



پیاده سازی پردازش موازی روی کارت گرافیک برای شبیه سازی جریان سیال با روش شبکه بولتزمن و نمایه هموار

بهنام خلیلی¹، محمد رهنما^{2*}، سعید جعفری³، ابراهیم جهانشاهی جواران⁴

1- کارشناس ارشد، مهندسی مکانیک، دانشگاه شهید باهنر کرمان، کرمان

2- استاد، مهندسی مکانیک، دانشگاه شهید باهنر کرمان، کرمان

3- استادیار، مهندسی نفت، دانشگاه شهید باهنر کرمان، کرمان

4- استادیار، مهندسی مکانیک، دانشگاه تحصیلات تکمیلی صنعتی و فناوری پیشرفته، کرمان

* کرمان، صندوق پستی 76169-133، rahnama@uk.ac.ir

اطلاعات مقاله

مقاله پژوهشی کامل

دریافت: 25 خرداد 1395

پذیرش: 30 مرداد 1395

ارائه در سایت: 11 مهر 1395

کلید واژگان:

روش شبکه بولتزمن

روش نمایه هموار

پردازش موازی

برهم کنش جامد- سیال

چکیده

بررسی برهم کنش میان ذرات جامد و سیال به عنوان مقدمه‌ای بر شبیه سازی بسیاری از مسائل مهندسی مانند بسترهای سیالی، ته نشینی ذرات و جوهر کاتالیست در سلول‌های سوختی مورد بررسی قرار گرفته است. یکی از روش‌های مناسب برای انجام این گونه شبیه سازی‌ها، ترکیب دو روش شبکه بولتزمن و نمایه هموار می‌باشد که دارای یک الگوریتم مناسب برای اجرا شدن به صورت موازی می‌باشند. روش نمایه هموار همانند روش شبکه بولتزمن از یک شبکه ثابت برای شبیه سازی ذرات جامد در سیال استفاده می‌کند و از این رو یک روش کارآمد برای پردازش موازی به کمک کارت گرافیک می‌باشد. در کار حاضر، پیاده سازی یک الگوریتم مناسب برای موازی سازی ترکیب دو روش نمایه هموار و شبکه بولتزمن روی کارت گرافیک ارائه می‌شود. به منظور بررسی صحت نتایج، ابتدا جریان سیال درون کانال مورد بررسی قرار گرفت. نتایج زمانی حاکی از آن بود که زمان حل می‌تواند تا 80 برابر بوسیله کارت گرافیک کاهش یابد. در ادامه نیروی پسای وارد بر یک کره در جریان سیال و همچنین شبیه سازی سقوط یک ذره در سیال ساکن بر اثر نیروی وزن مورد بررسی قرار گرفت. نتایج بدست آمده بر روی کارت گرافیک، نشان دهنده افزایش توان محاسباتی تا 6.5 میلیون گره محاسباتی در واحد زمان را نشان می‌دهد.

Implementation of parallel processing on GPU for fluid flow simulation using Lattice Boltzmann method and Smoothed Profile method

Behnam Khalili¹, Mohammad Rahnema^{1*}, Saeed Jafari², Ebrahim Jahanshahi Javaran³

1- Department of Mechanical Engineering, Shahid Bahonar University of Kerman, Kerman, Iran.

2- Department of Petroleum Engineering, Shahid Bahonar University of Kerman, Kerman, Iran.

3- Department of Energy, Graduate University of Advanced Technology, Kerman, Iran.

* P.O.B.76169-133, Kerman, Iran, rahnama@uk.ac.ir

ARTICLE INFORMATION

Original Research Paper
Received 14 June 2016
Accepted 20 August 2016
Available Online 02 October 2016

Keywords:
Lattice Boltzmann method
Smoothed Profile method
Parallel processing
Fluid- solid interaction

ABSTRACT

Investigation of fluid-solid interaction has been studied as an introduction to simulate a wide range of engineering problems such as fluidized beds, sediment transportation and catalyst inks in fuel cells. An efficient method for performing simulation is a combination of Lattice Boltzmann method (LBM) and Smoothed Profile method (SPM). In addition, the operations in the SPM are local; it can be easily programmed for parallel processing. In this approach, the flow is computed on fixed Eulerian grids which are also used for the particles. Owing to the use of the same grids for simulation of fluid flow and particles, this method is highly efficient for the purpose of parallel processing by means of GPU. In this study, a combination of Lattice Boltzmann method (LBM) and Smoothed Profile method has been implemented in parallel processing on GPU. For validation purpose, the fluid flow within a channel was investigated. Results suggest that computational time can be reduced up to 80 times by means of GPU. Then, drag force exerted on a sphere in fluid flow and the sedimentation of one sphere in a quiescent fluid were studied. Results show that performance of GPU can be increased up to 6.5 million fluid nodes per second by using this method.

1-مقدمه

اند. از این رو همواره تلاش کرده‌اند که از روش‌های محاسباتی استفاده کنند که قابلیت پردازش موازی را داشته باشد تا بتوانند بالاترین بازدهی را از آنها کسب کنند.

در چند دهه گذشته، روش شبکه بولتزمن به دلیل داشتن یک الگوریتم آسان برای پردازش موازی توجه زیادی را به‌عنوان یک روش مناسب برای

در سال‌های اخیر ظهور پردازنده‌های موازی درجه‌ای جدید برای بررسی مسائل پیچیده را فراهم نموده و پردازش موازی جایگاه ویژه‌ای در علوم مختلف پیدا کرده است. بسیاری از محققان که در زمینه روش‌های عددی کار می‌کنند همواره با مشکل سنگینی و زمان بر بودن محاسبات درگیر بوده-

Please cite this article using:

B. Khalili, M. Rahnema, S. Jafari, E. Jahanshahi Javaran, Implementation of parallel processing on GPU for fluid flow simulation using Lattice Boltzmann method and Smoothed Profile method, *Modares Mechanical Engineering*, Vol. 16, No. 9, pp. 449-458, 2016 (in Persian)

برای ارجاع به این مقاله از عبارت ذیل استفاده نمایید:

یکی از این روش‌ها، روش نمایه هموار⁴ است که در سال 2005، توسط ناکایاما و یاماموتو [11] ارائه شد. این روش از یک شبکه اولیری ثابت برای سیال میزبان استفاده می‌کند. در این روش، به جای استفاده از شرط مرزی در سطح مشترک جامد و سیال، سطح جامد با استفاده از یک نیروی حجمی هموار مشخص در معادلات ناویر-استوکس نمایش داده می‌شود. روش نمایه هموار یک معادله را در کل ناحیه حل شامل حجم اجسام جامد بدون هیچ شرط مرزی داخلی حل می‌کند. برای نمایش مرز جامد، از یک لایه مشترک بین جامد و سیال استفاده می‌شود که این لایه به طور هموار از سمت جسم جامد به طرف سیال گسترده می‌شود یا به عبارتی، از این لایه برای گذر از حرکت جسم صلب به حرکت سیال استفاده می‌شود. این فصل مشترک بین جامد صلب و سیال، حجم معینی دارد که در آن چندین نقطه شبکه قرار گرفته است. با استفاده از این تغییر ساده، مختصات کارتزین معمولی می‌تواند برای سیستم‌های پیچیده با هر شکل دلخواه استفاده شود. بدلیل ویژگی‌های مشترک روش نمایه هموار و روش شبکه بولتزمن مبنی بر استفاده از یک شبکه کارتزین ثابت، ترکیب روش شبکه بولتزمن و نمایه هموار برای اولین بار در سال 2011 توسط جعفری و همکاران انجام شد [12].

روش نمایه هموار به دلیل داشتن ویژگی‌های مناسب از جمله الگوریتم ساده و استفاده از یک شبکه کارتزین ثابت همانند روش شبکه بولتزمن، یک روش مناسب برای پردازش موازی می‌باشد. از این رو در پژوهش حاضر یک روش کارآمد برای پیاده سازی ترکیب دو روش شبکه بولتزمن و نمایه هموار بر روی کارت گرافیک ارائه می‌شود. در ادامه مختصری از روش‌های شبکه بولتزمن و نمایه هموار ارائه خواهد شد و ساختار کارت گرافیک برای پردازش موازی شرح داده می‌شود. در نهایت نتایج و نتیجه گیری مورد بررسی قرار می‌گیرند.

2- روش شبکه بولتزمن

برای شبیه سازی جریان سیال از روش شبکه بولتزمن (سه بعدی) استفاده شده است. در این روش کمیت‌های ماکروسکوپی از محاسبه تابع توزیع احتمال بدست می‌آیند که این تابع توسط حل معادله بولتزمن محاسبه می‌شود. بر اساس روش ارائه شده توسط بهاتنگار، گروس و کروک [13] (BGK) ترم برخورد در معادله بولتزمن ساده شده و معادله مورد نظر به صورت زیر تبدیل می‌گردد.

$$f_i(x + c_i \Delta t, t + \Delta t) = f_i(x, t) - \frac{1}{\tau} (f_i(x, t) - f_i^{eq}(x, t)) \quad (1)$$

که در معادله بالا τ زمان آرامش، c_i بردار یکه سرعت، $f_i(x, t)$ تابع توزیع احتمال و $f_i^{eq}(x, t)$ تابع توزیع احتمال تعادلی می‌باشد. زمان آرامش در معادله بولتزمن به صورت زیر محاسبه می‌گردد.

$$\tau = \frac{\eta}{c_s^2 \delta t} + 0.5 \quad (2)$$

همانطور که ذکر شد، اندازه بردارهای سرعت شبکه، با توجه به مدل انتخابی برای شبکه مشخص می‌شوند. در کار حاضر از یک مدل سه بعدی با نوزده مولفه سرعت، مدل D_3Q_{19} ، که معروف ترین مدل در محاسبات سه بعدی می‌باشد، برای شبیه سازی استفاده می‌شود (شکل 1).

مولفه‌های بردار سرعت برای این مدل به صورت زیر تعریف می‌گردند:

$$c_i = \begin{cases} (0, 0, 0), & i = 0 \\ (\pm 1, 0, 0), (\pm 1, 0, 1), (\pm 1, 0, -1), & i = 1-6 \\ (\pm 1, \pm 1, 0), (\pm 1, \pm 1, 1), (\pm 1, \pm 1, -1), & i = 7-18 \end{cases} \quad (3)$$

شبیه سازی انواع جریان‌های مختلف از جمله جریان‌هایی با هندسه‌های پیچیده به خود جلب کرده است [1]. روش شبکه بولتزمن مشکلات روش‌های معمول دینامیک سیالات محاسباتی از جمله تولید مش در هر تکرار را ندارد زیرا از یک شبکه محاسباتی اولیری و ثابت برای حل جریان سیال استفاده می‌کند. به دلیل اینکه این روش نیاز به تولید شبکه محاسباتی در هر گام زمانی ندارد برنامه نویسی آن آسان بوده و می‌توان آن را به راحتی روی پردازنده‌های موازی همانند کارت گرافیک برنامه نویسی کرد [2]. واحد پردازنده گرافیکی¹ برای پردازش کارهای گرافیکی که حاوی محاسباتی ساده اما با تعداد زیاد می‌باشد، طراحی شده است. بنابراین کارت‌های گرافیک قادر به انجام تعداد زیادی محاسبه شبیه به هم اما با تعداد بالا هستند. در چند دهه اخیر به خاطر این ویژگی پردازنده‌های گرافیکی، از آنها برای انجام محاسبات غیر گرافیکی نیز استفاده می‌شود. از جمله تفاوت‌های میان واحد پردازنده مرکزی² و واحد پردازنده گرافیکی، آن است که واحد پردازنده مرکزی مجموعه‌ای کوچک از هسته‌های بسیار قوی است در حالی که واحد پردازنده گرافیکی مجموعه‌ای بزرگ (تا چند هزار هسته) از هسته‌های نسبتاً قوی می‌باشد که این موضوع امکان پردازش موازی با تعداد بالا را فراهم می‌کند. از این رو محققین بسیاری حل شبکه بولتزمن را با استفاده از کارت گرافیک مورد بررسی قرار داده‌اند. در این رابطه، کوزنیک و همکاران [3] مدلی را برای اجرا کردن بخش‌های مختلف روش شبکه بولتزمن روی کارت گرافیک ارائه دادند. در ادامه ریگل و همکاران [4] از کارت گرافیک برای شبیه‌سازی جریان حول کره با استفاده از روش شبکه بولتزمن استفاده کردند. تولکی و همکاران [5] نیز یک کد دو بعدی بر اساس روش شبکه بولتزمن را به کمک کارت گرافیک اجرا کرده و نتایج زمانی آن را مقایسه کردند. آنها با انجام این مقایسه روی یک مسئله یکسان به این نتیجه رسیدند که به کمک کارت گرافیک می‌توان زمان محاسبات را تا 23 برابر کاهش داد.

یکی از کاربردهای روش شبکه بولتزمن، شبیه سازی مسائلی است که در آنها اندرکنش بین ذرات جامد و سیال وجود دارد. به دلیل کاربرد وسیع این مسائل در صنعت و طبیعت، بررسی آنها از اهمیت ویژه‌ای برخوردار می‌باشد. مهمترین موضوع در شبیه سازی چنین مسائلی ارضای شرط مرزی عدم لغزش در سطح مشترک جامد و سیال می‌باشد. این موضوع توسط محققین زیادی مورد بررسی قرار گرفته است [6, 7]. از پیشگامان شبیه‌سازی ذرات جامد در سیال به کمک روش شبکه بولتزمن می‌توان به لد و همکاران [8, 9] در سال 1994 اشاره کرد. لد و همکاران از شرط مرزی کمانه کردن³ برای اعمال شرط مرزی عدم لغزش استفاده کردند. شرط مرزی کمانه کردن ساده-ترین و پر استفاده‌ترین روش برای اعمال شرط مرزی عدم لغزش در روش شبکه بولتزمن می‌باشد. از این رو اوبرت و همکاران [10] یک روش کارآمد را برای موازی سازی شرط مرزی کمانه کردن روی کارت گرافیک ارائه دادند اما مهم‌ترین مشکل این روش هنگام شبیه‌سازی شکل‌های پیچیده می‌باشد که باعث به وجود آمدن نوساناتی در نیروی اعمالی محاسبه شده روی ذره می‌گردد.

از این رو به منظور رفع مشکل شرط مرزی کمانه کردن، روش‌های دیگری برای اعمال شرط مرزی عدم لغزش ارائه شد. در این روش‌ها یک نیروی حجمی به معادله حرکت سیال اضافه می‌شود که این نیرو نقش ذرات جامد در سیال را ایفا می‌کند و مانند یک شرط مرزی برای سیال می‌باشد.

¹GPU

²CPU

³Bounce Back

⁴Smoothed Profile Method (SPM)

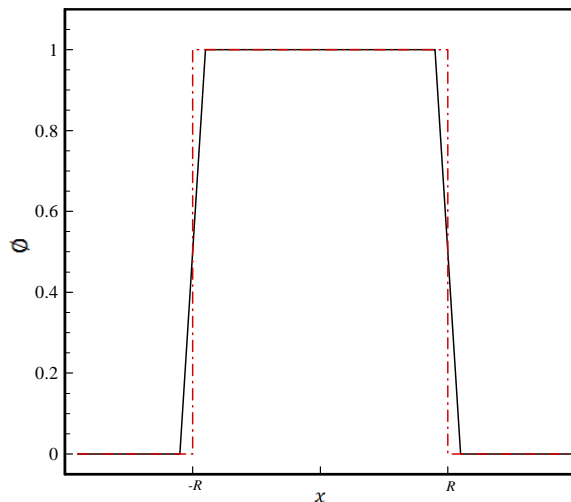


Fig. 2 Representation of a particle in smoothed profile (solid line)

شکل 2 نمایش یک ذره جامد به کمک روش نمایه هموار (خط جامد)

$$\phi(x, t) = \sum_{i=1}^{N_p} \phi_i(x, t) \quad (7)$$

در این رابطه، $\phi_i(x, t)$ که مقداری بین صفر و یک دارد، تابع موقعیت ذره i ام می باشد و N_p ، تعداد ذرات جامدی است که در سرتاسر ناحیه حل قرار دارند. توابع تحلیلی مختلفی از این نمایه هموار برای ذرات کروی شکل وجود دارد که پر استفاده ترین تابع به صورت زیر بیان می شود:

$$\phi_i(x, t) = \begin{cases} 0, & x < -\zeta/2 \\ \frac{1}{2} \sin\left(\frac{\pi x}{\zeta_i} + 1\right), & |x| < \zeta/2 \\ 1, & x > \zeta/2 \end{cases} \quad (8)$$

در این رابطه، R شعاع هر ذره و $R_i(t)$ و ζ_i به ترتیب بردار موقعیت مرکز ذره و ضخامت سطح مشترک ذره i ام می باشند.

بر اساس تابع موقعیت ذرات جامد، میدان سرعت ذرات جامد بصورت معادله زیر تعریف می شود:

$$\phi(x, t) u_p(x, t) = \sum_{i=1}^{N_p} \phi_i(x, t) [U_{c_i}(t) + \omega_i \times \{x - R_i(t)\}] \quad (9)$$

در آن $\{U_{c_i}, \omega_i\} (i = 1, \dots, N_p)$ به ترتیب، سرعت خطی مرکز جرم و سرعت زاویه ای ذره i ام می باشند. جریان سیال روی یک ذره جامد منجر به اعمال نیروهای عمودی و برشی روی این ذره بوسیله سیال و متقابلاً توسط ذره جامد روی سیال می شود که از این نیروها به نیروهای اندرکنش هیدرودینامیکی جامد-سیال یاد می شود. این نیروها سیال مجاور به سطح ذره جامد را وادار به حرکت با سرعت سطح جامد می کنند که همان شرط مرزی عدم لغزش می باشد. حال اگر بتوان یک نیروی خارجی معادل با این نیروی اندرکنش هیدرودینامیکی به گونه ای به سیال وارد کرد که سیال بتواند در ناحیه متعلق به جسم جامد با سرعتی برابر با سرعت جسم جامد حرکت کند، در این صورت می توان حرکت سیال حول جسم جامد را بدون هیچ شرط مرزی شبیه سازی کرد. روش نمایه هموار بر پایه چنین ایده ای استوار است. به بیان دقیق تر، گره های سیالی که بوسیله ذره جامد پوشیده شده اند (گره-های سیال مجازی) باید سرعتی برابر سرعت ذره جامد داشته باشند. بدین منظور، یک نیروی حجمی به کل میدان سیال وارد می شود تا سیال مجازی داخل این ذره را وادار به حرکت کردن جسم صلب کند. این نیروی

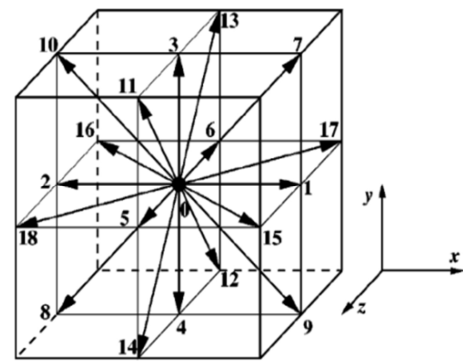


Fig. 1 LBM discrete velocity vector, D_3Q_{19} model

شکل 1 بردارهای تجزیه شده سرعت برای مدل D_3Q_{19}

تابع توزیع تعادلی $f_i^{eq}(x, t)$ در معادله 1 به صورت زیر محاسبه می شود:

$$f_i^{eq} = \rho w_i \left[1 + 3 \frac{c_i \cdot u}{c^2} + \frac{9}{2} \frac{(c_i \cdot u)^2}{c^4} - \frac{3}{2} \frac{u \cdot u}{c^2} \right] \quad (4)$$

در این رابطه، w_i ، ثابت وزنی و $c = \delta x / \delta t$ سرعت شبکه برای ذرات سیال می باشد که از یک نقطه شبکه به نقطه دیگر در حال حرکت هستند. علاوه بر این مقادیر ضرایب وزنی، w_i ، برای شبکه D_3Q_{19} به صورت زیر می باشند:

$$w_i = \begin{cases} 1/3, & i = 0 \\ 1/18, & i = 1 \sim 6 \\ 1/36, & i = 7 \sim 18 \end{cases} \quad (5)$$

در نهایت خواص ماکروسکوپی چگالی ρ و سرعت u برحسب واحدهای شبکه با استفاده از ممان های مرتبه صفر و اول از توابع توزیع حاصل می شوند و می توان روابط زیر را برای محاسبه خواص ماکروسکوپی نوشت:

$$\rho(x, t) = \sum_{i=0}^{18} f_i(x, t) \quad (6-الف)$$

$$\rho u(x, t) = \sum_{i=0}^{18} e_i f_i(x, t) \quad (6-ب)$$

$$P(x, t) = \sum_{i=0}^{18} c_s^2 \rho(x, t) \quad (6-ج)$$

که در آن $c_s = c / \sqrt{3}$ سرعت صوت در روش شبکه بولتزمن است.

3- روش نمایه هموار

روش نمایه هموار که در سال 2005 توسط ناکایاما و یاماموتو [11] معرفی شد، روشی برای شبیه سازی حرکت ذرات جامد در سیال و اعمال شرط مرزی عدم لغزش در سطح مشترک سیال و جامد می باشد. در این روش، سطح هر ذره جامد نه به عنوان یک سطح با ضخامت صفر، بلکه به عنوان یک مرز با ضخامتی قابل مقایسه با واحد شبکه معرفی می شود. به عبارتی، روش نمایه هموار هر ذره را با یک منحنی هموار موسوم به منحنی تابع موقعیت جسم جامد، $\phi(x)$ ، نشان می دهد که این منحنی در ناحیه جامد دارای مقدار یک، در ناحیه سیال دارای مقدار صفر و به طور هموار از مقدار یک به مقدار صفر در سطح مشترک جامد و سیال تغییر می کند (شکل 2).

در روش نمایه هموار، کمیت های میدانی از قبیل سرعت و تابع موقعیت ذرات جامد روی تمام ناحیه محاسباتی تعریف می شوند که این ناحیه، شامل سیال و کل ذرات جامد می باشد. برای تعیین نواحی که در آنها ذرات جامد وجود دارند، تابع موقعیت ذرات جامد به صورت زیر معرفی می شود:

گرافیکی باشد. در واقع ساختار برنامه نوشته شده در کودا می تواند ترکیبی از دستورات سری و موازی باشد به گونه ای که قسمت هایی از برنامه که قابلیت پردازش موازی را دارند بر روی کارت گرافیک اجرا شوند و قسمت هایی از برنامه که این قابلیت را ندارند روی واحد پردازنده مرکزی اجرا گردند تا از مزیت قدرت بالای پردازنده های مرکزی برای اجرا کردن برنامه های سری بهره مند شوند. در کار حاضر از نسخه 5.5 برنامه نویسی کودا در سیستم های پردازنده گرافیکی استفاده شده است.

4-1- ساختار کارت گرافیک

کارت گرافیک برای اجرا کردن توابع فراخوانی شده به چند زیر مجموعه تقسیم بندی می شود. هر یک از این زیر مجموعه ها می توانند یک تابع فراخوانی شده را انجام دهند. به تابعی که فراخوانی می شود تا بر روی کارت گرافیک اجرا گردد کرنل² می گویند. هنگامی که یک تابع (کرنل) فراخوانی می شود، تعداد زیادی از رشته های پردازشی³ توسط کارت گرافیک سازمان دهی می شوند تا تابع مورد نظر را به صورت همزمان اجرا کنند. رشته پردازشی کوچکترین واحد اجرایی می باشد که کرنل ها را بر روی کارت گرافیک اجرا می کند. این واحدهای اجرایی تقریباً شبیه رشته های پردازشی پردازنده مرکزی می باشند با این تفاوت که تعداد آنها بسیار بیشتر است. برای سازماندهی و طبقه بندی رشته های پردازشی، زبان برنامه نویسی کودا آنها را در دو سطح کلی طبقه بندی می کند. سطح اول بلوک⁴ می باشد که در این بلوک ها یک حافظه اشتراکی بسیار قوی وجود دارد که این امکان را فراهم می آورد تا رشته های پردازشی که در یک بلوک یکسان قرار دارند با هم در ارتباط باشند و بتوانند هماهنگ شوند. موقعیت رشته های پردازشی درون هر بلوک بوسیله مشخصه⁵ آن رشته پردازشی مشخص می گردد. این مشخصه می تواند به صورت یک، دو و یا سه بعدی تعریف گردد. حداکثر رشته های پردازشی درون یک بلوک به ساختار فیزیکی کارت گرافیک بستگی دارد. در کارت های گرافیکی امروزی بلوک ها می توانند حداکثر 1024 رشته پردازشی را در خود جای دهند. سطح دوم شبکه⁶ نام دارد که شامل مجموعه ای از بلوک ها می شود. بر خلاف رشته های پردازشی که درون یک بلوک قرار دارند، رشته های پردازشی که در بلوک های مختلف هستند نمی توانند با یکدیگر هماهنگ شوند. موقعیت بلوک ها همانند رشته های پردازشی نیز با شاخص بلوک⁷ در شبکه مشخص می گردد. بلوک ها نیز می توانند به صورت دو بعدی یا سه بعدی چیده شوند. همان طور که بیان شد تعداد حداکثر رشته های پردازشی درون یک بلوک به ساختار فیزیکی کارت گرافیک بستگی دارد و مقداری محدود است اما تعداد بلوک ها تقریباً محدودیتی ندارند. در کارت های گرافیکی امروزی (1 - 2³¹) بلوک می تواند در یک شبکه وجود داشته باشد. به طور کلی زمانی که یک کرنل برای اجرا شدن روی کارت گرافیک فراخوانی می شود، مجموعه ای از رشته های پردازشی درون شبکه ای از بلوک ها سازماندهی می شوند تا کرنل مورد نظر را اجرا کنند. مطابق شکل 3 وقتی که کرنل شماره یک فراخوانی می شود، یک شبکه دو بعدی از بلوک ها سازماندهی می شود تا کرنل مورد نظر را اجرا کند. درون بلوک ها نیز رشته های پردازشی به صورت سه بعدی چیده شده اند. تعداد بلوک ها و رشته های پردازشی توسط برنامه نویس در فراخوانی کرنل مشخص می گردد.

حجمی خارج از ناحیه جسم جامد صفر می باشد. نیروی اندرکنش هیدرودینامیکی جامد-سیال، f_H ، که روی گره های سیال مجازی قرار گرفته درون ذره جامد اعمال می شود به صورت زیر تعریف می شود.

$$f_H(x, t) = \phi(x, t_n) f_p(x, t_n) = \phi(x, t_n) \frac{(u_p(x, t_n) - u(x, t_n))}{\Delta t} \quad (10)$$

در این رابطه، $u_p(x, t_n)$ و $u(x, t_n)$ به ترتیب، سرعت گره هایی از شبکه با بردار موقعیت x و قرار گرفته درون ذره جامد و سرعت سیال در محل این گره ها در زمان t_n می باشند. در روش نمایه هموار، تنها یک معادله در کل میدان مورد نظر حل می شود و اثر ذرات جامد روی این میدان با یک نیروی حجمی لحاظ می شود. در روش شبکه بولتزمن روش های مختلفی برای وارد کردن نیروی حجمی به معادله تکاملی شبکه بولتزمن وجود دارد. مرسوم ترین روش، اضافه کردن یک جمله به تابع برخورد می باشد. بنابراین، ترکیب نمایه هموار با روش شبکه بولتزمن به صورت زیر در می آید:

$$f_i(x + e_i \Delta t, t_n + \Delta t) = f_i(x, t_n) - \frac{1}{\tau} [f_i(x, t_n) - f_i^{eq}(x, t_n)] + \frac{\omega_i \Delta t}{c_s^2} (f_H \cdot e_i) \quad (11)$$

با انتگرال گیری از نیروی اعمالی بر یک گره قرار گرفته درون جسم جامد روی کل حجم جسم جامد و همچنین انتگرال گیری از گشتاور حاصل از این نیروها روی کل حجم جسم جامد، می توان نیرو و گشتاور هیدرودینامیکی کل را که از طرف سیال به هر ذره جامد وارد می شوند به صورت زیر بدست آورد:

$$F_i^H = \int \rho \phi(x, t_n) f_p(x, t_n) dV_{p_i} = \int \rho \phi(x, t_n) (u(x, t_n) - u_p(x, t_n)) dV_{p_i} \quad (12)$$

$$T_i^H = \int (x - R_i^n) \rho \phi(x, t_n) f_p(x, t_n) dV_{p_i} = \int (x - R_i^n) \rho \phi(x, t_n) (u(x, t_n) - u_p(x, t_n)) dV_{p_i} \quad (13)$$

با استفاده از معادلات حرکت و روش صریح اویلر، سرعت خطی و سرعت زاویه ای ذرات جامد به صورت زیر بروز رسانی می شوند:

$$U_{c_i}^{n+1} = U_{c_i}^n + M_{p_i}^{-1} \int_{t_n}^{t_n + \Delta t} (F_i^H + F_{lubij} + F_i^{ext}) ds \quad (14-الف)$$

$$\omega_{c_i}^{n+1} = \omega_{c_i}^n + I_{p_i}^{-1} \int_{t_n}^{t_n + \Delta t} (T_i^H + T_i^{ext}) ds \quad (14-ب)$$

4- پردازش موازی

در چند دهه اخیر استفاده از پردازنده های گرافیکی برای انجام کارهای غیر گرافیکی مورد توجه محققین بسیاری قرار گرفته است. برای انجام برنامه نویسی بر روی کارت گرافیک از زبان کودا¹ استفاده می شود. کودا یک زبان برنامه نویسی برای انجام کارهای محاسباتی به وسیله کارت گرافیک می باشد. محیط برنامه نویسی کودا بر پایه دو زبان C++ و C نوشته شده است، به همین خاطر تمام ویژگی های دو زبان را دارا می باشد. ساختار برنامه کودا بر پایه همکاری میان واحد پردازنده مرکزی و واحد پردازنده گرافیکی طرح ریزی شده است به طوری که هر فایلی که بر اساس زبان کودا نوشته می شود، می تواند ترکیبی از کدهای مربوط به واحد پردازنده مرکزی و واحد پردازنده

² Kernel

³ Thread

⁴ Block

⁵ ThreadIdx

⁶ Grid

⁷ BlockIdx

¹CUDA

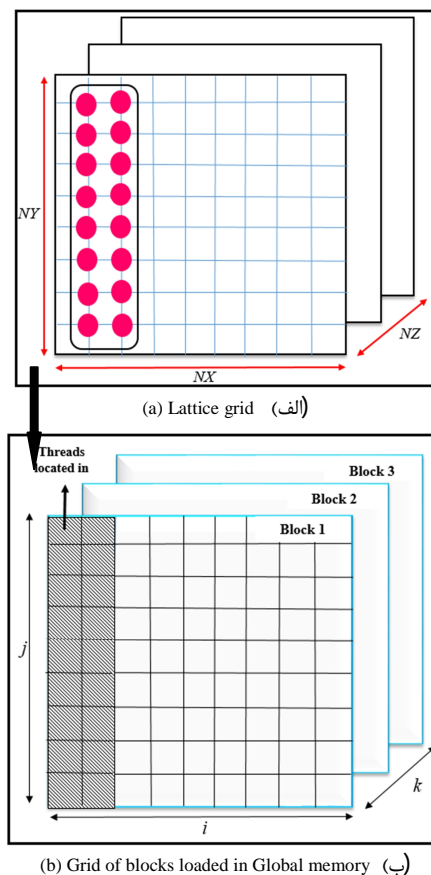


Fig. 4 Schematic view of (a) lattice grid and (b) thread block which is linked to a lattice grid

شکل 4 شماتیکی از الف) شبکه محاسباتی در شبکه بولتزمن و ب) چینش رشته های پردازشی در بلوکها متناسب با شبکه اصلی

صورت سه بعدی از ساختار Dim3 استفاده می گردد. این ساختار به طور کلی به صورت زیر تعریف می شود:

(بعد در راستای z ، بعد در راستای y ، بعد در راستای x) نام متغیر Dim3 به عنوان مثال در شکل 3 برای ایجاد یک بلوک که رشته های پردازشی در آن به صورت سه بعدی پیچیده شده اند از دستور زیر استفاده می گردد:

Dim3 Block (4, 2, 2)

در این حالت یک بلوک سه بعدی با نام مورد نظر با ابعاد 4، 2 و 2 در راستاهای x ، y و z ساخته می گردد که در شکل 3 قابل مشاهده است. برای چینش بلوکها و رشته های پردازشی به صورت دو و یا سه بعدی لازم است در ابتدا با استفاده از ساختار Dim3 ساختار مورد نظر ساخته شود و نام ساختار مورد نظر به جای تعداد بلوکها و رشته های پردازشی قرار گیرد.

4-3- اختصاص دهی حافظه

واحد پردازنده مرکزی و کارت گرافیک حافظه های جداگانه ای دارند. به دلیل اینکه کارت گرافیک به حافظه اصلی سیستم دسترسی ندارد، لذا برای اجرا کردن یک کرنل بر روی کارت گرافیک لازم است که برنامه نویس بر روی حافظه کارت گرافیک مقدار مناسبی حافظه برای انجام محاسبات تخصیص دهد¹ کند. سپس اطلاعات مربوطه را از حافظه اصلی سیستم به حافظه تخصیص یافته بر روی کارت گرافیک منتقل نماید و در نهایت پس از انجام

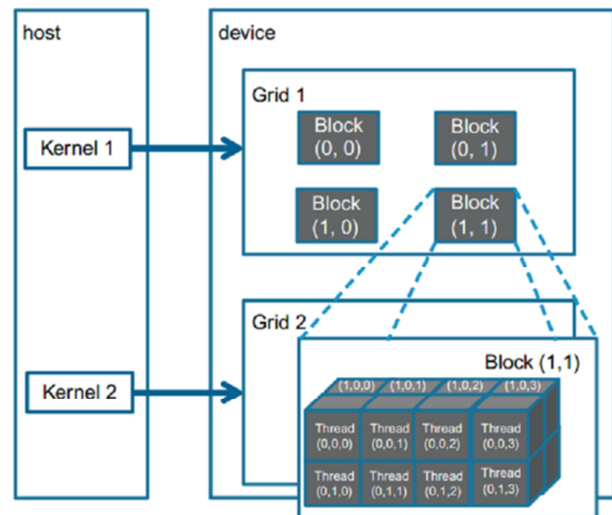


Fig. 3 Schematic view of threads arrangement in GPU

شکل 3 شماتیکی از چینش رشته های پردازشی در کارت گرافیک

در کارت گرافیک برای فراخوانی کرنلها از دستور زیر استفاده می شود: (ارگومان های تابع) <<< تعداد رشته پردازشی، تعداد بلوکها >>> نام کرنل همان گونه که بیان گردید، تعداد رشته های پردازشی و بلوکها توسط برنامه نویس و متناسب با هندسه مساله و الگوریتم حل انتخاب می شوند. تعداد آنها باید به گونه ای انتخاب شود که بالاترین بازدهی ممکن را داشته باشد. به عنوان مثال برای جمع کردن دو ماتریس که هر کدام 10 درایه دارند، برنامه نویس می تواند از راهکارهای زیر برای موازی سازی استفاده کند. اولین راهکار این است که یک بلوک که در آن یک رشته پردازشی قرار دارد برای انجام محاسبات فراخوانی شود. در این حالت جمع کردن تمامی درایه ها توسط یک رشته پردازشی انجام می شود. در حالت دوم برنامه نویس می تواند جمع کردن هر دو درایه را به یک رشته پردازشی اختصاص دهد. در این حالت به پنج بلوک که در آن یک رشته پردازشی قرار دارد نیاز می باشد. در حالت سوم برنامه نویس برای جمع کردن هر درایه از یک رشته پردازشی استفاده می کند. در این حالت عملیات جمع کردن هر 10 درایه همزمان با هم در یک لحظه انجام می شود و بالاترین بازدهی ممکن را خواهد داشت. به طور کلی تعداد بلوکها و رشته های پردازشی باید متناسب با الگوریتم موازی سازی مساله انتخاب شوند تا بالاترین بازدهی بدست آید. در روش حاضر نیز هر گره محاسباتی روش بولتزمن توسط یک رشته پردازشی انجام می شود. این عمل سبب می گردد که محاسبات مربوطه همزمان با هم انجام شوند و دیگر نیازی به حلقه های تکرار پی در پی نباشد.

4-2- شبکه بندی رشته های پردازشی

یک از مهمترین موضوعات در استفاده از کارت گرافیک چینش رشته های پردازشی در یک شبکه می باشد به گونه ای که با شبکه محاسباتی مورد نظر همخوانی داشته باشد. در کار حاضر رشته های پردازشی همانند شبکه محاسباتی به صورت سه بعدی پیچیده شده اند و هر گره محاسباتی در روش شبکه بولتزمن توسط یک رشته پردازشی انجام می شود. شکل 4 نحوه چینش بلوکها و رشته های پردازشی درون آنها را مطابق با هندسه اصلی نمایش می دهد.

در این چینش محاسبات هر گره در روش شبکه بولتزمن توسط یک رشته پردازشی انجام می گیرد. برای چینش رشته های پردازشی و بلوکها به

¹ Allocate

یک نکته مهم مرتبط با الگوریتم 1، محاسبه نیرو و گشتاور وارد بر ذره (معادلات 12 و 13) در پردازنده مرکزی به جای کارت گرافیک می باشد. همانطور که از معادلات مربوطه نیز مشخص می باشد نیرو و گشتاور وارد بر هر ذره از جمع بستن تمام نیروهای حجمی که در درون ذره قرار دارند محاسبه می شود. این نیروهای حجمی که از معادله 10 بدست می آیند، به نقاطی از سیال وارد می شوند که ذره در آنجا وجود داشته باشد. جمع کردن این نیروها در کارت گرافیک یک چالش مهم در این مساله می باشد. به دلیل اینکه عمل جمع کردن ذاتا یک عمل سری است و نیازمند چند حلقه تو در تو می باشد، لذا نمی توان به راحتی آن را در کارت گرافیک انجام داد چون جمع کردن این حلقه های تو در تو برای رشته های پردازشی سنگین بوده و راندمان کار را کاهش می دهد. از طرف دیگر همان طور که بیان شد محاسبات مربوط به هر گره محاسباتی در کارت گرافیک توسط یک رشته پردازشی انجام می گیرد و چون فقط رشته های پردازشی درون یک بلوک می توانند اطلاعات خود را به اشتراک بگذارند، لذا رشته های پردازشی در یک بلوک نمی توانند به اطلاعات رشته های پردازشی دیگر بلوک ها دسترسی داشته باشند. لذا جمع کردن اطلاعات آنها مساله را دچار مشکل می کند. به علاوه اینکه رشته های پردازشی همزمان با هم کار انجام دادن محاسبات را انجام می دهند ولی هیچ تضمینی وجود ندارد که همه آنها در یک زمان کار خود را تمام کنند. از این رو ممکن است هنگام جمع کردن نتایج، یک رشته پردازشی کارش تمام نشده باشد و مقدار عددی نیروی مورد نظر در مرحله جدید تغییر نکرده باشد و این موضوع سبب بوجود آمدن مشکلاتی در حل شود. از این رو برای حل این مشکل، نتایج حاصل از محاسبه نیروی حجمی در هر مرحله به پردازنده مرکزی فرستاده می شود تا عمل جمع کردن توسط پردازنده مرکزی صورت می گیرد.

5- نتایج

5-1- جریان درون یک کانال

در مرحله اول، به منظور اعتبار سنجی روش شبکه بولتزمن برای شبیه سازی جریان سیال و همچنین بررسی تاثیر کارت گرافیک بر کاهش زمان محاسبات، جریان سیال درون یک کانال مورد بررسی قرار گرفت. هندسه مورد بررسی جریان سیال درون یک کانال مکعبی می باشد. در این هندسه طول، عرض و ارتفاع 96 بر حسب ابعاد شبکه بولتزمن انتخاب شده است. شرایط مرزی در ورود و خروج، تناوبی و در چهار وجه دیگر شرط مرزی کمانه کردن اعمال شده است. برای بررسی صحت نتایج، نتایج حل عددی با حل تحلیلی مورد مقایسه قرار گرفت که در شکل 5 قابل مشاهده می باشد. به منظور بررسی تاثیر کارت گرافیک بر روی کاهش زمان محاسبات، مساله مورد نظر نیز برای شبکه $64 \times 64 \times 64$ به صورت کامل توسط پردازنده مرکزی و پردازنده گرافیکی برای 80000 تکرار حل شد و همان طور که از جدول 1 مشخص می باشد زمان پردازش با کارت گرافیک حدود 80 برابر نسبت به پردازنده مرکزی کاهش یافته است. مشخصات سیستم مورد استفاده در جدول 2 بیان شده است.

5-2- نیروی پسای وارد بر یک کره در جریان سه بعدی

در این مطالعه جریان سیال حول یک کره به منظور بدست آوردن نیروی وارد بر آن مورد بررسی قرار می گیرد. هندسه مسئله به گونه ای می باشد که یک کره ثابت در وسط میدان سیال مکعب شکلی که دارای شرایط مرزی پیرویک در هر شش وجه است، قرار داده شده و برای ایجاد جریان سیال،

محاسبات، نتایج را به حافظه اصلی انتقال دهد و حافظه اختصاص یافته در کارت گرافیک را آزاد² کند. حافظه عمومی کارت گرافیک برای واحد پردازنده مرکزی قابل دسترسی است تا بتواند اطلاعات را به کارت گرافیک بفرستد یا از آن بگیرد. برای تخصیص دادن حافظه کارت گرافیک یا انتقال اطلاعات بین حافظه واحد پردازنده مرکزی و کارت گرافیک نیاز به توابع API³ می باشد. توضیحات اضافی در مورد این توابع در مرجع [5] بیان شده است.

4-4- الگوریتم موازی سازی

الگوریتم محاسباتی برای موازی سازی ترکیب دو روش شبکه بولتزمن و نمایه هموار در الگوریتم 1 بیان شده است. در این الگوریتم، روند اجرای برنامه روی کارت گرافیک تشریح شده است. برای آنکه موضوع دقیق تر بررسی گردد یک شبه کد در پیوست ضمیمه شده است.

برای آنکه مشخص شود که یک تابع کرنل است و باید به کارت گرافیک فرستاده شود، از شناساگر³ قبل از نام تابع استفاده می شود. درواقع با آوردن یک شناسه قبل از نام تابع مشخص می شود آن تابع از چه نوعی است و باید بر روی کدام پردازنده پردازش شود. این شناسه ها عبارت اند از:

__global__: تابعی با این شناسه به وسیله واحد پردازنده مرکزی فراخوانی شده و به وسیله کارت گرافیک اجرا می شود. کرنل ها به وسیله این شناسه مشخص می شوند.

__device__: این نوع توابع به وسیله کارت گرافیک فراخوانی و اجرا می شوند و درواقع توابع کمکی کرنل ها هستند.

__host__: این نوع توابع به وسیله واحد پردازنده مرکزی فراخوانی و اجرا می شوند و می توان آن ها را بدون شناسه نیز آورد.

الگوریتم 1 الگوریتم پیاده سازی روش شبکه بولتزمن و نمایه هموار روی کارت گرافیک

Algorithm 1 Implementation of LBM and SPM on GPU algorithm

تخصیص دهی حافظه روس کارت گرافیک و پردازنده مرکزی
مقدار دهی اولیه برای خواص ماکروسکوپی، تابع موقعیت ذرات جامد \emptyset ، و موقعیت اولیه ذرات جامد در پردازنده مرکزی
انتقال خواص ماکروسکوپی، تابع موقعیت ذرات جامد \emptyset و موقعیت اولیه ذرات جامد از حافظه پردازنده مرکزی به حافظه کارت گرافیک
مقدار دهی اولیه برای تابع توزیع احتمال، $f_i(x, t)$ ، برای روی حافظه کارت گرافیک

محاسبه تابع موقعیت ذرات جامد و نیروی حجمی بر روی کارت گرافیک طبق معادلات (10,8)

اجرای کرنل روش شبکه بولتزمن به منظور حل کردن جریان سیال طبق معادله (11)

اعمال شرایط مرزی در روش شبکه بولتزمن

محاسبه خواص ماکروسکوپی در کارت گرافیک طبق معادله (6)

انتقال تابع موقعیت ذرات جامد، خواص ماکروسکوپی و نیروی حجمی از حافظه کارت گرافیک به حافظه پردازنده مرکزی

محاسبه نیرو و گشتاور هیدرودینامیکی وارد بر ذره مورد نظر در پردازنده مرکزی طبق معادلات (12,13)

به روز رسانی موقعیت ذرات جامد طبق معادله (14)

² Free

³ Application Programming Interface

⁴ Quantifier

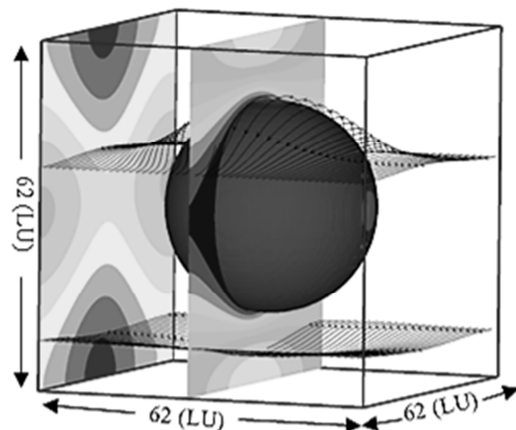


Fig. 6 Three dimensional view of contour and stream line around fixed sphere in steady flow (Lattice Unit)

شکل 6 نمای سه بعدی از خطوط جریان و کانتور سرعت اطراف یک کره ثابت در جریان پایا (بر حسب ابعاد شبکه بولتزمن)

5-3-ته نشینی یک ذره کروی درون یک محفظه مستطیلی

در سومین مطالعه، شبیه سازی عددی ته نشینی یک ذره در یک محفظه مکعب مستطیلی حاوی یک سیال لزج نیوتنی غیر قابل تراکم مورد بررسی قرار می گیرد. طول، عرض و ارتفاع این محفظه مطابق شکل 8 به ترتیب $L_x = 1 \text{ cm}$ ، $L_y = 1.6 \text{ cm}$ و $L_z = 1 \text{ cm}$ می باشند. یک ذره کروی با قطر $d_p = 0.15 \text{ cm}$ و چگالی $\rho_p = 1.12 \text{ g/cm}^3$ در یک سیال با چگالی $\rho_f = 1.12 \text{ g/cm}^3$ و چگالی $\rho_p = 1.12 \text{ g/cm}^3$ در یک سیال با چگالی $\rho_f = 1.12 \text{ g/cm}^3$ قرار می گیرد. مبدأ مختصات در گوشه پایین سمت چپ قرار دارد و موقعیت اولیه این ذره در $(0.5 \text{ cm}, 0.5 \text{ cm}, 1.2 \text{ cm})$ می باشد. شرایط مرزی برای شش وجه محفظه کمانه کردن استاندارد قرار داده شده است و سیال و ذره در ابتدا در حال سکون می باشند، سپس ذره تحت اثر نیروی جاذبه شروع به سقوط می کند. ناحیه محاسباتی دارای ابعاد $(131 \times 131 \times 209)$ در روش شبکه بولتزمن می باشد. قطر ذره بر حسب واحد شبکه 19.5 و زمان آرامش $\tau = 0.9$ انتخاب شده است.

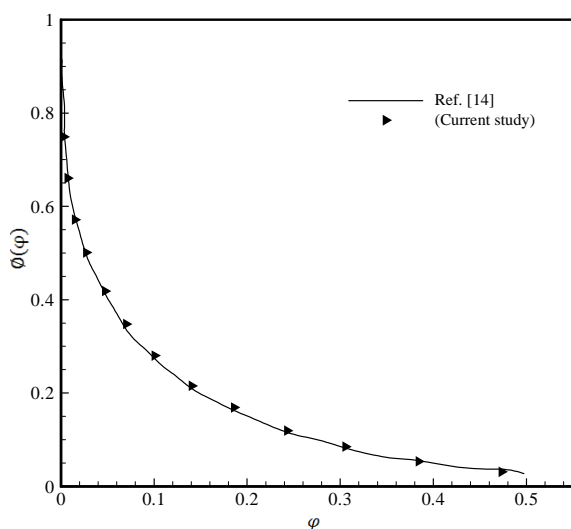


Fig. 7 Drag coefficient of fixed sphere in steady flow as a function of volume fraction

شکل 7 تغییرات ضریب پسا یک کره ثابت در جریان پایا بر حسب نسبت حجمی

جدول 1 مقایسه مدت زمان حل برای شبکه $64 \times 64 \times 64$ برای پردازنده مرکزی

و پردازنده گرافیکی

Table 1 Comparison of simulation time for $64 \times 64 \times 64$ grids between GPU and CPU

پردازنده های مختلف	زمان بر حسب دقیقه
پردازنده مرکزی سیستم 1	238
پردازنده گرافیکی سیستم 1	3

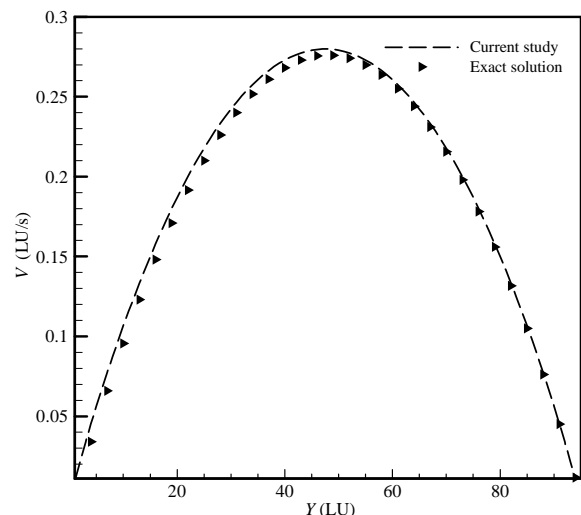


Fig.5 Velocity profile at the middle plane (Lattice Unit)

شکل 5 پروفیل سرعت در صفحه میانی کانال بر حسب ابعاد شبکه بولتزمن

جدول 2 مشخصات کارت های گرافیک و واحدهای پردازنده مرکزی مختلف

Table 2 Configuration of different GPU and CPU

نام پردازنده	مدل پردازنده مرکزی	مدل پردازنده گرافیکی	قابلیت محاسباتی	تعداد چند پردازنده ها	حافظه سیستم
سیستم 1	Intel® core i7-4770k	GTX Titan	3.5	14	16
سیستم 2	Intel® core i7-4790	GTX 980	5.2	16	32

گرادیان فشار ثابت به صورت یک نیروی حجمی به سیال وارد می شود (شکل 6). اندازه میدان سیال $L=62\Delta x$ است. شعاع کره از $R=8\Delta x$ تا $R=31\Delta x$ تغییر می کند که Δx فاصله گره ها می باشد. همچنین برای ایجاد کره جامد از روش نمایه هموار استفاده شده است.

برای اعداد رینولدز پایین، نیروی پسای وارد بر یک کره از رابطه استوکس به دست می آید:

$$F_D = 6 \pi R \mu U \quad (15)$$

که از آن برای بی بعد کردن نیروی وارد بر کره استفاده شده است [14]. در این رابطه F_D نیروی پسا، R شعاع کره، μ لزجت و U سرعت متوسط حجمی سیال می باشد. شکل 7 ضریب پسا $\phi(\varphi)$ را بر حسب نسبت حجمی φ نشان می دهد. مقادیر ضریب پسا و نسبت حجمی به ترتیب بر حسب روابط زیر بدست می آید:

$$\phi(\varphi) = -\frac{6 \pi R \mu U}{F_D} \quad (16)$$

$$\varphi = \left(\frac{4}{3}\right) \pi \left(\frac{R}{L}\right)^3 \quad (17)$$

همان گونه که از شکل 7 مشخص می باشد، نتایج کار حاضر سازگاری خوبی را با نتایج ارائه شده توسط زیگ و هموسی [14] دارد.

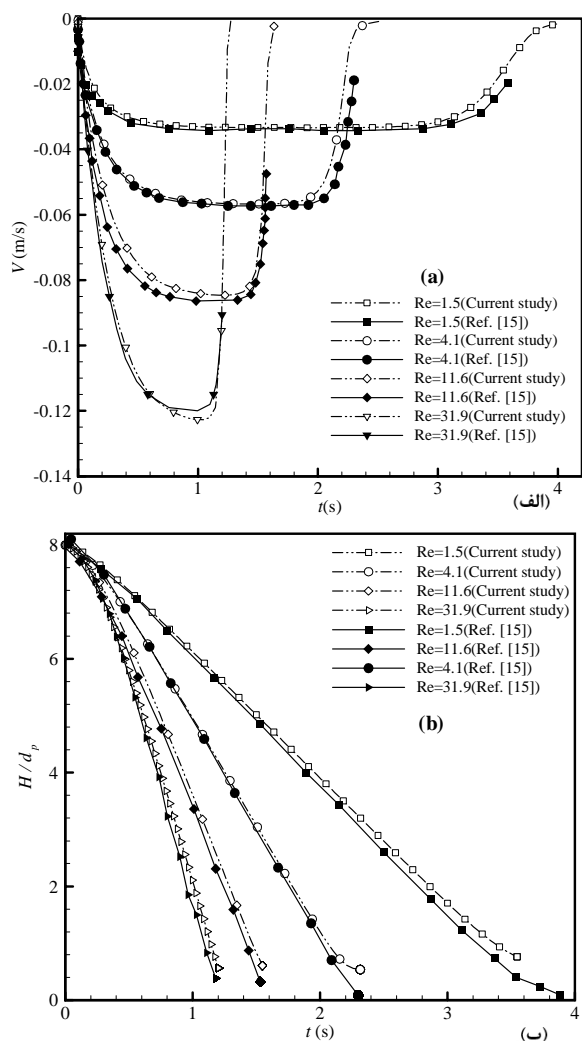


Fig. 9 Variation of vertical (a) velocity and (b) height of sphere with time for different Reynolds numbers

شکل 9 تغییرات عمودی (الف) سرعت (ب) ارتفاع یک کره همراه با زمان برای اعداد رینولدز مختلف

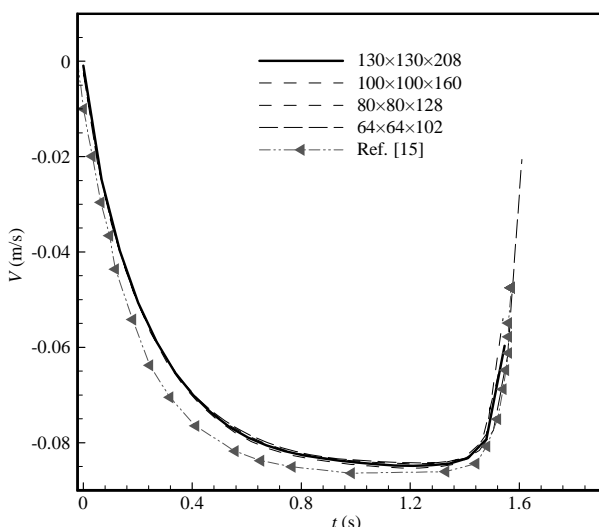


Fig. 10 Grid study for simulation of settling sphere in a quiescent fluid.

شکل 10 مطالعه عدم وابستگی شبیه سازی ته نشینی یک ذره کروی نسبت به شبکه محاسباتی

شکل 9 به ترتیب سرعت سقوط ذره کروی و موقعیت آن ذره را در زمان‌های مختلف نمایش می‌دهند. همان‌گونه که از شکل‌ها مشخص است ذره جامد در ابتدا در اثر نیروی وزن شتاب گرفته و سرعت آن افزایش می‌یابد. در ادامه هر چه قدر که سرعت آن زیاد می‌شود، نیروی درگ اصطکاکی وارد بر ذره نیز افزایش یافته تا زمانی که به اندازه نیروی وزن برسد. در این حالت برآیند نیروهای وارد بر ذره صفر شده و ذره با سرعت ثابت به حرکت خود ادامه می‌دهد و در نهایت در ته محفظه ته نشین می‌شود. سازگاری خوبی بین نتایج این مطالعه و نتایج ارایه شده توسط تین کیت [15] وجود دارد، به جز در نواحی نزدیک به دیواره که این ناشی از مدل‌های مختلف برخورد بین ذره جامد و دیواره پایین می‌باشد که در این مطالعه استفاده شده است.

در کار حاضر به منظور نشان دادن عدم وابستگی حل عددی حاضر به شبکه محاسباتی، شبیه سازی سقوط ذره در عدد رینولدز $Re = 11.6$ برای چندین شبکه مطابق شکل 10 بررسی شده است. همان‌گونه که از شکل مشخص می‌باشد حل حاضر به شبکه محاسباتی وابسته نیست.

5-4- بررسی کارایی کارت گرافیک در محاسبات

در نهایت به منظور بررسی راندمان¹ الگوریتم ارائه شده بر روی کارت گرافیک، شبیه سازی ته نشینی یک ذره در اثر نیروی وزن با دقت مرتبه دوم² بر روی سیستم شماره 2 برای شبکه‌های محاسباتی مختلف مورد مطالعه قرار گرفت. برای انجام این بررسی از روش تعداد گره‌های محاسباتی به روز رسانی شده بر واحد زمان³ استفاده شده است که معمول‌ترین روش در شبکه بولتزمن می‌باشد [10]. شکل 11 راندمان بدست آمده بر حسب تعداد گره‌های محاسباتی به روز رسانی شده را بر حسب ثانیه برای شبکه‌های مختلف نمایش می‌دهد. همان‌گونه که از شکل نیز مشخص می‌باشد، در کار حاضر می‌توان به توان محاسباتی 6.5 میلیون گره پردازش شده در واحد زمان دست یافت. همان‌طور که ملاحظه می‌گردد، راندمان کارت گرافیک با افزایش گره‌های محاسباتی افزایش می‌یابد که مبین آن است که استفاده از کارت گرافیک برای شبکه‌های بزرگتر مناسب‌تر است.

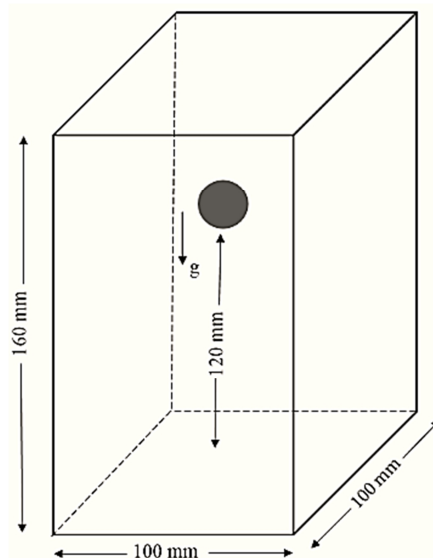


Fig. 8 Schematic view of a sphere in a square cylinder.

شکل 8 شماتیکی از یک ذره کروی درون یک محفظه مستطیلی

- 1- Performance
- 2- Double precision
- 3- Million fluid lattice node updates per second

جدول 3 اعداد رینولدز، چگالی و ویسکوزیته سیال به همراه سرعت ذره در فرایند سقوط ذره بیان شده در مرجع [15]

تعریف شده‌اند آورده شده است. در ادامه برای همین این ماتریس‌ها با نام $fin0_d$, $fin1_d$ و غیره در حافظه کارت گرافیک، حافظه اختصاص دهی می‌شود. سپس اطلاعات به کمک توابع API به کارت گرافیک فرستاده می‌شوند تا توابع مورد نظر روی کارت گرافیک اجرا گردند. در نهایت نتایج به حافظه پردازنده مرکزی انتقال داده شده و حافظه‌های اختصاص داده شده آزاد می‌شوند.

Pseudo-code for GPU program

```
int main( )
{
    Define parameters for CPU
    double *fin0_h,*fin1_h, u_h, XC_h, ...;
    int num; /* Element number for matrix */
    Define parameters for GPU
    double *fin0_d,*fin1_d, u_d, XC_d, ...;
    Allocate memory for CPU
    double *fin0_h=(float *)malloc((size od double*num));
    double *fin1_h=(float *)malloc((size od double*num));
    Allocate memory for GPU
    cudaMalloc((void **) &fin0_d,(size od double*num) );
    cudaMalloc((void **) &fin1_d,(size od double*num) );
    Initialize macroscopic properties and position of particles in CPU
    U_h=0;
    XC_h=10;
    Copy macroscopic properties and position of particles form CPU to GPU
    cudaMemcpy(u_d,u_h, size od double*num, cudaMemcpyHostToDevice);
    cudaMemcpy(XC_d,XC_h, size od double*num, cudaMemcpyHostToDevice);
    Initialize fin0_d and fin1_d in GPU
    While (...)
    {
        Call functions
        SPM kernel <<<Number of blocks, Number of theads>>>
        (fin0_d, fin1_d, ...);
        LBM kernel <<<Number of blocks, Number of theads>>>
        (fin0_d, fin1_d, ...);
        LBM boundry conditions <<<Number of blocks, Number of theads>>>
        (fin0_d, fin1_d, ...);
        Macroscopic properties kernel <<<Number of blocks, Number of theads>>>
        (fin0_d, fin1_d, ...);
        Copy macroscopic properties and body force form GPU to CPU
        cudaMemcpy(u_h,u_d, size od double*num, cudaMemcpyDeviceToHost);
    }
    Free fin0_h, fin1_h;
    cudaFree(d_a);
    cudaFree(d_b);
    return 0;
}
```

8- مراجع

- [1] C. K. Aidun, J. R. Clausen, Lattice-Boltzmann method for complex flows, *Annual review of fluid mechanics*, Vol. 42, pp. 439-472, 2010.
- [2] W. Li, X. Wei, A. Kaufman, Implementing lattice Boltzmann computation on graphics hardware, *The Visual Computer*, Vol. 19, No. 7-8, pp. 444-456, 2003.

Table 3 Reynolds number, fluid density and viscosity and terminal velocity of sphere in its sedimentation. (Re, ρF and μF were taken from Ref. [15])

عدد رینولدز	ویسکوزیته سیال ($\frac{10^{-3} NS}{m^2}$)	چگالی سیال ($\frac{kg}{m^3}$)	سرعت ذره ($\frac{m}{s}$)
1.5	373	970	0.038
4.1	212	965	0.06
11.6	113	962	0.09
31.9	58	960	0.128

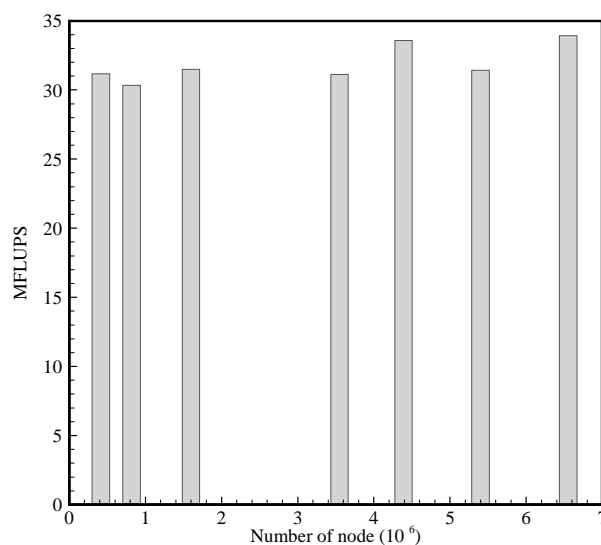


Fig. 11 Performance of GPU based on number of lattice nodes

شکل 11 راندمان کارت گرافیک بر حسب تعداد گره‌های محاسباتی

6- نتیجه گیری

در تحقیق حاضر، شبیه سازی جریان با استفاده از ترکیب دو روش شبکه بولتزمن و نمایه هموار از طریق پردازش موازی روی کارت گرافیک برای اولین بار مورد بررسی قرار گرفت. در ابتدا برای بررسی صحت کد نوشته شده و بررسی تأثیر پردازش موازی بر زمان حل، جریان آرام در یک کانال مربعی مورد بررسی قرار گرفت که نتایج به دست آمده از حل عددی با نتایج به دست آمده از حل تحلیلی کاملاً تطبیق داشت. با مقایسه بین مدت زمان حل به کمک پردازنده گرافیکی و پردازنده مرکزی مشخص شد که برای جریان درون یک کانال، زمان حل بوسیله پردازنده گرافیکی به $1/80$ زمان حل با پردازنده مرکزی کاهش می‌یابد. در ادامه سقوط یک ذره بر اثر نیروی وزن برای بررسی تأثیر کارت گرافیک بر روی موازی سازی الگوریتم این دو روش بررسی شد. در نهایت توان محاسباتی کارت گرافیک بر حسب تعداد گره‌های پردازش شده در واحد زمان مورد بررسی قرار گرفت. نتایج بدست آمده موید این بود که با استفاده از الگوریتم حاضر می‌توان به توان محاسباتی 6.5 میلیون گره محاسباتی پردازش شده در واحد زمان دست یافت.

7- پیوست

در این قسمت یک شبه کد مربوط شبیه سازی عددی ته نشینی یک ذره در یک محفظه مکعب مستطیلی روی کارت گرافیک به منظور فراگیری قسمت‌های مختلف کارت گرافیک آورده شده است. در این برنامه ماتریس‌های

- Journal of Fluid Mechanics*, Vol. 271, No. 1, pp. 285-309, 1994.
- [9] A. J. Ladd, Numerical simulations of particulate suspensions via a discretized Boltzmann equation. Part 2. Numerical results, *Journal of Fluid Mechanics*, Vol. 271, pp. 311-339, 1994.
- [10] C. Obrecht, F. Kuznik, B. Tourancheau, J.-J. Roux, Efficient GPU implementation of the linearly interpolated bounce-back boundary condition, *Computers and Mathematics with Applications*, Vol. 65, No. 6, pp. 936-944, 2013.
- [11] Y. Nakayama, R. Yamamoto, Simulation method to resolve hydrodynamic interactions in colloidal dispersions, *Physical Review E*, Vol. 71, No. 3, pp. 036707, 2005.
- [12] S. Jafari, R. Yamamoto, M. Rahnama, Lattice-Boltzmann method combined with smoothed-profile method for particulate suspensions, *Physical Review E*, Vol. 83, No. 2, pp. 026702_1-7, 2011.
- [13] P. Bhattachagor, E. Gross, M. Krook, A model for collision processes in gases, *Physical Review*, Vol. 94, No.3, pp. 511, 1954.
- [14] A. Zick, G. Homsy, Stokes flow through periodic arrays of spheres, *Journal of fluid mechanics*, Vol. 115, pp. 13-26, 1982.
- [15] A. Ten Cate, C. Nieuwstad, J. Derksen, H. Van den Akker, Particle imaging velocimetry experiments and lattice-Boltzmann simulations on a single sphere settling under gravity, *Physics of Fluids* (1994-present), Vol. 14, No. 11, pp. 4012-4025, 200.
- [3] F. Kuznik, C. Obrecht, G. Rusaouen, J.-J. Roux, LBM based flow simulation using GPU computing processor, *Computers & Mathematics with Applications*, Vol. 59, No. 7, pp. 2380-2392, 2010.
- [4] E. Riegel, T. Indinger, N. Adams, Implementation of a Lattice-Boltzmann method for numerical fluid mechanics using the nVIDIA CUDA technology, *Computer Science-Research and Development*, Vol. 23, No. 3-4, pp. 241-247, 2009.
- [5] J. Tölke, Implementation of a Lattice Boltzmann kernel using the compute unified device architecture developed by nVIDIA, *Computing and Visualization in Science*, Vol. 13, No. 1, pp. 29-39, 2010.
- [6] M. H. Sedaghati, M. M. Shahmardan, M. Nazari, M. Norouzi, Immersed boundary- Lattice Boltzmann method for modeling non-Newtonian fluid flow around curved boundaries, *Modares Mechanical Engineering*, Vol. 14, No. 8, pp. 146-156, 2013. (in Persian فارسی)
- [7] O. R. Mohammadipoor, H. Niazmand, S. A. Mirbozorgi, A new curved boundary treatment for the Lattice Boltzmann method, *Modares Mechanical Engineering*, Vol. 13, No. 8, pp. 21-48, 2013. (in Persian فارسی)
- [8] A. J. Ladd, Numerical simulations of particulate suspensions via a discretized Boltzmann equation. Part 1. Theoretical foundation,