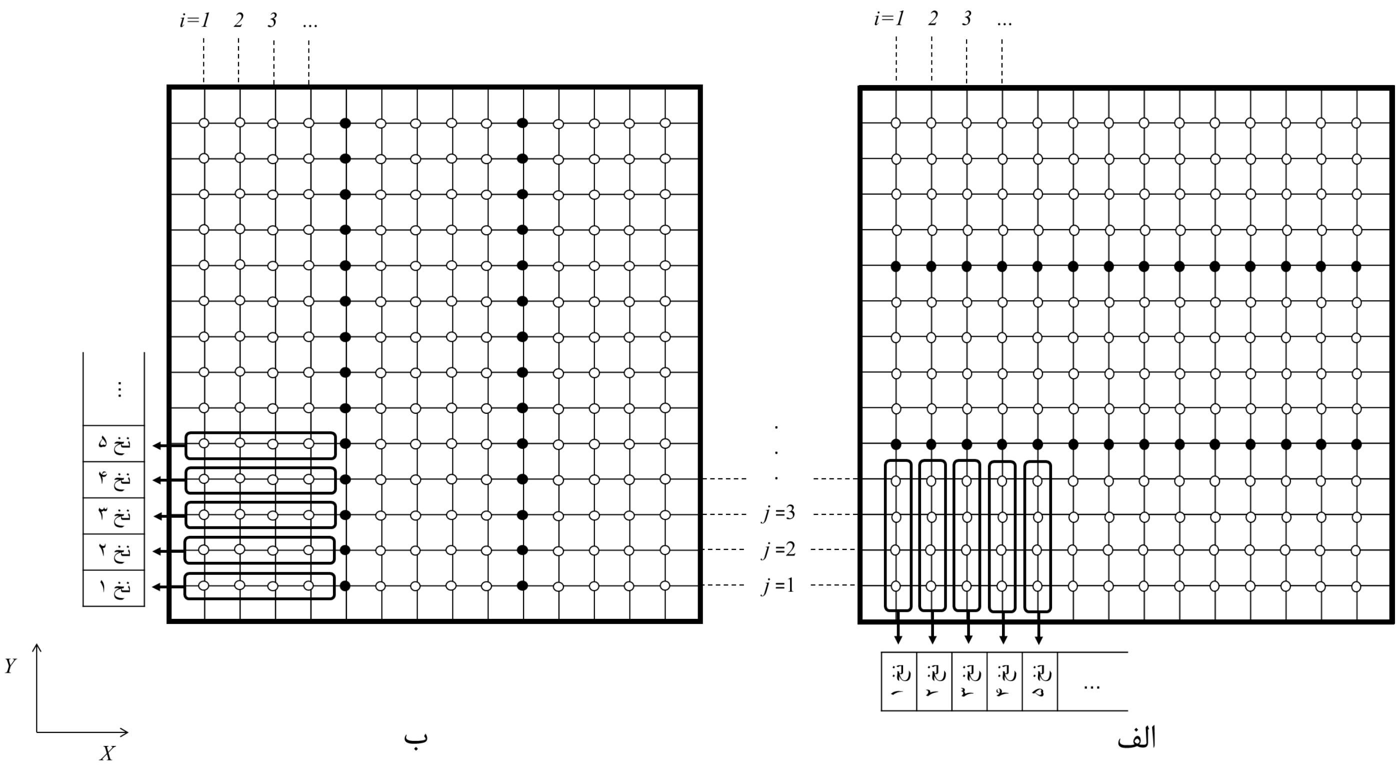
شکل ۵: طرح کلی از پیاده‌سازی روش تقسیم دامنه صریح- ضمنی جهت‌متغیر روی پردازنده گرافیکی

Fig. 5. Summarized code structure for ADI-CEIDD method



شکل ۶: ترتیب ذخیره مقادیر مربوط به نقاط شبکه در آرایه‌های یک بعدی بر حسب اندیس آن‌ها در شبکه

Fig. 6. Mapping two-dimmensional arrays into one-dimmensional arrays



شکل ۷: ارتباط بین نخ‌ها و دستگاه معادلات را برای الف)کرنل Y-۲ و ب)کرنل X-۲

Fig. 7. Correspondence between threads and systems of equations for kernel X-2 (left) and kernel Y-2) right

می‌شود. با وجود اینکه شبکه حل دوبعدی است اما تعریف آرایه‌های دو بعدی برای ذخیره اطلاعات مربوط به دستگاه معادلات و میدان دما توصیه نمی‌شود. دسترسی به اطلاعات ذخیره‌شده در آرایه‌های دو بعدی روی حافظه هزینه زمانی بسیار بیشتری نسبت به آرایه‌های یک بعدی دارد. بنابراین آرایه‌های dev_a, dev_b, dev_c و dev_T آرایه‌های ۱ بعدی هستند. شکل ۶ ترتیب ذخیره مقادیر مربوط به نقاط شبکه در آرایه‌های یک بعدی را بر حسب اندیس آن‌ها در شبکه نمایش می‌دهد.

حل ضمنی در راستای X و Y از سه کرنل اصلی تشکیل شده است. کرنل Y-۱ روی یک شبکه دو بعدی از نخ‌ها که ابعاد آن بر ابعاد شبکه محاسباتی منطبق است اجرا می‌شود. در این صورت هر نخ منطبق بر یک نقطه از شبکه محاسباتی است و مقادیر دستگاه معادلات مربوط به آن نقطه را محاسبه می‌نماید. نخ‌های مربوط به نقاط مجاور مرز داخلی برای بدست‌آوردن مقادیر ضرایب مربوطه باید ابتدا مقدار مرز داخلی را با استفاده از روش مربوط به مرحله ۱-۱ الگوریتم در بخش ۴ تخمین بزنند. این مقدار تخمینی در حافظه ثبات ذخیره شده و سپس برای بدست‌آوردن مقادیر دستگاه معادلات استفاده می‌شود. کرنل X-۱ نیز همین کار را برای بدست‌آوردن ضرایب دستگاه معادله برای حل ضمنی در راستای X انجام می‌دهد.

کرنل X-۲ و Y-۲ روی شبکه ۱ بعدی از نخ‌ها اجرا می‌شود. در این کرنل هر نخ منطبق بر یک دستگاه معادله است و محاسبه مربوط به هر دستگاه معادله سه‌قطری در یک نخ صورت می‌گیرد. شکل ۷ ارتباط بین نخ‌ها و دستگاه معادلات را برای کرنل Y-۱ و X-۱ نمایش می‌دهد. همانگونه که مشاهده می‌شود در کرنل Y-۱ نخ‌های یک وارپ به اطلاعاتی در حافظه نیاز دارند که در درایه‌های نزدیک به هم ذخیره شده‌اند. برای مثال در این کرنل نخ ۱ و نخ ۲ در میدان دما به ترتیب به مقادیر $T_{1,1}$ و $T_{2,1}$ دسترسی پیدا می‌کنند که همانگونه که در شکل ۶ نمایش داده شده در آرایه dev_T در کنار یکدیگر قرار دارند. اما در کرنل X-۱ مقادیر مورد نیاز نخ ۱ و ۲ به اندازه N_x با هم فاصله دارند. فاصله زیاد بین داده‌های مورد نیاز نخ‌های مجاور موجب دسترسی غیرهم‌مکان به حافظه می‌شود که نتیجه آن کاهش سرعت دسترسی به حافظه است. بنابراین کرنل

۱-X نسبت به کرنل ۱-Y زمان بیشتری صرف خواهد کرد. برای جلوگیری از دسترسی غیرهم‌مکان به حافظه سراسری در کرنل ۱-X می‌توان آرایه دما را ترانهاده نموده و حل ضمنی را در جهت Y تکرار نمود. تاثیر دسترسی غیرهم‌مکان به حافظه سراسری در بخش نتایج مورد بررسی قرار گرفته است. در ادامه این مقاله از روش ارائه‌شده با دو نام روش صریح- ضمنی جهت‌متغیر غیر هم‌مکان[1] و روش صریح- ضمنی جهت‌متغیر هم‌مکان[2] یاد شده است که به ترتیب مربوط به پیاده سازی این روش با دسترسی غیرهم‌مکان و دسترسی هم‌مکان به حافظه می‌باشد. کرنل‌های Y-۱_p، Y-۲_p و Y-۳_p در روش هم‌مکان به ترتیب همان کرنل‌های Y-۱ ، Y-۲ و Y-۳ هستند که در نقاط متفاوتی از الگوریتم فراخوانی می‌شوند.

در کرنل ۲-X و ۲-Y هر نخ منطبق بر یک نقطه از شبکه در ناحیه مرز داخلی می‌باشد و مقادیر مربوط به این نقاط را در مرحله تصحیح محاسبه می‌نماید. همچنین برای اجرای مراحل ۱-۱ و ۲-۱ از الگوریتم باید مقادیر مربوط به ½ گام قبل($T_{i,j}^{n-1/2}$ و $T_{i,j}^{n}$) را در یک آرایه ذخیره نماییم. کرنل copy به این منظور پیش از محاسبه مقادیر ضمنی در هر مرحله از الگوریتم اجرا می‌شود.

الگوریتم روش ضمنی جهت‌متغیر نیز با روشی مشابه روش تقسیم دامنه صریح- ضمنی جهت‌متغیر روی پردازنده گرافیکی پیاده‌سازی شده است با این تفاوت که بخش مربوط به تخمین مقادیر مرزی در کرنل ۱-Y و ۱-X حذف شده و کرنل‌های Y-۳، X-۳ و copy نیز به کلی نادیده گرفته می‌شوند. هر آنچه پیرامون دسترسی غیرهم‌مکان در روش تقسیم دامنه صریح- ضمنی جهت‌متغیر بیان شد در روش ضمنی جهت‌متغیر نیز صادق می‌باشد و به همین منظور این روش نیز با دو نام ضمنی جهت‌متغیر غیر هم‌مکان[3] و ضمنی جهت‌متغیر هم‌مکان[4] به ترتیب برای دسترسی غیرهم‌مکان و هم‌مکان به حافظه مورد بررسی قرار می‌گیرد.

مطابق آنچه در بخش ۲ بیان شد مقدار حافظه ثبات برای تمامی کرنل‌ها کمتر از ۳۲ است و اندازه بلاک‌ها نیز بزرگتر از ۶۴ نخ انتخاب شده تا این ۲ عامل موجب کاهش مشغولیت چندپردازنده‌ها نشوند.

1 ADI-CEIDD_un
2 ADI-CEIDD_co
3 ADI_un
4 ADI_co

۵- نتایج

هدف تحقیق حاضر ارائه و تحلیل روش تقسیم دامنه صریح- ضمنی جهت‌متغیر برای حل معادله دیفرانسیل انتقال حرارت هدایت دو بعدی روی پردازنده گرافیکی است. در این بخش روش مذکور از طریق آزمایشات عددی از نظر دقت، پایداری و افزایش سرعت مورد تحلیل قرار گرفته است. به این منظور حل معادله (۱) در دامنه $x \in [0,1]$ و $y \in [0,1]$ و با شرایط مرجع [۱۶] مد نظر است:

$$\begin{cases} k_x = 1, k_y = 1 \\ f(x,y,t) = e^t(x(1-x)y(1-y) \\ +2y(1-y)+2x(1-x)) \\ g_1(y,t)=0, g_1(y,t)=0, g_3(x,t) \\ =0, g_4(x,t)=0 \\ T_0(x,y) = x(1-x)y(1-y) \end{cases} \tag{۱۱}$$

حل تحلیلی این مسئله به این صورت است:

$$T_A(x,y,t) = e^t x(1-x)y(1-y) \tag{۱۲}$$

در ادامه با استفاده از این مثال روش تقسیم دامنه صریح- ضمنی جهت‌متغیر در مقایسه با روش ضمنی جهت‌متغیر تحلیل شده است.

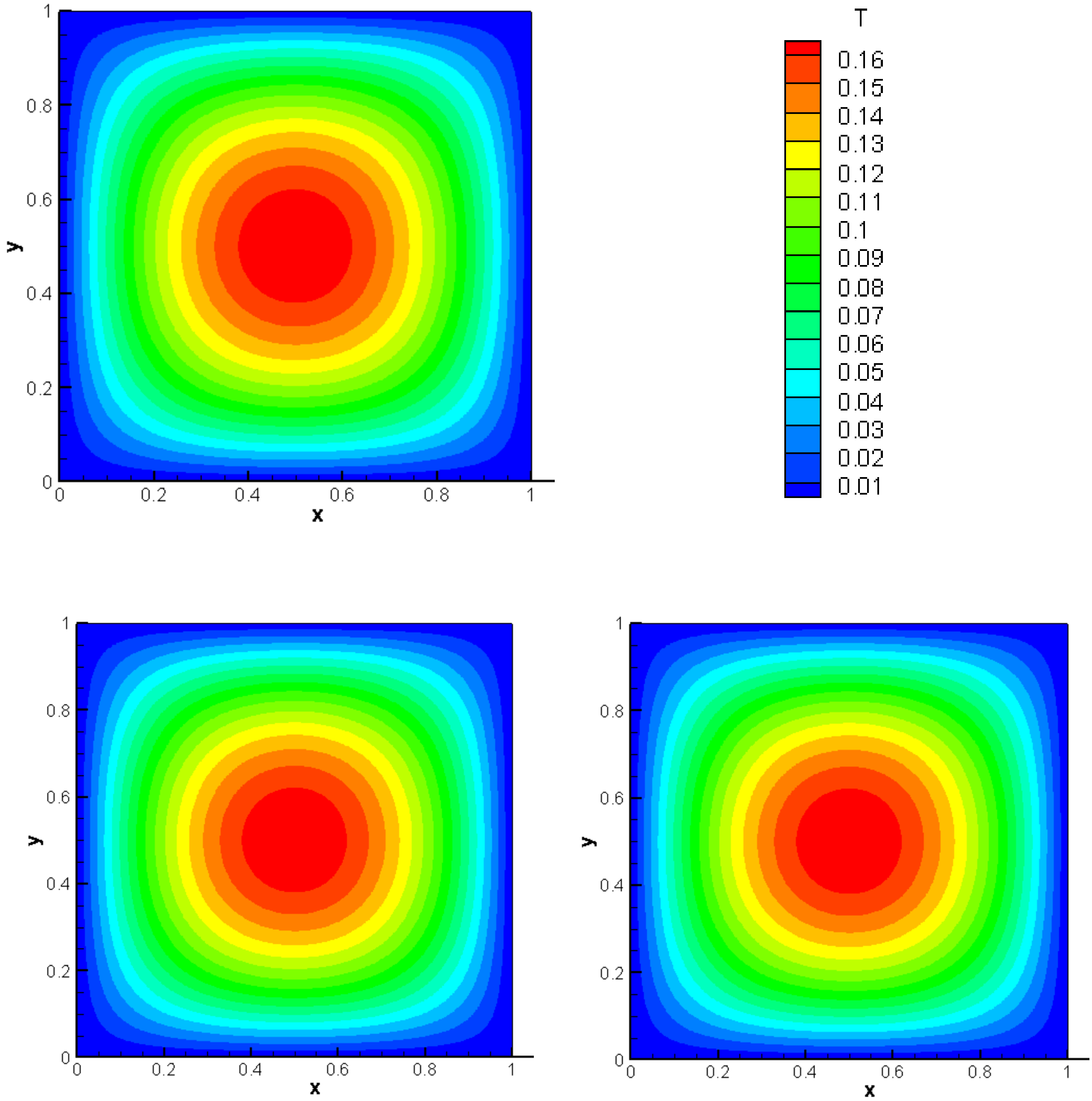
برای بررسی خطا از دو معیار نرم دوم خطا و نرم بینهایت خطا استفاده شده است:

$$E_2^n = \left\{ \Delta x \, \Delta y \sum_{i,j} (T_{i,j}^n - \mathrm{T}_A(x_i, y_j, t_n))^2 \right\}^{\frac{1}{2}} \tag{۱۳}$$

$$E_\infty^n = \max_{i,j} \left\{ |(T_{i,j}^n - \mathrm{T}_A(x_i, y_j, t_n))| \right\} \tag{۱۴}$$

E_2^n و E_∞^n به ترتیب نرم دوم و نرم بی‌نهایت خطا هستند.

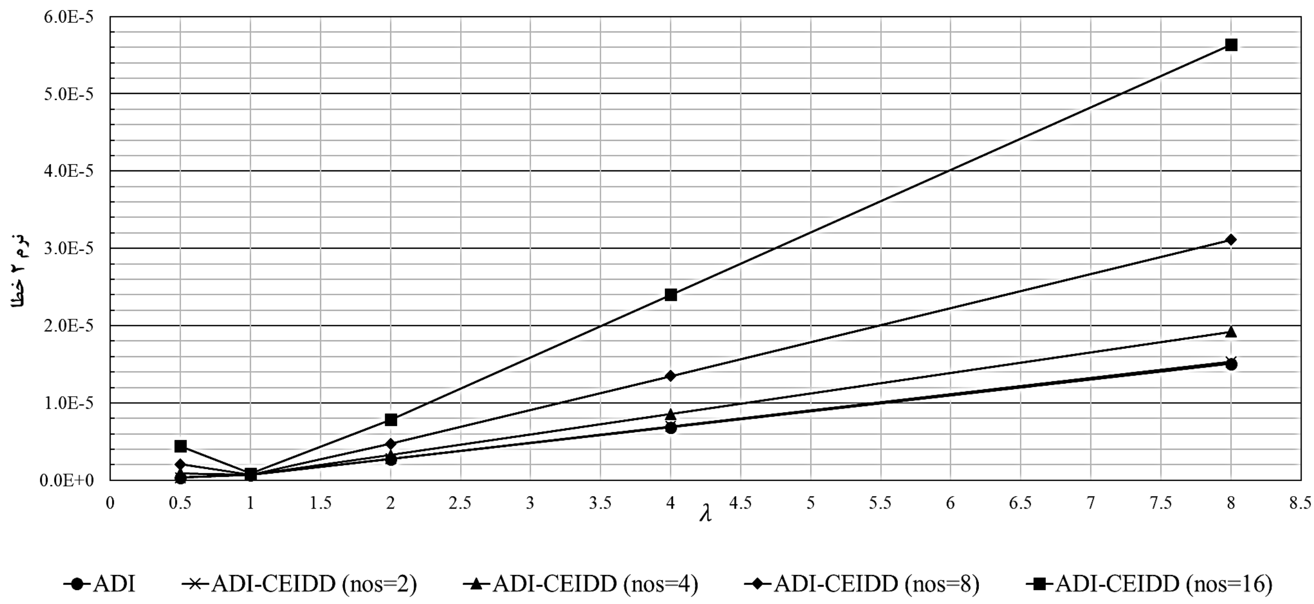
در شکل ۸ کانتور دمای پاسخ روش ضمنی جهت‌متغیر و روش تقسیم دامنه صریح- ضمنی جهت‌متغیر برای شبکه‌ای با اندازه ۲۵۶×۲۵۶ در $t = 1\mathrm{s}$ نمایش داده شده است. همانگونه که مشاهده می‌شود تطابق خوبی بین پاسخ دو روش مذکور و پاسخ تحلیلی وجود دارد. در در ابتدا تاثیر تعداد تقسیمات دامنه (nos) را بر خطا و پایداری مورد بررسی قرار می‌دهیم. در نمودار شکل ۹ با



شکل ۸: کانتور پاسخ روش ضمنی جهت‌متغیر و تقسیم دامنه صریح- ضمنی جهت‌متغیر برای شبکه‌ای با اندازه ۲۵۶×۲۵۶ در $t = 1s$

Fig. 8. Temperature contours for Analytical solution (top left), ADI method (bottom left) and ADI-CEIDD (bottom-) right

فرض تقسیمات شبکه یکنواخت ($\Delta x = \Delta y = h = \frac{1}{256}$) میزان خطای حل به ازای مقادیر مختلف ضریب شبکه $\lambda = (\Delta t .k)/h^2$ ارائه شده است. در این نمودار مقدار λ از ۰/۵ تا ۸ تغییر کرده است و مقادیر نرم ۲ خطا برای پاسخ روش ضمنی جهت‌متغیر و روش تقسیم دامنه صریح- ضمنی جهت‌متغیر به ازای مقادیر مختلف پارامتر nos ارائه شده است. همانگونه که مشاهده می‌شود تقسیمات دامنه حل در روش تقسیم دامنه صریح- ضمنی جهت‌متغیر موجب افزایش خطا نسبت به روش ضمنی جهت‌متغیر شده است. تقسیم دامنه در $\lambda = 1$ تاثیر ناچیزی بر خطا دارد با این حال با افزایش یا کاهش λ اختلاف خطای روش تقسیم دامنه صریح- ضمنی جهت‌متغیر با روش



شکل ۹: نمودار نرم ۲ خطا بر حسب λ برای روش ضمنی جهت‌متغیر و تقسیم دامنه صریح-ضمنی جهت‌متغیر برای شبکه‌ای با اندازه ۲۵۶×۲۵۶

Fig. 9. L_2-norm error for a 256×256 grid

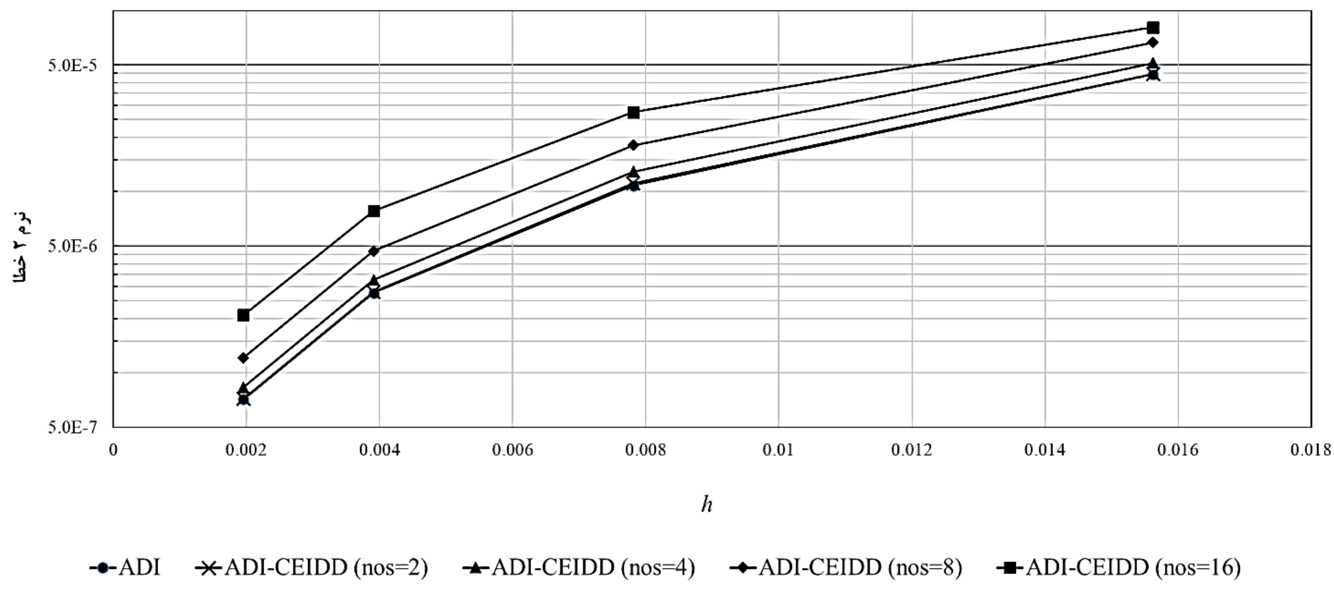
جدول ۱: نرم ۲ و نرم بی‌نهایت خطا برای شبکه‌ای با اندازه ۲۵۶×۲۵۶ در زمان‌های مختلف

Table 1. Numerical errors of the ADI-CEIDD for three different times

زمان (S)	روش ADI		روش ADI-CEIDD ($nos =$ ۴)		روش ADI-CEIDD ($nos =$ ۱۶)	
	E_2^n	E_∞^n	E_2^n	E_∞^n	E_2^n	E_∞^n
۰/۵	۹/۶۲۳ $\times$ ۱۰$^{-۵}$	۵/۰۲۸ $\times$ ۱۰$^{-۷}$	۲/۷۱۴ $\times$ ۱۰$^{-۴}$	۳/۱۱۴ $\times$ ۱۰$^{-۶}$	۹/۷۲۸ $\times$ ۱۰$^{-۶}$	۱/۱۴۱ $\times$ ۱۰$^{-۶}$
۱/۰	۷/۱۲۷ $\times$ ۱۰$^{-۴}$	۵/۱۹۹ $\times$ ۱۰$^{-۶}$	۸/۳۹۴ $\times$ ۱۰$^{-۴}$	۷/۳۴۳ $\times$ ۱۰$^{-۶}$	۲/۰۲۱ $\times$ ۱۰$^{-۳}$	۱/۸۲۷ $\times$ ۱۰$^{-۳}$
۱/۵	۸/۷۲۵ $\times$ ۱۰$^{-۴}$	۶/۳۶۵ $\times$ ۱۰$^{-۶}$	۱/۱۲۹ $\times$ ۱۰$^{-۳}$	۱/۰۸۵ $\times$ ۱۰$^{-۵}$	۳/۲۳۰ $\times$ ۱۰$^{-۳}$	۲/۸۸۲ $\times$ ۱۰$^{-۳}$

ضمنی جهت‌متغیر افزایش می‌یابد. خطای روش تقسیم دامنه صریح-ضمنی جهت‌متغیر به ازای $\lambda > 1$ با افزایش λ افزایش می‌یابد. و سرعت تغییرات خطا با افزایش پارامتر nos شدت می‌گیرد. همچنین نتایج نشان می‌دهد که روش تقسیم دامنه صریح- ضمنی جهت‌متغیر همانند روش ضمنی جهت‌متغیر از پایداری بالایی برخوردار است و در دامنه وسیعی از مقادیر λ پایدار می‌ماند.

شکل ۱۰ نمودار نرم ۲ خطا در روش ضمنی جهت‌متغیر و تقسیم دامنه صریح- ضمنی جهت‌متغیر را برای تقسیمات شبکه مختلف (h) نمایش می‌دهد. در این نمودار خطا برای $\lambda = 2$ و تقسیمات شبکه $\frac{1}{64}$، $\frac{1}{128}$، $\frac{1}{256}$ و $\frac{1}{512}$ ارائه شده است. همانگونه که مشاهده می‌شود به ازای تمامی مقادیر h خطای روش تقسیم دامنه صریح- ضمنی جهت‌متغیر از روش ضمنی جهت‌متغیر بیشتر است و با افزایش تعداد تقسیمات دامنه (nos) تفاوت خطای بین دو روش بیشتر می‌شود. بعلاوه در روش تقسیم دامنه صریح- ضمنی جهت‌متغیر همانند روش ضمنی جهت‌متغیر خطای حل با کاهش h کاهش یافته است. برای حلگر تقسیم دامنه صریح- ضمنی



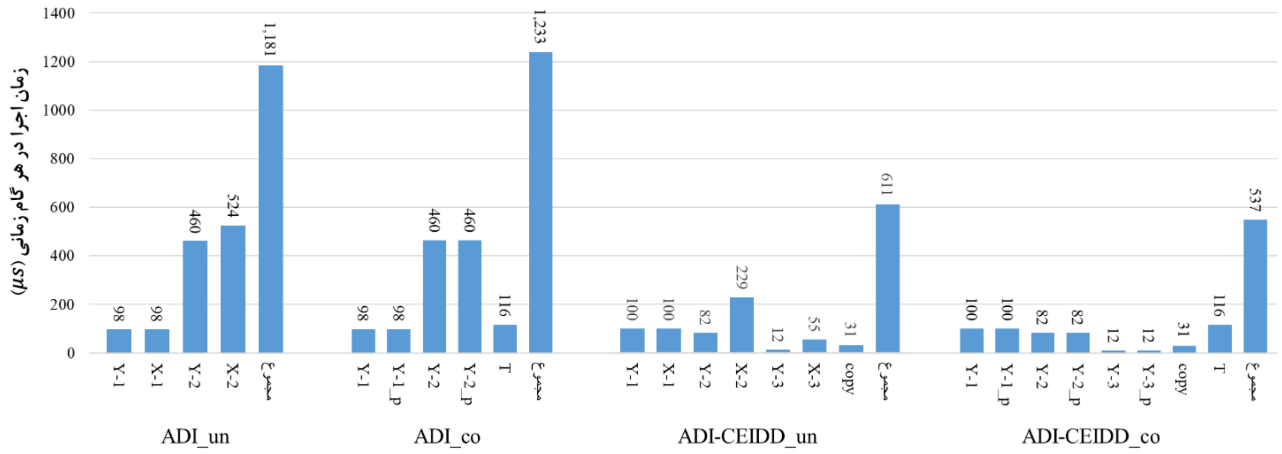
شکل ۱۰: نرم ۲ خطا در روش ضمنی جهت‌متغیر و تقسیم دامنه صریح-ضمنی جهت‌متغیر را برای تقسیمات شبکه مختلف (h)

Fig. 10. L_2-norm error for a 256×256 grid for three different grid resolutions

جهت‌متغیر $(nos=۱۶)$ با کاهش h از $\frac{1}{64}$ به $\frac{1}{128}$، نرم ۲ خطا از $8/07\times10^{-5}$ به $2/75\times10^{-5}$ کاهش یافته است. اما با کاهش □ از $\frac{1}{256}$ به $\frac{1}{512}$، نرم ۲ خطا از 8.73×10^{-6} به $2/09^{-6}$ کاهش می‌یابد. چنین روندی برای روش تقسیم دامنه صریح- ضمنی جهت‌متغیر به ازای nos های مختلف مشاهده می‌شود. این نتیجه حاکی از آن است که مطابق انتظار، با افزایش تقسیمات شبکه حساسیت دقت پاسخ به اندازه شبکه کاهش می‌یابد.

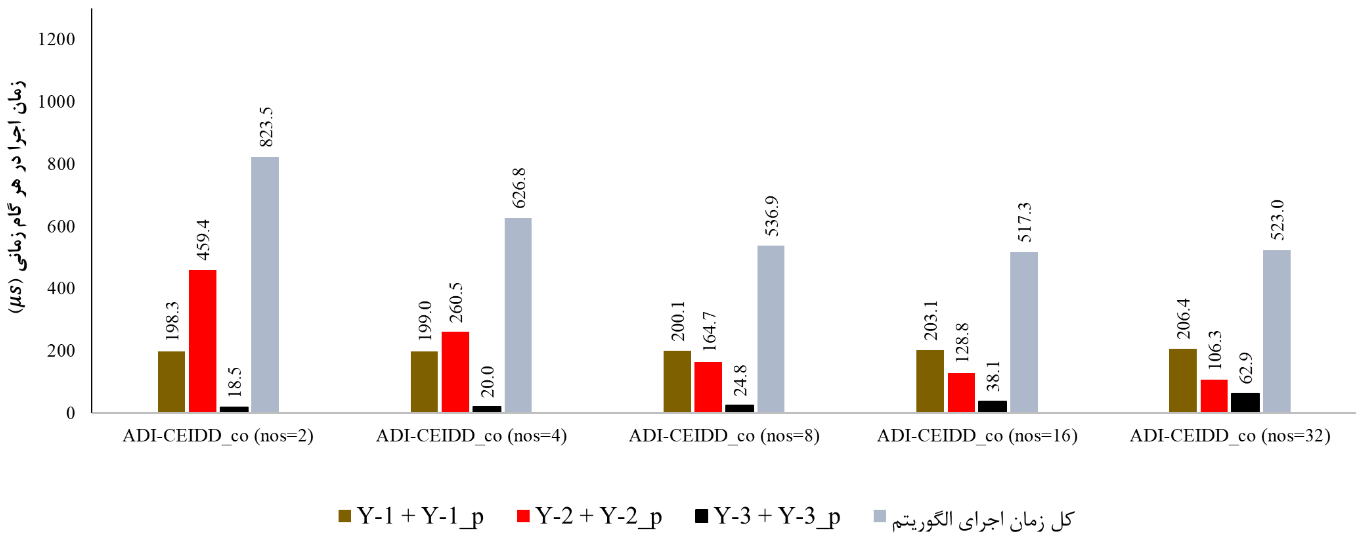
مقادیر خطای روش ضمنی جهت‌متغیر و تقسیم دامنه صریح- ضمنی جهت‌متغیر در سه زمان ۰/۵، ۱/۰ و ۱/۵ ثانیه در جدول ۱ ارائه شده است. همانگونه که مشاهده می‌شود مقدار خطای مذکور برای هر دو روش با افزایش زمان افزایش یافته است.

در ادامه نحوه عملکرد روش‌های ارائه‌شده از نظر زمان اجرا و عملکرد محاسباتی مورد بررسی قرار می‌گیرد. همانطور که در بخش ۴ بیان شد، پیاده‌سازی روش‌های ضمنی جهت‌متغیر و تقسیم دامنه صریح- ضمنی جهت‌متغیر روی پردازنده گرافیکی از کرنل‌های مختلفی تشکیل شده که هر یک بخش خاصی از محاسبات مربوطه را به انجام می‌رسانند. برای توسعه روش‌های عددی مناسب و بهینه‌سازی الگوریتم‌های محاسباتی برای پیاده‌سازی روی پردازنده گرافیکی ابتدا باید میزان اهمیت بخش‌های مختلف الگوریتم را مشخص نمود. بخش‌هایی از الگوریتم که درصد بیشتری از زمان اجرا را به خود اختصاص می‌دهند برای بهینه‌سازی باید در الویت قرار بگیرند. در شکل ۱۱ زمان اجرا برای کرنل‌های مختلف در یک گام زمانی برای شبکه‌ای با اندازه ۲۵۶×۲۵۶ ارائه شده است. همانگونه که در شکل ۱۱- الف و ۱۱- ب مشاهده می‌شود در روش ضمنی جهت‌متغیر بخش عمده زمان محاسبات در یک گام زمانی به حل دستگاه معادلات اختصاص دارد. در نسخه غیر هم‌مکان الگوریتم روش ضمنی جهت‌متغیر زمان اجرا برای کرنل ۲-Yو ۲-X در مجموع ۹۸۴ میکروثانیه می‌باشد که معادل ۸۳ درصد از کل زمان مربوط به یک گام زمانی در این الگوریتم است. بنابراین همانگونه که در اهداف تحقیق حاضر نیز مدنظر قرار گرفته است، تغییر شیوه حل به منظور تسریع در حل دستگاه معادلات در روش تقسیم دامنه صریح- ضمنی جهت‌متغیر می‌تواند به نحو قابل ملاحظه‌ای زمان حل را کاهش دهد. همانگونه که در شکل ۱۱- الف و ۱۱- ج ارائه



شکل ۱۱: زمان اجرای کرنل‌های مختلف برای شبکه‌ای با اندازه ۲۵۶×۲۵۶ در الگوریتم

Fig. 11. Calculation time for different kernels (Grid size: 256×256)

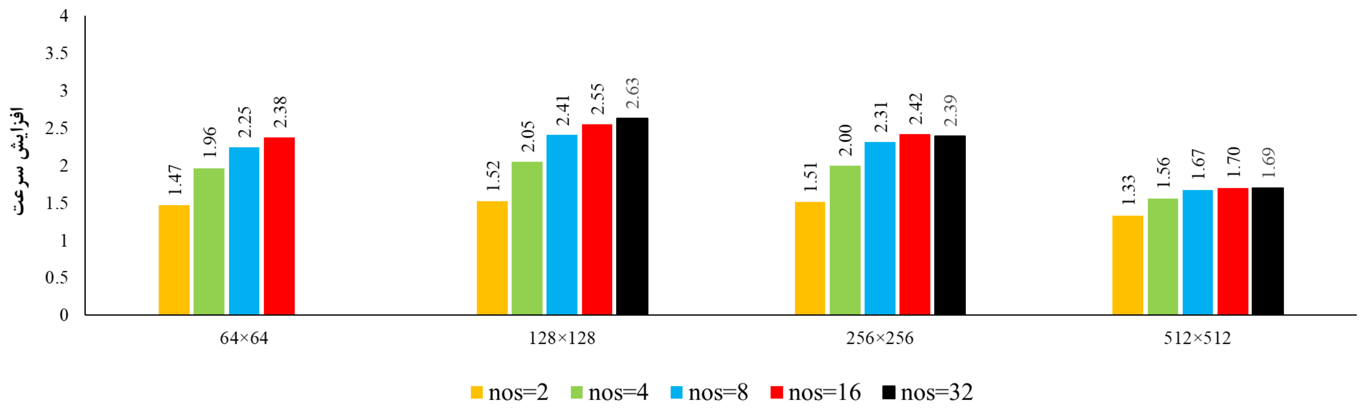


شکل۱۲: زمان اجرای بخش‌های مختلف الگوریتم تقسیم دامنه صریح- ضمنی جهت‌متغیر بر حسب nos برای شبکه‌ای با اندازه ۲۵۶×۲۵۶

Fig. 12. Calculation time for different part of the algorithm in ADI-CEIDD method

شده است، تقسیم دامنه حل در روش تقسیم دامنه صریح- ضمنی جهت‌متغیر توانسته با جایگزینی دستگاه معادلات سه‌قطری بزرگ در روش ضمنی جهت‌متغیر با تعداد زیادی دستگاه معادلات کوچک‌تر و فعال‌سازی تعداد زیادی نخ محاسباتی، زمان اجرای کرنل Y-۲ را از ۴۶۰ میکروثانیه به ۸۲ میکروثانیه و زمان اجرای X-۲ را از ۵۲۴ میکروثانیه به ۲۲۹ میکروثانیه کاهش دهد. بنابراین کل زمان حل دستگاه معادلات با بهره‌گیری از تقسیم‌دامنه ۶۸ درصد کاهش یافته است و این موضوع موجب شده است که کل زمان حل از ۱۲۳۳ میکروثانیه به ۶۱۰ میکروثانیه کاهش یابد.

همانگونه که در بخش ۴ بیان شد، دسترسی غیرهم‌مکان به حافظه



شکل ۱۳: افزایش سرعت نسخه هم‌مکان روش تقسیم دامنه صریح-ضمنی جهت‌متغیر برحسب اندازه شبکه و تعداد تقسیم دامنه

Fig. 13. Speedup of the ADI-CEIDD algorithm vs ADI

سراسری در کرنل X-۲ موجب کاهش سرعت محاسبات می‌شود. در نسخه غیرهم‌مکان الگوریتم روش ضمنی جهت‌متغیر زمان اجرای کرنل X-۲ ۵۲۴ ثانیه و کرنل Y-۲ ۴۶۰ میکروثانیه است. در نسخه هم‌مکان الگوریتم روش ضمنی جهت‌متغیر با ترانهاده‌کردن دامنه و اجرای مجدد حل ضمنی در راستای Y می‌توان این تفاوت در زمان حل دستگاه معادلات در دو مرحله الگوریتم را از بین برد با این حال زمان اجرای کرنل T که صرف ترانهاده‌کردن دامنه شده است به کل زمان اجرای الگوریتم اضافه می‌شود. همانگونه که شکل ۱۱- ب نشان می‌دهد، با ترانهاده‌کردن ماتریس دما زمان اجرای کرنل Y-۲ با Y-۲_p در نسخه هم‌مکان الگوریتم روش ضمنی جهت‌متغیر برابر شده و این موجب کاهش زمان اجرای حل دستگاه معادلات به اندازه ۶۳ میکروثانیه می‌شود. با این حال در این الگوریتم کرنل T به اندازه ۱۱۶ میکروثانیه به زمان اجرا افزوده است. بنابراین کل زمان حل از ۱۱۸۰ به ۱۲۳۳ افزایش یافته و در مجموع استفاده از نسخه هم‌مکان الگوریتم روش ضمنی جهت‌متغیر به جای نسخه غیرهم‌مکان نتوانسته بهبودی در سرعت اجرای روش ضمنی جهت‌متغیر روی پردازنده گرافیکی ایجاد نماید. در حقیقت تاثیر دسترسی غیرهم‌مکان به حافظه سراسری در روش ضمنی جهت‌متغیر قابل توجه نیست و رفع آن از طریق افزودن کرنل T به الگوریتم بصرفه نمی‌باشد.

با این حال در پیاده‌سازی روش تقسیم دامنه صریح- ضمنی جهت‌متغیر روی پردازنده گرافیکی نمی‌توان از تاثیر دسترسی غیرهم‌مکان به حافظه سراسری چشم‌پوشی نمود. همانگونه که در شکل ۱۱- ج نشان داده شده است، در نسخه غیر هم‌مکان الگوریتم روش تقسیم دامنه صریح- ضمنی جهت‌متغیر کرنل Y-۲ ۸۲ میکروثانیه و کرنل X-۲ ۲۲۹ میکروثانیه می‌باشد. بنابراین دسترسی غیرهم‌مکان به حافظه سراسری زمان حل را ۲/۸ برابر افزایش می‌دهد. در نسخه هم‌مکان الگوریتم روش تقسیم دامنه صریح- ضمنی جهت‌متغیر دسترسی هم‌مکان به حافظه سراسری زمان حل دستگاه معادلات در راستای X را به اندازه ۱۴۶ میکروثانیه کاهش داده و استفاده از کرنل T نیز ۱۱۶ میکروثانیه به زمان حل افزوده است. در مجموع دسترسی هم‌مکان به حافظه سراسری در حل دستگاه معادلات توانسته ۷۴ میکروثانیه از زمان حل بکاهد و موجب افزایش سرعت حل شود. در مجموع نتایج نشان می‌دهد که در روش ضمنی جهت‌متغیر استفاده از نسخه غیر هم‌مکان الگوریتم روش ضمنی جهت‌متغیر و در روش تقسیم دامنه صریح- ضمنی جهت‌متغیر استفاده از نسخه هم‌مکان الگوریتم روش تقسیم دامنه صریح- ضمنی جهت‌متغیر بصرفه می‌باشد.

در ادامه نسخه غیرهم‌مکان روش ضمنی جهت‌متغیر و نسخه هم‌مکان روش تقسیم دامنه صریح- ضمنی جهت‌متغیر با توجه به برتری به لحاظ زمان اجرا مبنای تحلیل‌های بعدی قرار می‌گیرند.

همانطورکه بیان شد، تقسیم دامنه می‌تواند از طریق افزایش تعداد نخ‌های فعال در کرنل‌های مربوط به حل دستگاه معادلات موجب کاهش زمان حل شود. از طرفی تقسیم دامنه می‌تواند بر زمان اجرای سایر کرنل‌ها نیز اثر بگذارد. زمان مورد نیاز برای تخمین

مقادیر مرز داخلی موجب می‌شود که زمان محاسبه ضرایب دستگاه معادلات (کرنل Y-۱ و X-۱(Y-۱_p)) در روش تقسیم دامنه صریح- ضمنی جهت‌متغیر نسبت روش ضمنی جهت‌متغیر بیشتر باشد. همچنین روش تقسیم دامنه صریح- ضمنی جهت‌متغیر دارای مرحله دیگری برای اصلاح مقادیر روی مرز است که در کرنل Y-۳ و X-۳(Y-۳_p) به انجام می‌رسد. با افزایش تقسیمات دامنه تعداد نقاط مرز داخلی نیز افزایش یافته و زمان اجرای این کرنل‌ها نیز افزایش می‌یابد. بنابراین پارامتر nos می‌تواند علاوه بر زمان اجرای کرنل‌های مربوط به حل دستگاه معادلات زمان اجرای بخش‌های دیگر الگوریتم را نیز تغییر دهد. شکل ۱۲ تاثیر تعداد تقسیمات دامنه بر زمان اجرای بخش‌های مختلف نسخه هم‌مکان الگوریتم روش تقسیم دامنه صریح- ضمنی جهت‌متغیر را برای شبکه‌ای با اندازه ۲۵۶×۲۵۶ نشان می‌دهد. همانگونه که مشاهده می‌شود زمان محاسبه ضرایب دستگاه معادلات (کرنل Y-۱ و Y-1_p) با افزایش nos افزایش می‌یابد. با این حال تغییرات آن بسیار اندک است و تاثیر قابل‌توجهی بر زمان حل نخواهد داشت. با این حال پارامتر nos زمان اجرای کرنل‌های مربوط به حل دستگاه معادلات (کرنل Y-۲ و Y-2_p) را به شکل قابل‌ملاحظه‌ای تغییر می‌دهد و مقدار آن را از ۴۵۹ میکروثانیه برای $nos =$ ۲ به ۱۰۶٫۳ میکروثانیه برای $nos =$ ۳۲ کاهش می‌دهد که معادل کاهش ۷۷ درصدی زمان اجرا برای این کرنل‌ها است. از طرفی قابل توجه است که با تغییر nos از ۲ به ۴ زمان اجرای کرنل‌های Y-۲ و Y-2_p ۱۹۸ میکروثانیه کاهش داشته است حال آنکه با تغییر nos از ۱۶ به ۳۲ زمان اجرای این کرنل‌ها تنها ۲۲ میکروثانیه کاهش پیدا کرده است. این موضوع به این دلیل است که با افزایش nos به تدریج چندپردازنده‌ها نخ‌های کافی برای مشغول نگه‌داشتن هسته‌های کودا و پنهان‌کردن تقسیمات دامنه را به‌دست‌می‌آورند و افزودن تعداد بیشتری نخ محاسباتی تاثیر چندانی بر بهبود عملکرد کرنل‌های مذکور ندارد.

از طرفی زمان اجرای کرنل‌های مربوط به اصلاح مقادیر مرز داخلی (Y-۳ و Y-۳_p) با افزایش پارامتر nos افزایش می‌یابد و مقدار آن از ۱۸ میکروثانیه برای $nos =$ ۲ به ۶۲ میکروثانیه برای $nos =$ ۳۲ می‌رسد که معادل افزایش ۳/۴ برابری زمان اجرا برای این کرنل‌ها است.

با توجه به توضیحات ارائه‌شده، کل زمان اجرای الگوریتم نیز وابسته به تعداد تقسیمات دامنه حل است. همانگونه که در شکل ۱۲ ارائه شده است کل زمان اجرای الگوریتم از ۸۲۳ میکروثانیه در $nos =$ ۲ به ۵۲۳ میکروثانیه در $nos =$ ۳۲ تغییر یافته است که معادل کاهش ۳۷ درصدی زمان حل است و به صورت عمده ناشی از کاهش زمان اجرای کرنل‌های Y-۲ و Y-2_p با افزایش تعداد تقسیمات دامنه می‌باشد. با این حال همانطور که مشاهده می‌شود زمان اجرای الگوریتم با تغییر تعداد تقسیمات دامنه از ۱۶ به ۳۲ افزایش یافته است. این موضوع به این دلیل است که با افزایش nos از ۱۶ به ۳۲ افزایش زمان اجرای کرنل‌های Y-۳ و Y-3_p بر کاهش زمان اجرای Y-۲ و Y-2_p غلبه کرده و در مجموع زمان اجرای الگوریتم افزایش یافته است.

شکل ۱۳ تاثیر اندازه شبکه و تعداد تقسیم دامنه را بر عملکرد نسخه هم‌مکان الگوریتم روش تقسیم دامنه صریح- ضمنی جهت‌متغیر نشان می‌دهد. برای اندازه‌گیری عملکرد روش ارائه‌شده میزان افزایش سرعت آن به نسبت روش ضمنی جهت‌متغیر مورد بررسی قرار گرفته است. نسخه غیرهم‌مکان الگوریتم روش ضمنی جهت‌متغیر باتوجه به عملکرد بهتر آن نسبت به نسخه هم‌مکان، به عنوان معیار مقایسه در نظر گرفته شده است. پارامتر (افزایش سرعت) Sp به این صورت تعریف می‌شود:

$$Sp(N_x, N_y, nos) = \frac{ADI\_un\_Time(N_x, N_y)}{ADI-CEIDD\_co\_Time(N_x, N_y, nos)} \tag{۱۵}$$

که در آن ADI_un_Time و ADI-CEIDD_co_Time به ترتیب زمان اجرای نسخه غیرهم‌مکان الگوریتم روش ضمنی جهت‌متغیر و نسخه هم‌مکان روش تقسیم دامنه صریح- ضمنی جهت‌متغیر به ازای ۶۰۰۰۰ گام زمانی است که تابعی از اندازه شبکه و تعداد تقسیمات شبکه می‌باشد. این زمان شامل یک بار کپی کل اطلاعات مقادیر اولیه دامنه‌ها در حافظه دستگاه و یک بار انتقال پاسخ در گام زمانی نهایی از حافظه دستگاه به میزبان نیز می‌باشد. پارامتر Sp نشان‌دهنده میزان مزیت بهره‌گیری از روش تقسیم دامنه صریح- ضمنی جهت‌متغیر به جای روش ضمنی جهت‌متغیر است و نشان می‌دهد که در چه محدوده‌ای از اندازه شبکه و با چه تعداد تقسیم دامنه‌ای می‌توان از روش تقسیم دامنه صریح- ضمنی جهت‌متغیر بهره بیشتری برد.

همانگونه که در شکل ۱۳ مشاهده می‌شود تقسیم دامنه تاثیر قابل‌ملاحظه‌ای بر عملکرد روش تقسیم دامنه صریح- ضمنی جهت‌متغیر دارد. همانطور که مشاهده می‌شود به صورت عمده با افزایش nos پارامتر Sp افزایش یافته است. با این حال در اندازه شبکه ۲۵۶×۲۵۶ و ۵۱۲×۵۱۲ با افزایش nos از ۱۶ به ۳۲ پارامتر Sp کاهش یافته است. همانگونه که قبل از این بیان شد این موضوع به این دلیل است که با افزایش nos از ۱۶ به ۳۲ افزایش زمان اجرای کرنل‌های Y-۳ و Y-3_p بر کاهش زمان اجرای Y-۲ و Y-2_p غلبه کرده و در مجموع زمان اجرای الگوریتم افزایش یافته و مزیت بهره‌گیری از آن در برابر روش ضمنی جهت‌متغیر کاهش می‌یابد.

پارامتر Sp در اندازه شبکه ۶۴×۶۴، ۱۲۸×۱۲۸ و ۲۵۶×۲۵۶ در محدوده ۱/۴۷ تا ۲/۶۳ قرار دارد. با این حال در اندازه شبکه ۵۱۲×۵۱۲ به ازای تمامی مقادیر nos پارامتر Sp کاهش یافته و در محدوده ۱/۳۳ تا ۱/۶۹ می‌باشد. این کاهش مزیت نسخه هم‌مکان الگوریتم روش تقسیم دامنه صریح- ضمنی جهت‌متغیر نسبت به نسخه غیر هم‌مکان الگوریتم روش ضمنی جهت‌متغیر به این دلیل است که در اندازه شبکه ۵۱۲×۵۱۲ کرنل Y-۲ و X-۲ در نسخه غیر هم‌مکان الگوریتم روش ضمنی جهت‌متغیر تعداد نخ کافی برای مشغول نگه‌داشتن هسته‌های کودا و پنهان‌کردن حافظه را برخوردار است و تقسیم دامنه با هدف افزایش تعداد نخ‌ها تغییر چندانی در زمان اجرای این کرنل ایجاد نمی‌کند. این موضوع از طرفی نشان‌دهنده این است که روش تقسیم دامنه صریح- ضمنی جهت‌متغیر در موارد بهره‌گیری از پردازنده‌های بزرگ با تعداد زیادی چندپردازنده می‌تواند مزیت بیشتری ارائه نماید.

۶- نتیجه‌گیری

رویکرد جدید تقسیم دامنه صریح- ضمنی جهت‌متغیر با ترکیب روش ضمنی جهت‌متغیر و روش تقسیم دامنه صریح- ضمنی برای حل معادله انتقال حرارت هدایت دو بعدی روی پردازنده گرافیکی ارائه شده است. در این روش تخمین مقادیر مرزی با یک طرح عددی صریح صورت گرفته و برای حل درون زیردامنه‌ها از یک طرح ضمنی برمبنای روش ضمنی جهت‌متغیر استفاده می‌شود. سپس از یک طرح ضمنی برای تصحیح مقادیر روی مرز استفاده می‌شود. در روش ضمنی جهت‌متغیر تعداد دستگاه معادلات مستقل تشکیل‌شده در هر گام زمانی محدود است به نحوی که امکان فعال‌سازی تعداد زیادی نخ برای مشغول نگه‌داشتن پردازنده گرافیکی را فراهم نمی‌سازد. در روش تقسیم دامنه صریح- ضمنی جهت‌متغیر تقسیم دامنه تعداد دستگاه معادلات سه‌قطری مستقل را چندین برابر افزایش می‌دهد و این موضوع موجب افزایش تعداد نخ‌های محاسباتی فعال برای حل دستگاه معادلات در این روش نسبت به روش ضمنی جهت‌متغیر می‌شود. در تحقیق حاضر عملکرد روش ارائه‌شده به لحاظ دقت و سرعت مورد تحلیل قرار گرفته است. اهم نتایج حاصل از تحقیق حاضر شامل موارد زیر می‌باشد:

۱. خطای روش ضمنی جهت‌متغیر نسبت به روش تقسیم دامنه صریح- ضمنی جهت‌متغیر کمتر است.
۲. در $\lambda = 1$ تفاوت خطای روش تقسیم دامنه صریح- ضمنی جهت‌متغیر و روش ضمنی جهت‌متغیر در کمترین مقدار خود قرار دارد و با افزایش یا کاهش λ این تفاوت افزایش می‌یابد.
۳. افزایش تعداد تقسیمات دامنه (nos) خطای روش تقسیم دامنه صریح- ضمنی جهت‌متغیر را افزایش می‌دهد.
۴. نتایج آزمایشات عددی نشان‌دهنده پایداری بسیار بالای این روش است.
۵. در روش تقسیم دامنه صریح- ضمنی جهت‌متغیر زمان حل دستگاه معادلات روی پردازنده گرافیکی به نحو قابل‌ملاحظه‌ای کاهش می‌یابد. برای شبکه‌ای با اندازه ۲۵۶×۲۵۶ در روش تقسیم دامنه صریح- ضمنی جهت‌متغیر به ازای $nos = ۸$ زمان حل دستگاه معادلات ۶۳ درصد نسبت به روش ضمنی جهت‌متغیر کاهش می‌یابد.
۶. روش تقسیم دامنه صریح- ضمنی جهت‌متغیر محاسبات بیشتری نسبت به روش ضمنی جهت‌متغیر دارد که مربوط به تخمین و تصحیح مقادیر مرزهای داخلی می‌شود. این محاسبات با افزایش پارامتر nos تاثیر قابل‌ملاحظه‌ای بر زمان اجرای الگوریتم دارد به طوریکه در مقادیر nos بالا افزایش این پارامتر با افزودن به زمان اجرای کرنل Y-۳ و Y-3_p می‌تواند موجب افت عملکرد الگوریتم شود.
۷. روش تقسیم دامنه صریح- ضمنی جهت‌متغیر باعث افزایش سرعت حل نسبت به روش ضمنی جهت‌متغیر می‌شود. با افزایش پارامتر nos مقدار پارامتر Sp افزایش می‌یابد با این حال در اندازه

شبکه‌های بزرگ افزودن تعداد تقسیمات بیش از حد مشخصی می‌تواند موجب افت عملکرد این روش شود.

در اندازه شبکه‌های بزرگ از مزیت روش تقسیم دامنه صریح-ضمنی جهت‌متغیر نسبت به روش ضمنی جهت‌متغیر کاسته می‌شود چراکه برای اندازه شبکه‌های بزرگ در روش ضمنی جهت‌متغیر می‌توان تعداد نخ کافی را برای مشغول نگه‌داشتن پردازنده فراهم نمود و تقسیم دامنه نقش قابل‌توجهی ایفا نمی‌کند. این موضوع از طرفی نشان‌دهنده این است که روش تقسیم دامنه صریح- ضمنی جهت‌متغیر در موارد بهره‌گیری از پردازنده‌های بزرگ با تعداد زیادی چندپردازنده می‌تواند مزیت بیشتری ارائه نماید.

تحقیق حاضر می‌تواند عرصه گسترده‌ای را برای تحقیقات آتی فراهم نماید. از جمله می‌توان به موارد زیر اشاره کرد:

۱. بهینه‌سازی بخش‌های مختلف الگوریتم تقسیم دامنه صریح-ضمنی جهت‌متغیر از جمله کرنل حل دستگاه معادلات و کرنل ترانهاده‌کردن ماتریس میدان می‌تواند در تحقیقات آینده مورد توجه قرار گیرد.

۲. روش تقسیم دامنه صریح- ضمنی جهت‌متغیر در آزمایشات عددی همگرایی و پایداری بالایی را نشان می‌دهد با این حال اثبات ریاضی پایداری بدون قید و شرط روش تقسیم دامنه صریح- ضمنی جهت‌متغیر برای حل مساله انتقال حرارت هدایت باید در تحقیقات آینده مورد بررسی قرار گیرد.

۳. پیاده‌سازی و تحلیل روش تقسیم دامنه صریح- ضمنی جهت‌متغیر برای حل مسائل دیگری نظیر مساله جابجایی- پخش و یا هدایت غیر فوریه‌ای[1] می‌تواند موضوع مطالعات آینده باشد.

فهرست علائم

علائم انگلیسی

c	ظرفیت گرمایی ویژه، $kJ/(K.kg)$
E	خطا
E_2^n	نرم ۲ خطا
E_∞^n	نرم بی‌نهایت خطا
k_x	هدایت حرارتی در راستای X، $kW/(K.m)$
k_y	هدایت حرارتی در راستای Y، $kW/(K.m)$
N_x	تعداد گره‌های شبکه در راستای X
N_y	تعداد گره‌های شبکه در راستای Y
nos	تعداد زیردامنه‌ها
Sp	افزایش سرعت
T	دما، K

علائم یونانی

ρ	چگالی، kg/m^3
λ	ضریب شبکه، $kJ.m/K$

زیرنویس

i	شمارنده گره شبکه در راستای X
j	شمارنده گره شبکه در راستای Y

بالانویس

n	شمارنده گام زمانی

پیوست ۱. پردازش بهینه روی پردازنده گرافیکی

در چندپردازنده هر بلاک به دسته‌های ۳۲ عددی از نخ‌ها به نام وارپ تقسیم می‌شوند. نخ‌های داخل یک وارپ، دستوالعملی کاملا یکسان را به صورت هم‌زمان اجرا می‌کنند. اگر در بین این ۳۲ نخ، تفاوتی به لحاظ دستوالعمل‌های تخصیص داده شده از سوی کاربر وجود داشته باشد، محاسبات با افت عملکرد روبرو می‌شود. در چنین وضعیتی تمامی این ۳۲ نخ دستورالعمل یکسانی را اجرا می‌کنند؛ اما اگر بخشی از این دستوالعمل‌ها برای یک یا چند نخ معتبر نباشد، آن بخش از پردازش نادیده گرفته می‌شود. از طرفی اگر تعداد نخ‌های یک بلاک مضربی از ۳۲ نباشد باز هم تمامی نخ‌ها باید در دسته‌های ۳۲ تایی دسته‌بندی می‌شوند. در این مواقع چند نخ در انتهای بلاک باقی می‌مانند که تشکیل یک دسته ۳۲ تایی را نمی‌دهند. کودا در این مواقع چند نخ فرضی که حاوی محاسبه مفیدی نیستند را به دسته آخر اضافه می‌کند تا یک وارپ تکمیل شود که این موضوع موجب هدررفتن بخشی از پردازش و افت عملکرد است. بنابراین باید سعی شود که تا حد امکان هر ۳۲ نخ مجاور دستوالعمل یکسانی را انجام دهند و از طرفی تعداد نخ‌های یک بلاک مضربی از ۳۲ باشد.

همانگونه که بیان شد چندپردازنده‌ها برای مشغول نگه‌داشتن هسته‌های کودا و پنهان‌نمودن تاخیرات حافظه به تعداد زیادی

1 Non-Fuorier heat transfer

نخ (وارپ) نیاز دارند. بیشینه تعداد وارپ‌هایی که می‌تواند در یک چندپردازنده ساکن باشد با توجه به معماری آن مشخص می‌شود. این تعداد در پردازنده تحقیق حاضر ۶۴ عدد است. هرچه تعداد وارپ‌ها به این حد بیشینه نزدیک شود امکان مشغول‌نمودن هسته‌های کودا و پنهان‌کردن تاخیرات حافظه افزایش می‌یابد. به نسبت تعداد وارپ ساکن در یک چندپردازنده به حداکثر تعداد وارپ قابل سکونت در آن مشغولیت گفته می‌شود. عوامل متعددی می‌توانند روی تعداد وارپ‌های ساکن در چندپردازنده اثر بگذارند. یکی از این عوامل اندازه بلاک است. در معماری ماکسول تعداد حداکثر بلاک‌هایی که می‌توانند در یک چندپردازنده ساکن شوند ۳۲ عدد است. اگر هر یک از این ۳۲ بلاک حاوی ۶۴ نخ باشد آن‌گاه تعداد وارپ‌های فعال به ۳۲ عدد می‌رسد که با بیشینه ظرفیت چندپردازنده برابر است (مشغولیت= ۱). اگر اندازه بلاک‌ها کمتر از ۶۴ نخ باشد بخشی از ظرفیت چندپردازنده خالی مانده و مشغولیت کم می‌شود. یک عامل موثر دیگر مقدار حافظه رجیستری است که توسط هر نخ استفاده می‌شود. هر چندپردازنده مقدار محدودی حافظه ثبات دارد که بین نخ‌های فعال توزیع می‌شود. بنابراین تعداد نخ‌های ساکن در چندپردازنده متناسب با مقدار ثبات چندپردازنده و مقدار ثبات مورد نیاز برای هر نخ است. در معماری ماکسول مقدار ثبات چندپردازنده ۶۵۵۳۶ ثبات ۳۲ بیتی است. اگر هر نخ در یک کرنل خاص ۳۲ ثبات استفاده کند تعداد نخ‌های فعال برابر $\frac{65536}{32} = 2048$ خواهد بود و تعداد وارپ‌های فعال نیز برابر $\frac{2048}{32} = 64$ است که برابر با بیشنه تعداد وارپ‌های قابل سکونت در یک چندپردازنده است (مشغولیت= ۱). اگر نخ‌ها بیش از مقدار مذکور از حافظه ثبات استفاده کنند چندپردازنده با توجه به محدودیت حافظه تعداد نخ‌های ساکن را کاهش می‌دهد و مشغولیت کاهش می‌یابد. از جمله عوامل دیگر موثر در مشغولیت چندپردازنده میزان حافظه اشتراکی مورد استفاده در هر بلاک است که بررسی آن در محدوده اهداف تحقیق حاضر نیست.

یکی از مواردی که باید در پیاده‌سازی روش‌های عددی روی پردازنده گرافیکی به آن توجه شود، موضوع دسترسی بهینه به حافظه است. هر ۱۲۸ بایت پی‌درپی از حافظه سراسری با یک بار مراجعه توسط وارپ برای پردازش نخ‌ها در دسترس قرار می‌گیرد. دسترسی به حافظه در صورتی هم‌مکان است که تمام اطلاعات مورد نیاز نخ‌ها، در این ۱۲۸ 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 نخ‌های مجاور در یک وارپ اطلاعات نزدیک به هم را از یک آرایه مورد استفاده قرار دهند. دسترسی غیرهم‌مکان به حافظه موجب کاهش سرعت دسترسی می‌شود.

در این بخش توضیحاتی درباره ساختار پردازنده گرافیکی و نکات مهمی پیرامون بهره‌گیری بهینه از ظرفیت این سخت‌افزار بیان شد. در بخش ۴ با توجه به این نکات نحوه پیاده‌سازی روش تقسیم دامنه صریح- ضمنی جهت‌متغیر مورد بررسی قرار گرفته است.

مراجع

[1] K.E. Niemeyer, C.-J. Sung, Recent progress and challenges in exploiting graphics processors in computational fluid dynamics, The Journal of Supercomputing, 67(2) (2013) 528-564.

[2] G. Alfonsi, S.A. Ciliberti, M. Mancini, L. Primavera, GPGPU implementation of mixed spectral-finite difference computational code for the numerical integration of the three-dimensional time-dependent incompressible Navier–Stokes equations, Computers & Fluids, 102 (2014) 237-249.

[3] F. Salvadore, M. Bernardini, M. Botti, GPU accelerated flow solver for direct numerical simulation of turbulent flows, Journal of Computational Physics, 235 (2013) 129-142

[4] S. Vanka, A.F. Shinn, K.C. Sahu, Computational Fluid Dynamics Using Graphics Processing Units: Challenges and Opportunities, Fluids and Thermal Systems; Advances for Process Industries, (2011) 429-437.

[5] L. Deng, H. Bai, F. Wang, Q. Xu, Cpu/Gpu Computing for an Implicit Multi-Block Compressible Navier-Stokes Solver on Heterogeneous Platform, International Journal of Modern Physics: Conference Series, 42 (2016) 1660163.

[6] G. Borrell, J.A. Sillero, J. Jiménez, A code for direct numerical simulation of turbulent boundary layers at high Reynolds numbers in BG/P supercomputers, Computers & Fluids, 80 (2013) 37-43.

[7] N. Sakharnykh, Tridiagonal solvers on the GPU and

on Numerical Analysis, 44(4) (2006) 1584-1611.

[13] L. Zhu, An Explicit-Implicit Predictor-Corrector Domain Decomposition Method for Time Dependent Multi-Dimensional Convection Diffusion Equations, Numerical Mathematics: Theory, Methods and Applications, 2(3) (2009) 301-325.

[14] H. Liao, H. Shi, Z. Sun, Corrected explicit-implicit domain decomposition algorithms for two-dimensional semilinear parabolic equations, Science in China Series A: Mathematics, 52(11) (2009) 2362-2388.

[15] C. Du, D. Liang, An efficient S-DDM iterative approach for compressible contamination fluid flows in porous media, Journal of Computational Physics, 229(12) (2010) 4501-4521.

[16] D. Liang, C. Du, The efficient S-DDM scheme and its analysis for solving parabolic equations, Journal of Computational Physics, 272 (2014) 46-69.[17]

[17] Z. Zhou, D. Liang, Y. Wong, The new mass-conserving S-DDM scheme for two-dimensional parabolic equations with variable coefficients, Applied Mathematics and Computation, 338 (2018) 882-902.

applications to fluid simulation, in: NVIDIA GPU Technology Conference, San Jose, California, USA, 2009, pp. 17-19.

[8] S. Ha, J. Park, D. You, A GPU-accelerated semi-implicit fractional-step method for numerical solutions of incompressible Navier–Stokes equations, Journal of Computational Physics, 352 (2018) 246-264.

[9] C.N. Dawson, Q. Du, T. F. Dupont, A finite difference domain decomposition algorithm for numerical solution of the heat equation, mathematics of computation, 57 (1991) 63-71.

[10] Q. Du, Mu, M, Wu, Z N, Efficient parallel algorithms for parabolic problems, SIAM Journal on Numerical Analysis, 39(5) (2002) 1469-1487.

[11] X.-h. Sun, Y. Zhuang, stablized explici-implicit domain decomposition method for the numerical solution of finit difference equation, SIAM Journal on Numerical Analysis, 24(1) (2002) 335-358.

[12] H.S. Shi, H.-L. Liao, Unconditional Stability of Corrected Explicit-Implicit Domain Decomposition Algorithms for Parallel Approximation of Heat Equations, SIAM Journal

چگونه به این مقاله ارجاع دهیم

A. Foadaddini1, S. A. Zolfaghari, H. Mahmoodi Darian. Developing an alternating direction explicit-implicit domain-decomposition approach to solve heat transfer equation on graphics processing unit. Amirkabir J. Mech Eng., 53(special issue 3) (2021). 1915-1936.

DOI: 10.22060/mej.2019.16178.6295

این صفحه عمداً خالی است