





دانشکده : مهندسی مکانیک

گروه : تبدیل انرژی

پایان نامه کارشناسی ارشد

شبیه سازی جریان عبوری از استوانه با حل دستگاه

معادلات سه قطری روی GPU

میلاذ سوری

اساتید راهنما

دکتر پوریا اکبرزاده

دکتر محسن نظری

استاد مشاور

دکتر حسین محمودی داریان

شهریور ۱۳۹۴



دانشگاه گیلان
مدیریت تحصیلات تکمیلی
فرم شماره (۶)

باسمه تعالی

شماره:
تاریخ:
ویرایش:

فرم صورت جلسه دفاع از پایان نامه تحصیلی دوره کارشناسی ارشد

با تأییدات خداوند متعال و با استعانت از حضرت ولی عصر (عج) نتیجه ارزیابی جلسه دفاع از پایان نامه کارشناسی ارشد آقای میلاد سوری به شماره دانشجویی ۹۱۰۳۹۷۴ رشته مهندسی مکانیک گرایش تبدیل انرژی تحت عنوان شبیه سازی جریان عبوری از استوانه با حل دستگاه معادلات سه قطری روی GPU که در تاریخ ۱۳۹۴/۰۶/۳۰ با حضور هیأت محترم داوران در دانشگاه شاهرود برگزار گردید به شرح ذیل اعلام می گردد:

قبول (با درجه : بیاض - امتیاز ۱۸۸۳) ☐ دفاع مجدد ☐ مردود ☐

۲- بسیار خوب (۱۸ - ۱۸/۹۹)

۱- عالی (۱۹ - ۲۰)

۴- قابل قبول (۱۴ - ۱۵/۹۹)

۳- خوب (۱۶ - ۱۷/۹۹)

۵- نمره کمتر از ۱۴ غیر قابل قبول

عضو هیأت داوران	نام و نام خانوادگی	مرتبه علمی	امضاء
۱- استاد راهنمای اول	پوریا اکبرزاده	استادیار	
۲- استاد راهنمای دوم	محسن نظری	دانشیار	
۳- استاد مشاور	حسین محمودی داریان	استادیار	
۴- نماینده شورای تحصیلات تکمیلی	احمد مددی	مربی	
۵- استاد ممتحن	محمد حسن کیهانی	استاد	
۶- استاد ممتحن	علی جباری مقدم	دانشیار	

امضاء

دانشکده :

رئیس



تقدیم به:

پدرو مادر مهربان و بزرگوارم

تقدیر و تشکر

از خداوند متعال سپاسگزارم که توفیق کسب علم و دانش را به من عطا فرمود تا بتوانم این مرحله از علم‌آموزی را با موفقیت به پایان برسانم. از خانواده عزیزم به خاطر حمایت‌ها و محبت‌های بی‌دریغی که نسبت به من داشته و دارند، کمال تشکر و سپاس را دارم. از استادان ارجمندم، جناب آقای دکتر پوریا اکبرزاده و جناب آقای دکتر محسن نظری به خاطر راهنمایی‌های ارزشمند و زحمات ایشان در کلیه مراحل انجام پایان‌نامه تقدیر و تشکر می‌نمایم. همچنین از زحمات و راهنمایی‌های ارزشمند جناب آقای دکتر محمودی بسیار ممنون و سپاسگزارم که بدون ایشان انجام این مطالعه میسر نبود.

در انتها از تمامی اساتید محترم دانشکده مکانیک دانشگاه صنعتی شاهرود که توفیق شاگردیشان را داشتم، سپاس‌گزاری نموده و از خداوند منان آرزوی سلامت و توفیق روزافزون برایشان دارم.

میلاذ سوری

شهریور ۱۳۹۴

دانشجو تأیید می‌نماید که مطالب مندرج در این پایان نامه نتیجه تحقیقات خودش می‌باشد و در صورت استفاده از نتایج دیگران مرجع آن را ذکر نموده است.

کلیه حقوق مادی مترتب از نتایج مطالعات، آزمایشات و نوآوری ناشی از تحقیق موضوع این پایان نامه متعلق به دانشگاه شاهرود می‌باشد .

شهریور ۱۳۹۴

تعهد نامه

اینجانب **میلاد سوری** دانشجوی دوره کارشناسی ارشد رشته تبدیل انرژی دانشکده مکانیک دانشگاه صنعتی شاهرود نویسنده پایان نامه شبیه سازی جریان عبوری از استوانه با حل دستگاه معادلات سه قطری روی **GPU** تحت راهنمایی دکتر پوریا اکبرزاده و دکتر محسن نظری متعهد می شوم :

- تحقیقات در این پایان نامه توسط اینجانب انجام شده است و از صحت و اصالت برخوردار است .
- در استفاده از نتایج پژوهشهای محققان دیگر به مرجع مورد استفاده استناد شده است .
- مطالب مندرج در پایان نامه تاکنون توسط خود یا فرد دیگری برای دریافت هیچ نوع مدرک یا امتیازی در هیچ جا ارائه نشده است .
- کلیه حقوق معنوی این اثر متعلق به دانشگاه شاهرود می باشد و مقالات مستخرج با نام « دانشگاه شاهرود » و یا « Shahrood University » به چاپ خواهد رسید.
- حقوق معنوی تمام افرادی که در به دست آمدن نتایج اصلی پایان نامه تأثیرگذار بوده اند در مقالات مستخرج از پایان نامه رعایت می گردد.
- در کلیه مراحل انجام این پایان نامه، در مواردی که از موجود زنده (یا بافتهای آنها) استفاده شده است ضوابط و اصول اخلاقی رعایت شده است .
- در کلیه مراحل انجام این پایان نامه، در مواردی که به حوزه اطلاعات شخصی افراد دسترسی یافته یا استفاده شده است اصل رازداری، ضوابط و اصول اخلاق انسانی رعایت شده است.

تاریخ

امضای دانشجو

مالکیت نتایج و حق نشر

- کلیه حقوق معنوی این اثر و محصولات آن (مقالات مستخرج، کتاب، برنامه های رایانه ای، نرم افزار ها و تجهیزات ساخته شده است) متعلق به دانشگاه شاهرود می باشد. این مطلب باید به نحو مقتضی در تولیدات علمی مربوطه ذکر شود .

* متن این صفحه نیز باید در ابتدای نسخه های تکثیر شده پایان نامه وجود داشته باشد .

چکیده

در سال‌های اخیر استفاده از پردازنده‌های گرافیکی به عنوان یک پردازشگر موازی و شتاب‌دهنده به محاسبات علمی رواج بسیاری یافته است. این نوع از پردازش موازی در علم دینامیک سیالات محاسباتی نیز جایگاه ویژه‌ای پیدا کرده است. محققان به شبیه‌سازی پدیده‌های مختلف طبیعی و مهندسی با کمک پردازنده‌های گرافیکی پرداخته و به افزایش سرعت قابل توجهی دست یافته‌اند. در این مقاله دو الگوریتم حلگر دستگاه معادلات سه‌قطری با عناوین الگوریتم کاهش متناوب و کاهش متناوب موازی به کار برده شده‌اند. همچنین رهیافت جدیدی برای حل این نوع دستگاه معادلات با نام رهیافت توماس موازی نیز ارائه شده است. جریان درون حفره برای ارزیابی این الگوریتم‌ها استفاده شده و نتایج با الگوریتم توماس کلاسیک اجرا شده روی پردازنده مرکزی، هم از نظر زمان حل و هم از نظر مقادیر عددی کمیت‌های فیزیکی مورد مقایسه قرار گرفته‌اند. بیشینه افزایش سرعتی که الگوریتم کاهش متناوب، کاهش متناوب موازی و رهیافت توماس موازی اجرا شده در پردازنده گرافیکی در برابر الگوریتم توماس کلاسیک (اجرا شده روی پردازنده مرکزی) از خود نشان می‌دهند به ترتیب در حدود $4/4$ ، $2/5$ و $45/38$ می‌باشد. همچنین در این تحقیق، اثر دسترسی هم‌مکان و غیر هم‌مکان به حافظه سراسری نیز مورد بحث و بررسی قرار گرفته و نشان داده شده است که با دسترسی هم‌مکان، افزایش سرعت حدوداً دو برابری برای پردازنده گرافیکی حاصل می‌شود.

از طرفی دیگر این رهیافت در مسائل وابسته به زمان هم می‌تواند کاربرد داشته باشد. روش‌های متعددی موجود هستند که با گسسته کردن معادلات شامل مشتقات زمانی را به معادلات سه‌قطری تبدیل می‌کنند مانند روش ضمنی جهت متناوب. برای بررسی عملکرد این رهیافت مسأله وابسته به زمان عبور جریان از روی استوانه در نظر گرفته شده است که با اعمال رهیافت توماس یک ضریب تسریع حدود ۹ برابر حاصل شد.

واژه‌های کلیدی: دستگاه معادلات سه‌قطری، پردازنده گرافیکی، رهیافت توماس، جریان عبوری

از روی استوانه، روش ضمنی جهت متناوب

فهرست مطالب

فصل اول: ۱

۱-۱ مقدمه ۲

۲-۱ مروری بر استفاده از پردازنده‌های گرافیکی در دینامیک سیالات محاسباتی ۸

فصل دوم: ۱۷

۱-۲ مقدمه ۱۸

۲-۲ محاسبه موازی ۱۸

۳-۲ معماری کامپیوتر موازی ۲۱

۱-۳-۲ سیستم حافظه مشترک ۲۳

۲-۳-۲ سیستم حافظه توزیع شده ۲۴

۴-۲ شبکه‌های ارتباط داخلی ۲۴

۱-۴-۲ مدل برنامه نویسی موازی ۲۵

۲-۴-۲ مدل حافظه مشترک ۲۶

۳-۴-۲ مدل انتقال پیام ۲۸

۵-۲ الگوریتم‌های موازی ۲۹

- ۳۰ ۱-۵-۲ طراحی الگوریتم موازی
- ۳۱ ۶-۲ معیار ارزیابی سیستم موازی
- ۳۳ ۷-۲ یک نمونه الگوریتم موازی

۳۷ فصل سوم:

- ۳۸ ۱-۳ مقدمه
- ۳۸ ۲-۳ معماری پردازنده گرافیکی
- ۴۱ ۳-۳ اجزای سخت‌افزاری و نرم‌افزاری پردازنده گرافیکی
- ۴۱ ۱-۳-۳ چندپردازنده
- ۴۲ ۲-۳-۳ ریسمان
- ۴۳ ۳-۳-۳ انشعاب
- ۴۴ ۴-۳-۳ انواع حافظه در پردازنده گرافیکی
- ۴۹ ۵-۳-۳ ملاحظات نرم‌افزاری و برنامه‌نویسی روی پردازنده‌های گرافیکی

۵۳ فصل چهارم:

- ۵۴ ۱-۴ مقدمه
- ۵۵ ۲-۴ دستگاه معادلات سه‌قطری
- ۵۶ ۱-۲-۴ الگوریتم توماس کلاسیک
- ۵۶ ۲-۲-۴ الگوریتم کاهش متناوب
- ۵۸ ۳-۲-۴ الگوریتم کاهش متناوب موازی

۴-۲-۴	الگوریتم توماس موازی	۶۰
۳-۴	شبیه سازی جریان درون حفره با استفاده از پیاده سازی معادلات سه قطری روی پردازنده گرافیکی	۶۲
۴-۴	شبیه سازی جریان با روش تابع جریان - تاوایی	۶۳
۵-۴	هندسه مسأله و گسسته سازی	۶۶
۶-۴	نتایج	۷۰

فصل پنجم: ۸۱

۱-۵	مقدمه	۸۲
۲-۵	شبیه سازی جریان روی استوانه مربعی، معادلات حاکم و پیاده سازی روی پردازنده گرافیکی	۸۴
۱-۲-۵	معادلات حاکم	۸۶
۲-۲-۵	شرایط مرزی	۹۱
۳-۵	نتایج	۹۳
۱-۳-۵	بررسی استقلال از شبکه	۹۳
۲-۳-۵	جریان پایا	۹۴
۳-۳-۵	جریان ناپایا	۹۷
۴-۳-۵	بررسی پیاده سازی روی پردازنده گرافیکی	۱۰۴

فصل ششم: ۱۰۹

۱-۶ مروری بر مطالعه حاضر و نتیجه گیری ۱۱۰

۲-۶ پیشنهادات ۱۱۲

منابع و مراجع ۱۱۴

فهرست اشکال

- شکل ۱-۱: خنک کاری یک پردازنده با کلاک بالاتر از ۴ گیگا هرتز توسط نیتروژن مایع با دمای (۱۹۵-) درجه سانتی گراد [1] ۳
- شکل ۲-۱: تغییرات سرعت پردازنده‌های مرکزی از ابتدا تاکنون [1] ۵
- شکل ۱-۲: شکلی شماتیک از محاسبات سریال ۱۸
- شکل ۲-۲: شکلی شماتیک از محاسبات موازی ۱۹
- شکل ۳-۲: انواع معماری مدل MIMD ۲۳
- شکل ۴-۲: انواع شبکه‌های ارتباطی ایستا ۲۵
- شکل ۵-۲: مدل PRAM برای محاسبات موازی ۲۸
- شکل ۶-۲: مثالی از الگوریتم Sum_EREW با ۸ ورودی ۳۵
- شکل ۱-۳: مقایسه ساختار معماری پردازنده مرکزی و گرافیکی ۴۰
- شکل ۲-۳: شکل شماتیک یک چندپردازنده (الف) معماری فرمی ب) معماری تسلا ۴۱
- شکل ۳-۳: انشعاب ایجاد شده در هسته‌های پردازشی و نخها توسط دستور شرطی if [22] ۴۴
- شکل ۴-۳: سلسله مراتب حافظه در کارت گرافیک [22] ۴۴
- شکل ۵-۳: دسترسی غیر همزمان نخهای محاسباتی به حافظه سراسری ۴۸
- شکل ۶-۳: دسترسی مناسب و همزمان نخهای محاسباتی به حافظه سراسری ۴۸
- شکل ۷-۳: رابطه بین انواع حافظه و نخهای محاسباتی و ارتباط کلی بین میزبان و پردازنده مرکزی ۴۹

- شکل ۳-۸: ساختار اجرای برنامه با کمک پردازنده گرافیکی و تقسیم‌بندی نخ‌ها در هنگام فراخوانی کرنل‌ها..... ۵۱
- شکل ۴-۱: نمایش ماتریس سه‌قطری ۵۵
- شکل ۴-۲: الگوریتم کاهش متناوب برای یک دستگاه ۸ معادله و ۸ مجهول ۵۸
- شکل ۴-۳: مراحل الگوریتم کاهش متناوب موازی برای یک دستگاه هشت مجهولی ۵۹
- شکل ۴-۴: یک مرحله از پیشرو از الگوریتم کاهش متناوب موازی روی یک دستگاه با چهار مجهول که منجر به ایجاد دو دستگاه مستقل می‌شود ۶۰
- شکل ۴-۵: ساختار نخ‌های محاسباتی برای اجرای رهیافت توماس موازی روی یک شبکه عددی ۶۲
- شکل ۴-۶: هندسه مسأله جریان درون حفره ۶۷
- ۴-۷: شبکه عددی تولید شده و شرایط مرزی برای مسأله جریان درون حفره ۶۸
- شکل ۴-۸: نمایش مقدار تاوایی در مرز متحرک برای سه عدد رینولدز ۱۰۰، ۴۰۰ و ۱۰۰۰ حاصل از نتایج: (الف) رهیافت توماس موازی (ب) الگوریتم کاهش متناوب (ج) الگوریتم کاهش متناوب موازی ۷۲
- شکل ۴-۹: نمایش مؤلفه عمودی سرعت روی خط $Y = 0.5$ برای سه عدد رینولدز ۱۰۰، ۴۰۰ و ۱۰۰۰ حاصل از نتایج: (الف) رهیافت توماس موازی (ب) الگوریتم کاهش متناوب (ج) الگوریتم کاهش متناوب موازی ۷۳
- شکل ۴-۱۰: نمایش مؤلفه افقی سرعت روی خط $X = 0.5$ برای سه عدد رینولدز ۱۰۰، ۴۰۰ و ۱۰۰۰ حاصل از نتایج: (الف) رهیافت توماس موازی (ب) الگوریتم کاهش متناوب (ج) الگوریتم کاهش متناوب موازی ۷۴

- شکل ۴-۱۱: کانتورهای تابع جریان در جریان درون حفره با الگوریتم توماس موازی ۷۵
- شکل ۴-۱۲: کانتورهای تاوایی در جریان درون حفره با الگوریتم توماس موازی ۷۶
- شکل ۴-۱۳: نمایش زمان حل الگوریتم‌های مختلف برای رینولدز ۱۰۰۰ در طی ۱۰۰۰ تکرار ۷۷
- شکل ۴-۱۴: نمایش افزایش سرعت زمان حل الگوریتم‌های موازی برای رینولدز ۱۰۰۰ ۷۷
- شکل ۴-۱۵: مقایسه افزایش سرعت پردازش موازی در برابر پردازنده مرکزی در حالت دسترسی هم‌مکان و غیرهم‌مکان به حافظه برای تعداد گره محاسباتی مختلف در رینولدز ۱۰۰۰ ۷۸
- شکل ۵-۱: هندسه جریان روی استوانه مربعی ۸۴
- شکل ۵-۲: شبکه عددی یکنواخت برای حل جریان روی استوانه مربعی ۸۵
- شکل ۵-۳: نمایش مراحل روش ضمنی جهت متغیر ۸۸
- شکل ۵-۴: آرایش نخ‌های محاسباتی در مرحله اول اعمال روش ضمنی جهت متناوب ۹۰
- شکل ۵-۵: آرایش نخ‌های محاسباتی در مرحله دوم اعمال روش ضمنی جهت متناوب ۹۰
- شکل ۵-۶: بررسی طول گردابه‌های ایجاد شده پشت استوانه در جریان پایا در رینولدزهای مختلف برای بررسی استقلال از شبکه ۹۴
- شکل ۵-۷: مقایسه طول گردابه‌های ایجاد شده در پشت سیلندر در رینولدزهای متفاوت از ۱۰ تا ۴۰ و مقایسه با نتایج گزارش شده در مرجع [37] ۹۵
- شکل ۵-۸: خطوط تابع جریان برای رینولدزهای: الف) ۱ (ب) ۳۰ (ج) ۵۰ ۹۶
- شکل ۵-۹: خطوط تابع جریان طی ۱۰ ثانیه برای رینولدز ۱۰۰ ۹۸
- شکل ۵-۱۰: خطوط تابع جریان طی ۱۰ ثانیه برای رینولدز ۲۰۰ ۹۹
- شکل ۵-۱۱: مولفه‌های سرعت الف) عمودی و ب) افقی در طول خط $Y = 0$ و در لحظه‌های

که مولفه عمودی سرعت در مکان $X = 10.5$ صفر میشود و پروفیل تغییر علامت می‌دهد. ۱۰۰

شکل ۵-۱۲: پروفیل‌های سرعت الف) عمودی و ب) افقی در سه مقطع عرضی مختلف

$X = 0,4,8$ که در لحظه $t = 105.5$ به نمایش در آمده‌اند. ۱۰۱

شکل ۵-۱۳: خطوط همتراز تاوایی به ترتیب از بالا به پایین برای رینولدزهای ۷۵، ۱۰۰، ۱۵۰ و

۲۰۰ در لحظه $t = 100 s$ ۱۰۲

شکل ۵-۱۴: ضریب درگ برای سه رینولدز مختلف ۱۰۲

شکل ۵-۱۵: روند کاهش خطا از گام زمانی دوم به بعد در رینولدز ۱۰۰ ۱۰۳

شکل ۵-۱۶: تغییرات مقدار میانگین تابع جریان روی استوانه برای رینولدز ۲۰۰ در طی زمان

۱۰۰ تا ۱۲۰ ۱۰۴

شکل ۵-۱۷: افزایش سرعت در رهیافت توماس موازی نسبت به زمان حل پردازنده مرکزی ۱۰۷

شکل ۵-۱۸: سهم زمانی توابع مختلف اجرا شده روی پردازنده گرافیکی از یکبار اجرای کل

برنامه ۱۰۸

فهرست علائم

i	شمارنده گره محاسباتی در راستای محور طول
j	شمارنده گره محاسباتی در راستای محور عرض
L	طول (m)
m	تعداد سطر
n	تعداد ستون
U_0	سرعت روی مرز متحرک (ms^{-1})
u	مؤلفه افقی سرعت (ms^{-1})
U	مؤلفه افقی سرعت (بدون بعد)
v	مؤلفه عمودی سرعت (ms^{-1})
V	مؤلفه عمودی سرعت (بدون بعد)
x	محور طول (m)
X	طول (بدون بعد)
y	محور عرض (m)
Y	عرض (بدون بعد)

علائم یونانی

ψ	تابع جریان ($\text{m}^2 \text{s}^{-1}$)
ω	تاوایی (s^{-1})
Ω	تاوایی
ν	لزجت سینماتیکی ($\text{m}^2 \text{s}^{-1}$)

فصل اول:

مقدمه

۱-۱ مقدمه

معادلات ناویر استوکس^۱ برای شبیه‌سازی جریان سیالات در رشته‌های مختلف علمی و مهندسی استفاده می‌شوند. این معادلات در تحقیقات تغییرات اقلیمی و پیش‌بینی وضعیت آب‌وهوا، در تحقیقات پزشکی مانند بررسی جریان خون و بررسی چگونگی حرکت داروها در شریان‌ها و در مهندسی در طراحی اتومبیل، هواپیما و غیره کاربرد فراوانی دارند. حل دقیق و شبیه‌سازی مسائل مهندسی مرتبط با جریان سیالات اغلب دارای محاسبات سنگین و وقت‌گیر کامپیوتری هستند که هم به زمان و هم به حافظه‌ی بسیار زیادی احتیاج دارد. این شبیه‌سازی‌ها به راحتی می‌توانند ساعت‌ها و یا روزها وقت بگیرند. در بعضی موارد مانند بررسی و پیش‌بینی وضع آب‌وهوا سرعت در تجزیه تحلیل داده‌ها امری ضروری است، برای مثال اگر برای پیش‌بینی قدرت، سرعت و جهت حرکت یک طوفان اگر ساعت‌ها صرف محاسبات شود ممکن است قبل از رسیدن طوفان محاسبات به پایان نرسیده باشد و در اثر نداشتن اطلاعات دقیق از کم و کیف وضعیت جوی، خسارات فراوانی به بار آید. بنابراین افزایش سرعت در حل مسائل می‌تواند بسیار مشکل‌گشا باشد و به محققان و طراحان اجازه خواهد داد تا مسائل را با دقت و سرعت بیشتری بررسی کنند.

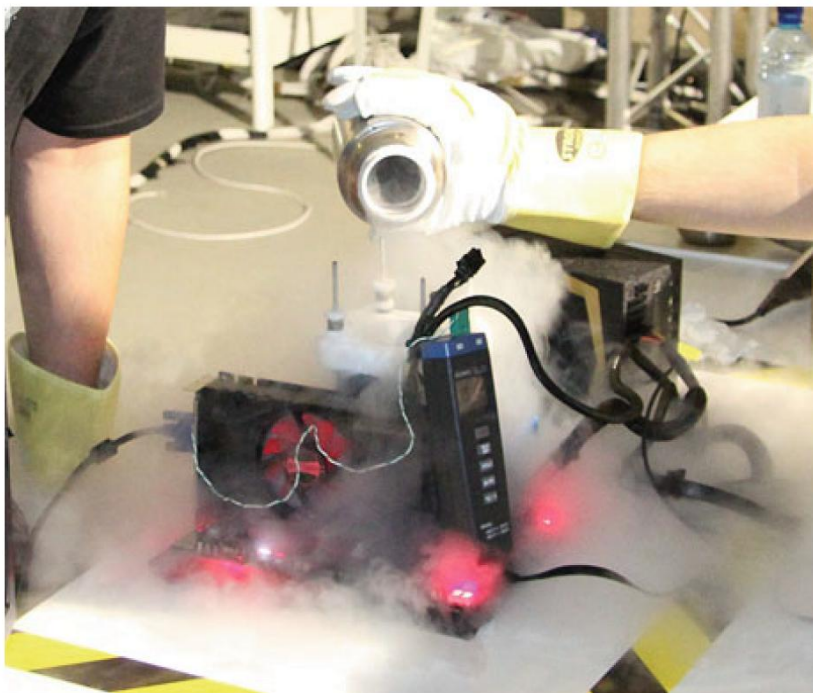
برای دستیابی به سرعت پردازش بیشتر، تولیدکنندگان پردازنده مرکزی^۲ تا سال ۲۰۰۳ برای افزایش قدرت پردازشی پردازنده، عمدتاً بر افزایش تعداد ترانزیستورهای موجود در پردازنده مرکزی و افزایش سرعت پردازش هسته (کلاک^۳: سرعت پردازش با واحد هرتز) تکیه داشتند. اما در این سال مشاهده کردند که با افزایش سرعت هسته‌ی پردازشی پردازنده، حرارت بسیار بالایی تولید می‌شود که دفع این حرارت به تمهیدات ویژه‌ای نیاز دارد که در واقع برای عموم قابل‌استفاده

^۱ Navier Stokes Equations

^۲ Central Processor Unit(CPU)

^۳ Clock

نخواهند بود و در شرایط خاص پیاده‌سازی و استفاده از چنین سرعتی از لحاظ عملی مشکل خواهد بود. در واقع امروزه این امکان وجود دارد به کلاک بالاتر از ۴ گیگاهرتز برای یک پردازنده رسید اما خنک نگاه داشتن پردازنده کار آسانی نخواهد بود (شکل ۱-۱).



شکل ۱-۱: خنک‌کاری یک پردازنده با کلاک بالاتر از ۴ گیگاهرتز توسط نیتروژن مایع با دمای (۱۹۵-)
درجه سانتی‌گراد [1]

راه‌حل دیگر در دستیابی به سرعت پردازش بالا استفاده از روش پردازش موازی^۱ است. به بیان ساده در این روش چندپردازنده‌ی معمولی با همکاری یکدیگر به اجرای یک برنامه می‌پردازند که طی این همکاری، برنامه با سرعت بالاتری اجرا می‌شود. برای اینکه بتوان از امکانات و تجهیزات یک سیستم موازی استفاده نمود، نیاز به نرم‌افزارها و سیستم‌عامل‌هایی است که موازی‌سازی‌ها پشتیبانی کنند، همچنین برای حل مسائلی که قابلیت موازی‌سازی را دارند، لازم است که

¹ Parallel Processing

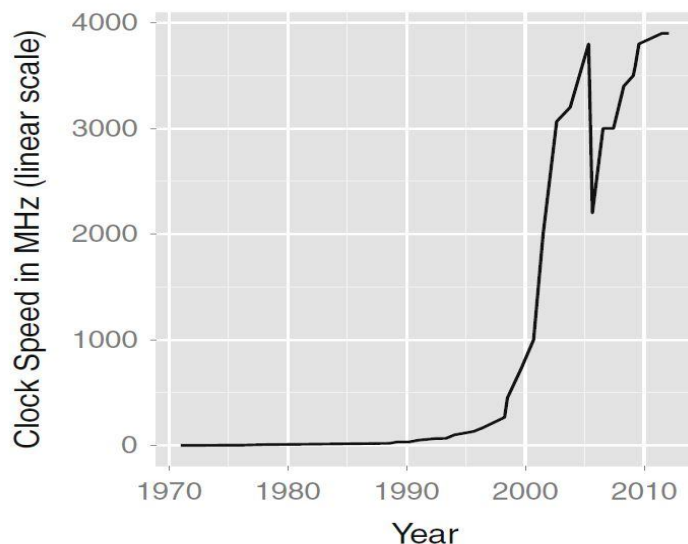
الگوریتم‌هایی موازی نیز ارائه شوند. اولین گام در برنامه‌نویسی موازی این است که طراح برنامه، بتواند به صورت موازی فکر کند، مسأله را به درستی شناخته و بر روی قسمت‌هایی از مسأله که قابلیت اجرای همزمان را دارند، تمرکز کند. در غیر این صورت استفاده از پردازش موازی ممکن نخواهد بود و یا نتیجه معکوس خواهد داد و به جای اینکه افزایش سرعت در زمان حل مسأله داشته باشیم، کاهش سرعت خواهیم داشت.

رایانه‌های خوشه‌ای^۱ اولین نوع از سامانه‌ها با قابلیت پردازش موازی بودند. رایانه خوشه‌ای به دسته‌ای از رایانه‌ها (حداقل دو رایانه) اطلاق می‌شود که برای انجام یک هدف مشترک با هم همکاری می‌کنند. اولین سیستم خوشه‌ای در سال ۱۹۹۴ در سازمان تحقیقات فضایی آمریکا پیاده‌سازی شد. این سیستم از ۱۶ رایانه خانگی که توسط یک شبکه به هم متصل شده بودند به وجود آمد. نکته جالب اینکه هزینه این پروژه تنها ۴٪ هزینه یک پردازنده با توان پردازشی معادل این ۱۶ رایانه بود. اما رایانه‌های خوشه‌ای با اینکه امروزه در بسیاری مراکز علمی و تحقیقاتی به کار برده می‌شوند اما در دسترس همگان نمی‌تواند باشد.

بعد از مدتی در سال ۲۰۰۳ ایده‌ی پردازنده‌ی مرکزی با دو یا چند هسته پردازشی به وجود آمد که هر هسته می‌تواند مستقل از هسته‌ی دیگر با حافظه در ارتباط باشد و دستوراتی مجزا از همتای خود اجرا کند که این نوع پردازش که سیستم‌عامل از آن بهره می‌برد نیز نوعی پردازش موازی است. بدین ترتیب فصل جدیدی روی پردازنده‌های مرکزی گشوده شد. به مرور زمان تعداد هسته‌های پردازشی افزایش یافتند و به ۴، ۸ و ۱۲ هسته رسیدند بنابراین با تعداد هسته پردازشی بالاتر نیاز به کلاک کمتری در مقایسه با تعداد هسته کمتر بود. همچنین با تولید چنین پردازنده‌هایی نرم‌افزارهای علمی نیز طوری برنامه‌ریزی شدند که در صورت لزوم بتوانند از قدرت

¹ Cluster Copmuters

تمام هسته‌ها بهره ببرند. اما پس از مدتی مجدداً با پیشرفت نرم‌افزارها و همچنین پیچیده‌تر شدن محاسبات علمی علاوه بر تعداد هسته زیاد کلاک بالاتر نیز مورد احتیاج بود. در شکل ۲-۱ تاریخچه تغییرات کلاک پردازنده‌های مرکزی با گذشت زمان مشخص شده است.



شکل ۲-۱: تغییرات سرعت پردازنده‌های مرکزی از ابتدا تاکنون [1]

با توجه به شکل ۲-۱ سرعت در این سالیان متمادی از ۴ گیگاهرتز بالاتر نرفته است، اگرچه رسیدن به کلاک بالاتر از ۴ گیگاهرتز امکان‌پذیر است اما در این کلاک همان‌طور که گفته شد حرارت ایجاد شده باعث می‌شود که سازندگان از یک حدود سرعت پردازش بالاتر نروند و به این مشکل تولید حرارت زیاد، دیوار حرارتی می‌گویند. همچنین بعد از سال ۲۰۰۳ یک افت در سرعت کلاک وجود دارد که ناشی از تولید پردازنده‌های چند هسته‌ای است [1].

از طرفی دیگر کارت‌های گرافیکی خود بر مبنای فلسفه پردازش موازی طراحی شده‌اند و قابلیت بالقوه پردازش موازی را دارا هستند. پردازنده‌های گرافیکی از تعداد زیادی هسته پردازشی

با سرعت پردازش نه‌چندان بالا و همچنین پهنای باند حافظه^۱ وسیع برای انتقال همزمان حجم بالایی از اطلاعات، تشکیل شده‌اند. اما چرا کارت گرافیکی به چنین چیزی احتیاج دارد؟ در واقع جواب این سؤال در صفحه نمایشگر نهفته است! هر نمایشگر از تعداد بسیار زیادی نقاط نورانی به نام پیکسل^۲ ساخته شده است. هر پیکسل خود از چند واحد نورانی دیگر به نام زیر پیکسل تشکیل شده است که هرکدام از این زیر پیکسل‌های کوچک قابلیت نمایش یک رنگ را دارند. وظیفه کارت گرافیک محاسبه رنگ و شدت رنگ تک‌تک این ریز پیکسل‌ها است. پردازش مربوط به هر پیکسل بسیار ساده است اما تعداد آن بسیار زیاد است. اگر پردازنده مرکزی بخواهد این عملیات را انجام دهد باید از اولین پیکسل شروع کند و به ترتیب تمام پیکسل‌های صفحه نمایش را پوشش دهد که این روش پردازش جوابگوی چنین نیازی نیست و نتیجه خوبی در بر ندارد. برای مثال زمانی که درایور کارت گرافیک نصب نباشد صفحه نمایشگر برای نشان دادن تصاویر ثابت هیچگونه لرزشی نخواهد داشت اما حتی اگر یک پنجره ی ساده را جابه‌جا شود پردازنده مرکزی از پس این کار ساده بر نخواهد آمد! پس اینجا بحث پردازنده ای مطرح می‌شود که بتواند تعداد زیادی از داده‌های بسیار ساده را پردازش کند که همان کارت‌های گرافیکی و به‌طور خاص پردازنده‌های گرافیکی^۳ هستند.

اما آیا کارت‌های گرافیکی از لحاظ سخت‌افزاری طراحی شده‌اند تا فقط رنگ‌ها را محاسبه کنند؟! مسلماً نه! یک پردازنده گرافیکی از تعداد زیادی پردازنده ساخته شده که می‌تواند چهار عمل اصلی را انجام دهند. اما دلیل اینکه برای سال‌ها پتانسیل نهفته این نوع پردازنده‌ها استفاده نشده بود تماماً به دلیل مشکلات نرم‌افزاری بود. در گذشته تعداد کمی از محققین به این پتانسیل روی

¹ Memory band Width

² Pixel

³ Graphic Processor Unit (GPU)

آوردند اما به دلیل مشکلات بسیار زیاد در برنامه‌نویسی، راه به جایی نبردند. مهم‌ترین مشکل پیش روی این افراد این بود که تنها راه برقراری ارتباط با کارت گرافیک از طریق نرم‌افزارهای خاصی مثل دایرکت‌ایکس^۱ و اپن جی ال^۲ بودند و این واسطه‌های نرم‌افزاری فقط و فقط مخصوص کار با پیکسل‌ها ساخته شده بودند. پس برای جمع دو عدد باید ابتدا آن‌ها رو به دو رنگ تبدیل، سپس نتیجه تلاقی این دو رنگ را به دست آورده حاصل را دوباره به عدد تبدیل کرد. که این امر دشواری فراوانی برای برنامه‌نویس ایجاد می‌کند. به همین دلیل تا سال‌ها استفاده از قدرت پردازنده‌های گرافیکی در محاسبات امکان‌پذیر نبود. اولین تلاش‌ها برای به کار بردن پردازنده‌های گرافیکی در محاسبات علمی و عددی در سال ۲۰۰۱ رخ داد. در این سال لارسن و مک‌آلیستر [2] برای اولین با نتایج تحقیقی را منتشر کردند که توسط کار با مفاهیم گرافیکی نظیر رنگ‌ها توانستند دو ماتریس را در هم ضرب کنند. اما این کار بسیار مشکل است و در محاسبات پیچیده می‌تواند اشکالات و موانع زیادی ایجاد کند. به عنوان یک نمونه دیگر در سال ۲۰۰۵ گالوپو و همکاران [3] جز اولین کسانی بودند که با استفاده از پردازنده گرافیکی و با روش فاکتورگیری ال یو^۳ دستگاه معادلات را سریع‌تر از کد مشابه روی پردازنده مرکزی حل کردند. اما همچنان کار و ارتباط با کارت و پردازنده گرافیکی مشکل و سخت بود.

اما در سال ۲۰۰۶ شرکت انویدیا^۴ توانست یک پل ارتباطی بین کاربر و کارت گرافیک ایجاد کند که برای برنامه‌نویسی و محاسبات کار دشواری پیش‌رو نباشد. این میان‌افزار کودا^۵ نامیده شد. با استفاده از میان‌افزار کودا می‌توان در کنار قدرت پردازنده مرکزی از قدرت پردازنده‌ی گرافیکی در

¹ DirectX

² OpenGL

³ LU Factorization

⁴ Nvidia

⁵ CUDA(Compute Unified Device Architecture)

یک برنامه محاسباتی بهره برد. کودا بستری فراهم می کند که در زمانی که بتوان قسمتی از برنامه را به صورت موازی پردازش کرد، اطلاعات را به پردازنده گرافیکی منتقل و پردازش کرد سپس با انتقال مجدد نتایج ادامه محاسبات روی پردازنده مرکزی پیگیری شود. به این نوع پردازش، پردازش ناهمگن^۱ می گویند. کودا صرفاً برای پردازنده های شرکت انویدیا طراحی شده است. از طرفی دیگر میان افزار اپن سی ال^۲ نیز نقش یک رابط را برای کابر و پردازنده های ساخت شرکت ای تی آی^۳ بازی می کند.

به این ترتیب مانع بزرگ بین محققان و پتانسیل بالای پردازنده های گرافیکی برای پردازش، برداشته شد و عصر جدیدی از پردازش موازی به نام پردازش همه منظوره با پردازنده های گرافیکی^۴ آغاز شد. پس از آن در زمینه های مختلف علمی استفاده از پردازنده های گرافیکی رواج پیدا کرد، از جمله علوم مربوط به مکانیک سیالات و به طور خاص دینامیک سیالات محاسباتی^۵ که با توجه به پردازش های حجیم و زمان بری که در این علم وجود دارد، توجه محققان در این رشته نیز به پردازش با پردازنده های گرافیکی جلب کرده است.

۲-۱ مروری بر استفاده از پردازنده های گرافیکی در دینامیک سیالات

محاسباتی

همان طور که در بخش گذشته اشاره شد با آغاز قرن جدید تلاش ها برای استفاده از پردازنده گرافیکی به عنوان یک پردازش گر موازی به طور مستقیم شروع شد و با پیدایش میان افزارهایی مانند کودا فصل جدیدی از پردازش موازی آغاز گردید.

^۱ Heterogeneous Process

^۲ Open CL

^۳ ATI

^۴ General Purpose Graphical Processor Unit (GPGPU)

^۵ Computational Fluid Dynamics (CFD)

در دینامیک سیالات محاسباتی با توجه به اینکه بعد از گسسته سازی معادلات حاکم یک دستگاه معادلات حاصل می‌شود بخش بزرگی از تحقیقات روی پیاده‌سازی روش‌های حل دستگاه معادلات و اجرای آن‌ها روی پردازنده گرافیکی انجام گرفته است. با در نظر گرفتن این موضوع که بسیاری از الگوریتم‌های موجود برای حل و شبیه‌سازی یک پدیده فیزیکی و مکانیکی ترتیبی هستند، یافتن راه‌حلی دیگر یا یک الگوریتم جایگزین یک چالش جدی محسوب می‌شود. البته مفهوم پردازش موازی و استفاده از آن با اتصال مجموعه‌ای از پردازنده‌های مرکزی تا قبل از پیدایش میان‌افزارهایی مانند کودا رواج داشته است، بنابراین الگوریتم‌های موازی مختلفی نیز تا آن زمان ارائه شده بودند اما پیاده‌سازی این الگوریتم‌ها روی پردازنده گرافیکی با توجه به معماری آن (توضیحات بیشتر در فصل ۳) و انتخاب یا ارائه یک الگوریتم متناسب با پردازنده گرافیکی و بهینه‌سازی و پیاده‌سازی، مباحثی هستند که محققان علوم محاسباتی را به خود مشغول کرده‌اند.

همان‌طور که گفته شد، تقریباً تمامی تلاش‌ها برای حل و بررسی عددی مسائل مهندسی منجر به یک سری دستگاه معادلات جبری خطی یا غیرخطی می‌شوند. این دستگاه‌ها می‌توانند متراکم^۱ یا نامتراکم^۲ (تنک) باشند. یعنی در حالت اول ماتریس ضرایب دارای درایه‌هایی اکثراً مقدار دار است، و در حالت دوم اکثر درایه‌های ماتریس صفر هستند مانند ماتریس سه‌قطری^۳. در تحقیقات مختلف روی هردو نوع این دستگاه‌ها بررسی‌هایی انجام شده است. با توجه به اینکه در ماتریس‌های نامتراکم اکثر درایه‌ها صفر هستند برای انجام یک عمل جبری مانند ضرب عاقلانه نیست که تمام درایه‌ها در یکدیگر ضرب شوند، و این هزینه محاسباتی سنگینی ایجاد می‌کند. پس بهتر است از الگوریتم‌های متناسب با این‌گونه ماتریس‌ها استفاده شوند. در اجرا و پیاده‌سازی

¹ Dense

² Sparse

³ TriDiagonal Matrix

عملیات جبری و محاسباتی مختلف روی پردازنده‌های گرافیکی، دستگاه‌های متراکم افزایش سرعت قابل توجهی از خود نشان داده‌اند اما در دستگاه‌های نامتراکم افزایش سرعت حل به اندازه دستگاه‌های متراکم نبود. اما دستگاه‌های نامتراکم یکی از کلیدی‌ترین قسمت‌های علوم محاسباتی هستند که در بسیاری از کاربردهای مهندسی با چنین دستگاه‌هایی مواجه می‌شویم و همان‌طور که گفته شد افزایش سرعت چشم‌گیری هم در پردازش موازی از خود نشان نداده‌اند پس می‌توان بررسی این نوع ماتریس‌ها روی پردازنده گرافیکی را یک مبحث پر چالش در پردازش موازی نامید. پرکاربردترین نوع دستگاه‌های نامتراکم، دستگاه‌های سه‌قطری هستند که موضوع این پایان نامه هم به بررسی و پیاده‌سازی دستگاه‌های سه‌قطری به شکل موازی اختصاص دارد.

در مورد تحقیقاتی که در مورد دستگاه‌های نامتراکم ارائه شده، برای مثال می‌توان به عملیات مختلف جبری روی ماتریس- بردار نامتراکم^۱ اشاره کرد. در سال ۲۰۰۸ بل و همکاران [4] عملیات جبری روی فرم‌های مختلف ماتریس نامتراکم را توسط میان‌افزار کودا بررسی کردند و نتایج را با پیاده‌سازی روی پردازنده مرکزی مقایسه کردند. آن‌ها خاطرنشان کردند که استفاده از حافظه بافت^۲ در پردازنده گرافیکی می‌تواند تا ۲۵٪ راندمان را افزایش دهد. باسکاران و همکاران [5] با پیاده‌سازی الگوریتم‌های ماتریس- بردار نامتراکم و به‌طور خاص روی ساختار ذخیره‌سازی ردیف فشرده^۳، افزایش سرعت و نتایج را با توابع کتابخانه‌ای انویدیا مقایسه کرده‌اند.

حل‌گرهای تکراری پیش شرط‌سازی شده در حل با پردازنده‌های گرافیکی در مرجع [6] مطالعه شده‌اند. تمام مراحل پیش‌شرط‌سازی و حل‌گرها، روی پردازنده گرافیکی توسط رویپنگ و همکاران انجام شده است. آن‌ها گزارش کرده‌اند که عملیات جبری ماتریس- بردار نامتراکم روی پردازنده

¹ Sparse Matrix-Vector

² Texture Memory

³ Compressed Row Storage(CRS)

گرافیکی پرسرعت‌تر از مشابه آن روی پردازنده مرکزی هستند. همچنین مطالعاتی روی فرم‌های ذخیره‌سازی و اثرات آن‌ها روی پردازنده گرافیکی انجام شده است. همچنین آن‌ها روش‌های گرادین مزدوج پیش‌شرط‌سازی شده به روش چولسکی ناقص را هم روی پردازنده گرافیکی پیاده‌سازی کردند و به افزایش سرعت متوسطی (۵ برابر پردازش موازی با پردازنده‌های مرکزی) رسیدند. در سال ۲۰۰۸ مولمیکر و همکاران [7] روش چندشبکه‌ای^۱ را برای حل معادله پواسون فشار روی پردازنده گرافیکی پیاده‌سازی کردند. افزایش سرعت حل به حدود ۵۰ برابر سرعت پردازنده مرکزی نیز رسید. البته در تحقیق اشاره شده است که روش به‌کاربرده شده برای هندسه‌های پیچیده کاربرد ندارد.

چند تحقیقی که تاکنون معرفی شد مربوط به حالت کلی ماتریس‌های نامتراکم بود. اما یکی از پرکاربردترین انواع دستگاه‌های نامتراکم، دستگاه معادلات سه‌قطری هستند. یکی از روش‌های حل دستگاه‌های سه‌قطری، حل توسط الگوریتم توماس^۲ که بر مبنای روش حذفی گوس^۳ است. این الگوریتم یک روش بسیار قدرتمند، سریع و کم‌هزینه از نظر محاسباتی است و استفاده از آن رواج بسیاری دارد. اما این الگوریتم یک الگوریتم ترتیبی است و قابلیت اجرای مستقیم روی پردازنده گرافیکی ندارد، اما برای حل این موضوع، الگوریتم‌های جایگزینی برای حل معادلات سه‌قطری با رویکرد پردازش موازی در پردازنده‌های گرافیکی نظیر کاهش متناوب^۴ و کاهش متناوب موازی^۵ معرفی شده‌اند. این الگوریتم‌ها و نظایر آن‌ها که به همین منظور ابداع شده‌اند (مانند الگوریتم‌های موجود در مرجع [8]) ابزارهای مناسبی برای بسیاری از محققین هستند که به دنبال تقویت

¹ MultiGrid

² Thomas Algorithm

³ Gauss Elimination

⁴ Cyclic Reduction(CR)

⁵ Parallel Cyclic Reduction(PCR)

عملکرد کاربرد پردازنده‌های گرافیکی در حل مسائل مختلف می‌باشند.

ژانگ و همکاران [8] در سال ۲۰۰۹ ابتدا روی قابلیت اجرای الگوریتم‌های کاهش متناوب و کاهش متناوب موازی روی پردازنده‌های گرافیکی بحث کرده‌اند، سپس یک روش ترکیبی از الگوریتم‌های معرفی شده را پیشنهاد و عملکرد هر الگوریتم را روی حل ماتریس‌های سه‌قطری آزمایش کردند. لازم به توضیح است که الگوریتم کاهش متناوب مشکل تداخل حافظه^۱ در حافظه مشترک^۲ کارت گرافیکی را دارد و همچنین برای به کار بردن الگوریتم‌های کاهش متناوب و کاهش متناوب موازی اندازه دستگاه سه‌قطری حتماً باید محدود به^۳ باشد. در سال ۲۰۱۱ گودک و همکاران [9] در مورد محدودیت ابعاد ماتریس در حافظه مشترک کارت گرافیکی بحث کرده‌اند. آن‌ها نشان دادند که انتقال پیاپی اطلاعات بین حافظه مشترک و حافظه سراسری^۳ کارت گرافیکی بر عملکرد الگوریتم تأثیر می‌گذارد. در سال ۲۰۱۱ دیویدسون و همکاران [10] یک الگوریتم چندمرحله‌ای برای حل دستگاه معادلات سه‌قطری با پردازنده گرافیکی پیشنهاد کردند که نتایج کار آن‌ها نشان می‌داد سرعت الگوریتم پیشنهادی تا ۵ برابر افزایش سرعت در مقابل الگوریتم‌های حل غیرموازی دستگاه معادلات سه‌قطری به دست آورده است. اگلوف [11] الگوریتم کاهش متناوب موازی را برای حل دستگاه معادلات سه‌قطری با اندازه بسیار بزرگ در معادلات دیفرانسیل با مشتقات جزئی به کار برد و به دلیل استفاده از حافظه سراسری، ۶۰ درصد کاهش سرعت را گزارش داد. کیم و همکاران [12] به دلیل محدودیت در حافظه مشترک، پیشنهاد دادند که برای دستگاه‌های بزرگ ابتدا آن‌ها را توسط الگوریتم کاهش متناوب موازی به دستگاه‌های کوچک‌تر تبدیل، سپس این دستگاه معادلات مجزا توسط الگوریتم توماس حل شوند. به کار بستن این ایده و

¹ Bank Conflict

² Shared Memory

³ Global Memory

اجرا روی پردازنده گرافیکی در مقایسه با حل گر ام کا ال^۱ که یک کتابخانه از توابع مفید در محاسبات علمی است، باعث افزایش ۸ تا ۴۹ برابری سرعت حل شد. اصفهانیان و همکاران [13] در سال ۲۰۱۳ با به کار بستن الگوریتم کاهش متناوب بهینه شده به حل و شبیه سازی جریان روی استوانه و ایرفویل در دو و سه بعد پرداختند و به افزایش سرعت حل ۱/۹ تا ۱۵/۲ برابر در دو بعد و ۶/۴ تا ۲۰/۳ برابر در سه بعد دست یافتند.

از دیگر تحقیقات در مورد استفاده پردازنده های گرافیکی در دینامیک سیالات محاسباتی می توان به موارد زیر اشاره کرد. بولنت توتکن و همکاران [14] بر روی شبیه سازی جریان توسط معادلات مرتبه بالا و حل آن بر روی پردازنده های گرافیکی بحث شده است. آن ها نشان داده اند که عملکرد مؤثر یک الگوریتم پردازش موازی برای شبیه سازی عددی مرتبه بالا از نظر سرعت حل، بستگی زیادی به سرعت حل گر دستگاه سه قطری دارد. در سال ۲۰۱۲ اصفهانیان و همکاران [15] با پیاده سازی روش های مرتبه بالا کارایی پردازنده گرافیکی را برای حل معادلات هذلولوی، بررسی کردند. محمودی و همکاران [16] نیز به بررسی و حل معادلات اویلر پرداختند، این ارزیابی توسط دو نوع پردازنده گرافیکی مختلف انجام شد و به حداکثر افزایش سرعت ۱۰۵ برابری نسبت به پردازنده مرکزی دست یافتند.

روش شبکه بولتزمن^۲ یک روشی است که نظر بسیاری از محققین را به خاطر ذات موازی اش به خود جلب کرده است و برای اجرا روی پردازنده گرافیکی بسیار مناسب است. تحقیقات بسیاری روی اجرای روش شبکه بولتزمن به صورت موازی انجام گرفته است. یک تحقیق در سال ۲۰۰۴ قبل از آن که کودا پا به عرصه بگذارد در مورد استفاده از پردازنده های گرافیکی به عنوان شتاب دهنده به

¹ Math Kernel Library (MKL)

² Lattice Boltzman Method

سرعت حل روش شبکه بولتزمن انجام شده [17] و به عنوان نمونه در تحقیقی دیگر رینالدی و همکاران [18] با اجرای روش شبکه بولتزمن روی پردازنده گرافیکی به افزایش سرعت ۱۰۰ برابر نسبت به پردازنده مرکزی در حل مسأله جریان درون حفره دست یافته‌اند. بنابراین روش شبکه بولتزمن هم یکی از مباحثی است که پتانسیل بالایی برای اجرای پردازش موازی دارد.

۳-۱ نوآوری و مروری بر کارهای انجام شده در این مطالعه

در این مطالعه با توجه به محدودیت الگوریتم‌ها و حل گرهای معادلات سه قطری موازی مانند کاهش متناوب و کاهش متناوب موازی در مورد اندازه دستگاه سه قطری (حتماً اندازه دستگاه باید توانی از دو باشد) و همچنین مشکل بودن استفاده از آن‌ها، یک رهیافت جدید برای حل این دسته از دستگاه معادلات با نام رهیافت توماس موازی ارائه شده است. این رهیافت چندین دستگاه سه قطری را با الگوریتم توماس به طور موازی و مستقل از هم حل می‌کند. برای مقایسه سرعت و دقت حل این الگوریتم‌ها مسأله جریان درون حفره، توسط روش تفاضل محدود، شبیه سازی شده است. معادلات حاکم گسسته شده و این معادلات به شکل سه قطری درآمده و سپس با الگوریتم توماس کلاسیک روی پردازنده مرکزی حل شده‌اند. پس از آن الگوریتم‌های کاهش متناوب، کاهش متناوب موازی و رهیافت توماس موازی برای حل استفاده شده‌اند. نتایج حاصل از حل مسأله جریان درون حفره توسط الگوریتم‌های موازی با نتایج مرجع [25] مقایسه شده و از صحت نتایج اطمینان حاصل شده است. در نهایت زمان حل تمام روش‌های حل، اندازه گیری و با هم مقایسه شده‌اند. حداکثر افزایش سرعتی که الگوریتم‌های کاهش متناوب، کاهش متناوب موازی و رهیافت توماس موازی اجرا شده در پردازنده گرافیکی در برابر الگوریتم توماس کلاسیک از خود نشان می‌دهند به ترتیب ۴/۴، ۵/۲ و ۳۸/۴۵ می‌باشد. همچنین در این تحقیق، اثر دسترسی هم‌مکان و غیر هم‌مکان به حافظه سراسری نیز مورد بحث و بررسی قرار گرفته و نشان داده شده است که با دسترسی

هم‌مکان، افزایش سرعت قابل توجهی (حدوداً دو برابری) برای پردازنده گرافیکی حاصل می‌شود.

در ادامه این مطالعه برای بررسی بیشتر و ارزیابی عملکرد رهیافت توماس موازی، جریان عبوری از استوانه توسط این رهیافت روی پردازنده گرافیکی و توماس کلاسیک روی پردازنده مرکزی، با روش *ADI* حل و با هم مقایسه شدند. در نهایت رهیافت توماس با افزایش سرعت حدود ۹ برابری در این مسأله عملکرد نسبتاً مناسبی داشت.

۴-۱ مروری اجمالی بر فصل‌های پایان‌نامه

در ادامه این مطالعه فصل‌های ۲ تا ۶ به شرح ذیل آورده شده‌اند. در فصل دوم مبانی پردازش موازی معرفی شده‌اند. مفاهیمی چون معماری پردازنده‌های موازی و سیستم‌های حافظه معرفی شده‌اند. در فصل سوم معماری پردازنده‌های گرافیکی و به‌طور خاص معماری پردازنده‌های شرکت انویدیا معرفی شده و مورد بحث قرار گرفته‌اند. در مورد نسل‌ها و مدل‌های مختلف پردازنده‌های گرافیکی صحبت شده و مقدماتی از آنچه که باید در برنامه‌نویسی موازی در نظر گرفته شود از نظر گذرانده شده است.

در فصل چهارم به معرفی دو الگوریتم حل دستگاه معادلات سه‌قطری به نام‌های کاهش متناوب و کاهش متناوب موازی پرداخته شده و همچنین به معرفی رهیافت توماس موازی به‌عنوان ایده اصلی این مطالعه پرداخته شده است. در ادامه برای ارزیابی این الگوریتم‌های معرفی شده هندسه جریان درون حفره انتخاب و شبیه‌سازی و نتایج ارائه شده است. در فصل پنجم نیز جریان گذرای عبوری از استوانه با اعمال روش *ADI* مورد بررسی قرار گرفته است که دستگاه معادلات سه‌قطری منتج از این روش با استفاده از رهیافت توماس موازی اجرا شده روی پردازنده گرافیکی و الگوریتم توماس کلاسیک اجرا شده روی پردازنده مرکزی، حل شده‌اند.

در نهایت در فصل ششم نیز نتایج کلی حاصل از این مطالعه و جمع بندی کلی آورده شده است و پیشنهاداتی برای پژوهش های آتی در این زمینه ارائه شده است.

فصل دوم:

مقدمه‌ای بر پردازش

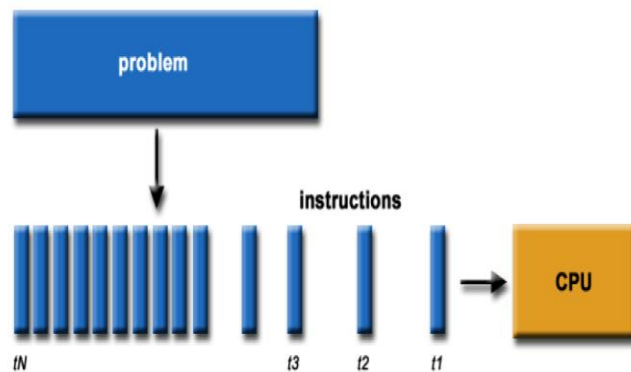
موازی

۲-۱ مقدمه

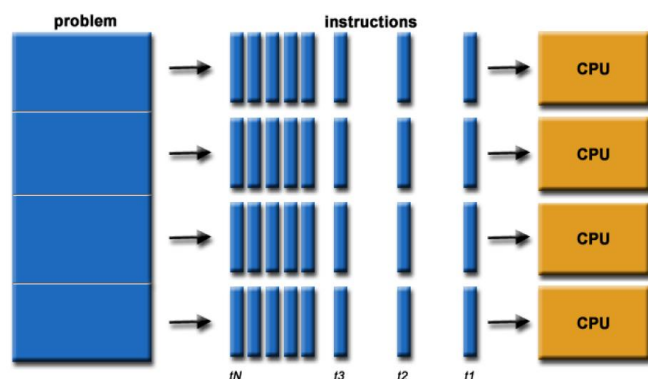
در این فصل مروری اجمالی بر مفهوم پردازش موازی، اهمیت و کاربرد آن و همچنین مروری بر مفاهیم و ملزومات برنامه‌نویسی خواهد بود.

۲-۲ محاسبه موازی

محاسبه موازی اجرای همزمان قسمت‌های مختلف یک برنامه در چندپردازنده به‌منظور حصول سریع‌تر نتایج است. در پردازش ترتیبی، دستورات به ترتیب در پردازنده اجرا می‌شوند و سرعت اجرا متناسب با سرعت پردازنده است (شکل ۱-۲). در پردازش موازی دستورات در چندپردازنده اجرا می‌شوند ولی سرعت اجرا الزاماً برابر با تعداد پردازنده‌ها در سرعت یک پردازنده نیست (شکل ۲-۲) [19]. محاسبه موازی در بخش‌های مختلف کامپیوتر اعم از سخت‌افزار و نرم‌افزار شکل می‌گیرد، بنابراین برای بررسی کلیات محاسبه موازی باید به جنبه‌های مختلف سخت‌افزاری و نرم‌افزاری آن پرداخت.



شکل ۲-۱: شکلی شماتیک از محاسبات سریال



شکل ۲-۲: شکلی شماتیک از محاسبات موازی

از پردازش موازی در جهت افزایش قدرت کامپیوترها استفاده می‌شود. اما اصلی‌ترین استفاده آن در حل مسائل علمی و مهندسی است. (شکل ۲-۳). از جمله این حوزه‌ها می‌توان به موارد زیر اشاره کرد:

- فیزیک کاربردی، هسته‌ای، ذرات بنیادی، ماده چگال، گداخت هسته‌ای، فوتونیک و نانو
- اتمسفر، زمین و محیط‌زیست
- فناوری زیستی و ژنتیک
- مهندسی مکانیک؛ از اندام‌های مصنوعی تا سازه‌های فضایی
- مهندسی الکترونیک؛ طراحی مدار، میکروالکترونیک
- علوم کامپیوتر و ریاضی

اما امروزه نه تنها حل مسائل علمی احتیاج به پردازش موازی دارد بلکه برخی نرم‌افزارهای تجاری نیز به کامپیوترهای سریع نیاز دارند. بسیاری از این برنامه‌ها احتیاج به پردازش حجم زیادی از داده‌ها به شکل بسیار پیچیده دارند. از جمله این برنامه‌ها می‌توان به موارد زیر اشاره کرد:

- پایگاه عظیم داده‌ها و عملیات داده‌کاوی^۱

- موتورهای جستجو و تشخیص پزشکی

- طراحی و شبیه‌سازی دارو

- مدل‌سازی اقتصادی

- گرافیک پیشرفته و فناوری واقعیت مجازی^۲

- فناوری چندرسانه‌ای و شبکه ویدیویی

از دلایل اصلی استفاده از محاسبات موازی عبارت‌اند از:

- صرفه‌جویی در زمان و هزینه: استفاده از منابع بیشتر در یک کار، زمان انجام کار را

کاهش می‌دهد. علاوه بر این استفاده از چندین منبع ارزان به‌جای استفاده از نوع

گران قیمت آن‌ها سبب کاهش هزینه‌ها می‌شود.

- حل مسائل بزرگ‌تر: بسیاری از مسائل بزرگ و پیچیده که حل آن‌ها توسط یک

کامپیوتر، به خصوص در کامپیوتر با حافظه محدود، غیر عملی و ناممکن است، با

استفاده از کامپیوترهای موازی شده قابل حل هستند. برای مثال در موتورهای

جستجوگر وب و پایگاه‌های داده لازم است تا میلیون‌ها تراکنش در ثانیه پردازش

شوند.

- فراهم نمودن همزمانی: در منابع منفرد، در یک زمان تنها یک عمل را می‌توان انجام

داد، در حالی که در منابع محاسباتی چندگانه، چندین کار به‌طور همزمان قابل انجام

هستند.

¹ Data Mining

² Virtual Reality Technology

۲-۳ معماری کامپیوتر موازی

پرونده‌دارترین طبقه‌بندی معماری سامانه‌های کامپیوتری، در سال ۱۹۶۶ توسط فلین^۱ تعریف و ارائه شد. طرح طبقه‌بندی فلین^۲ بر اساس جریان اطلاعات صورت گرفته است. اطلاعاتی که پردازنده با آن سروکار دارد را می‌توان به دودسته داده و دستورالعمل تقسیم کرد. طبقه‌بندی فلین جریان دستورالعمل یا داده می‌تواند به یکی از صورت‌های تکی یا چندتایی باشد. بر اساس معماری سامانه‌های کامپیوتری را می‌توان به چهار دسته تقسیم نمود:

۱. جریان یک دستورالعمل، یک داده (SISD): این معماری برای یک کامپیوتر ترتیبی یا سریال است. در این روش تنها یک جریان دستوری توسط پردازنده در طول هر پالس ساعت مورد عمل واقع می‌شود و همین‌طور یک جریان داده در طول پالس ساعت مورد استفاده قرار می‌گیرد. در این روش دستورها به‌طور قطعی انجام می‌شوند و وابسته به عمل پردازنده دیگر نیستند. کامپیوترهای ترتیبی متعارف ون نیومن در این دسته قرار می‌گیرند. این روش مورد استفاده در بیشتر کامپیوترها، از سامانه‌های قدیمی تا کامپیوترهای شخصی امروزی است.

۲. جریان یک دستورالعمل چند داده (SIMD): این معماری، یکی از انواع کامپیوترهای موازی است. آرایه پردازنده‌ها^۵ در این دسته قرار می‌گیرند. ماشین‌های SIMD یک واحد کنترل^۶ دارند که مثل ماشین فن نیومن عمل می‌کند (یعنی دستور اجرا می‌کند) ولی بیش از یک عنصر پردازشگر دارند. واحد کنترل، سیگنال‌های کنترل را برای تمام پردازنده‌ها تولید می‌کند و این

^۱ Flynn

^۲ Flynn's Taxonomy

^۳ Single Instruction-Single Data

^۴ Single Instruction-Multi Data

^۵ Array Processors

^۶ Control Unit

عناصر پردازنده، یک عمل مشابه را در طول هر پالس ساعت، روی داده‌های متفاوت انجام می‌دهند. این روش برای حل مسائل خاص که از داده‌هایی با الگوی ثابت پیروی می‌کنند مناسب است. از جمله این مسائل می‌توان به پردازش تصویر اشاره کرد.

۳. جریان چند دستورالعمل، یک داده (MISD)^۱ : در این طرح موازی، یک جریان داده به واحد پردازش داده، ارسال می‌شود. هر واحد پردازش به‌طور مستقل با جریان‌های دستور مستقل روی داده‌ها عمل می‌کند. تاکنون تعداد کمی از نمونه‌های واقعی این کلاس از کامپیوترهای موازی وجود دارند. به‌عنوان نمونه، می‌توان به کامپیوتر آزمایشی کارنگی ملون^۲ اشاره کرد. از جمله کاربردهایی که برای این روش می‌توان مثال زد، اعمال چند فیلتر فرکانسی روی یک جریان سیگنال و دیگری، اعمال چند الگوریتم رمزگذاری در باز کردن یک پیغام کد شده است.

۴. جریان چند دستورالعمل، چند داده (MIMD)^۳ : به این دسته چندپردازنده نیز می‌گویند. در این معماری هر پردازنده امکان اجرای چند جریان دستوری جداگانه را دارد و این دستورات روی چند جریان داده مختلف اعمال می‌شود. سوپر کامپیوترهای امروزی، کامپیوترهای موازی خوشه‌ای^۴، کامپیوترهای چندپردازنده متقارن^۵ و کامپیوترهای چند هسته‌ای^۶ امروزی از این معماری استفاده می‌کنند. البته بیشتر کامپیوترهای با معماری MIMD از زیر مولفه‌های اجرایی SIMD بهره می‌برند.

یک ماشین MIMD (دسته چهارم)، دارای پردازنده‌هایی با واحد کنترل اختصاصی بوده و قادر

^۱ Multi Instruction-Single Data

^۲ Carnegie-Mellon

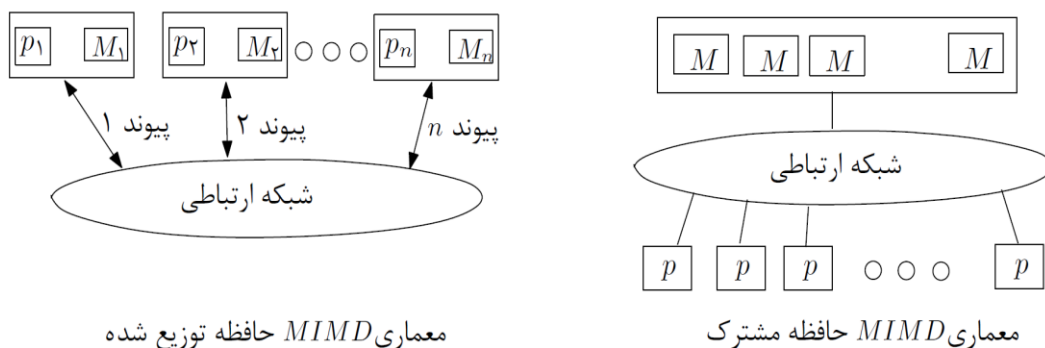
^۳ Multi Instruction-Multi Data

^۴ Cluster Parallel Computers

^۵ Symmetric Multiprocessor(SMP)

^۶ Multi Core Computers

است دستورالعمل‌های مختلف را بر روی داده‌های مختلف به صورت همزمان اجرا نماید. این روش معمول‌ترین طرح کامپیوتر موازی است و کامپیوترهای مدرن به سمت این معماری حرکت می‌کنند. این نوع معماری‌ها از چندین پردازنده و چندین نوع حافظه که از طریق شبکه‌های ارتباطی به هم متصل شده‌اند، ساخته می‌شوند و به دو گروه اصلی تقسیم می‌شوند: حافظه مشترک و حافظه توزیع شده (انتقال پیام). شکل ۲-۴ ساختار کلی این دو گروه را نشان می‌دهد. در این شکل، p نشان‌دهنده پردازنده و M نشان‌دهنده حافظه است [1].



شکل ۲-۳: انواع معماری مدل MIMD

۲-۳-۱ سیستم حافظه مشترک

در سامانه‌های حافظه مشترک^۱ تمام پردازنده‌ها یک حافظه سراسری مشترک دارند. ارتباط میان عملیات در حال اجرا روی پردازنده‌های مختلف با نوشتن و خواندن از حافظه سراسری برقرار می‌شود. همچنین هماهنگ‌سازی و همزمانی تمام پردازنده‌های میانی از طریق این حافظه مشترک انجام می‌شود. چنانچه تمامی پردازنده‌ها زمان دسترسی یکسانی به هر مکان حافظه داشته باشند، سیستم حافظه مشترک را سیستم چندپردازنده متقارن می‌نامند. در طراحی سامانه‌های حافظه

¹ Shared Memory

مشترک برخی مسائل اساسی نظیر کنترل دسترسی و وابستگی داده‌ها، همزمانی، حفاظت و امنیت وجود دارد که باید مورد توجه قرار گیرد [19].

۲-۳-۲ سیستم حافظه توزیع شده

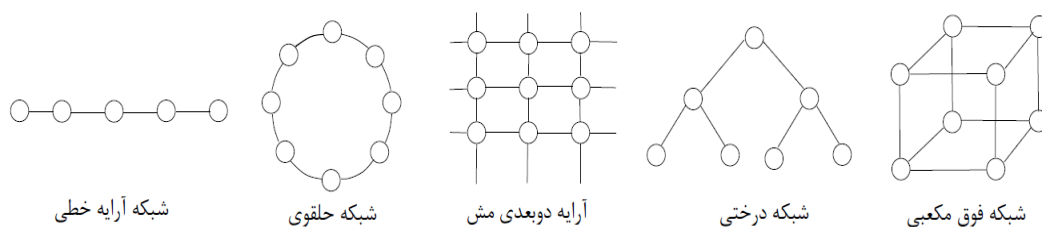
سامانه‌های مبتنی بر حافظه توزیع شده^۱ (که سیستم انتقال پیام هم نامیده می‌شود)، گروهی از چندپردازنده‌ها هستند که در آن هر پردازنده به حافظه محلی خود دسترسی دارد. برخلاف سیستم‌های حافظه مشترک، ارتباطات در این سیستم از طریق دستورات و دریافت پیام صورت می‌گیرد که باید توسط برنامه نویس در نرم‌افزار کاربردی نوشته شود. یک گره در چنین سیستمی شامل یک پردازنده و حافظه محلی آن می‌باشد. هر گره معمولاً قابلیت ذخیره‌سازی پیام‌ها در میان‌گیر (مکان‌های موقتی از حافظه که پیام‌ها در آنجا منتظر می‌مانند تا ارسال یا دریافت شوند) را دارد و عملیات ارسال/دریافت را همزمان با پردازش انجام دهند. پردازش پیام و انجام محاسبات به صورت همزمان توسط سیستم عامل موجود صورت می‌گیرد. سیستم‌های با حافظه توزیع شده از قابلیت توسعه بسیار بالایی برخوردار هستند و واحدهای پردازشگر آن‌ها را با روش‌های متنوع، از ساختارهای ارتباطی با معماری خاص تا شبکه‌های پراکنده جغرافیایی می‌توان به یکدیگر مرتبط نمود. قابلیت توسعه بدان معناست که تعداد پردازنده‌ها را بدون کاهش قابل توجه کارایی، بتوان افزایش داد.

۲-۴ شبکه‌های ارتباط داخلی

شبکه‌های ارتباط سامانه‌های چندپردازنده را می‌توان بر اساس معیارهای مختلفی طبقه‌بندی نمود که یکی از این معیارها توپولوژی شبکه است. توپولوژی مشخص می‌کند که چگونه پردازنده‌ها

¹ Distributed Memory

و حافظه‌ها به پردازنده‌ها و حافظه‌های دیگر موجود در سیستم متصل می‌شوند. به‌صورت کلی توپولوژی شبکه را می‌توان به دو گروه ایستا و پویا تقسیم کرد. در شبکه‌های ایستا تمامی ارتباطات در زمان طراحی بدون توجه به این که مورد نیاز هستند یا نه ساخته می‌شوند. در این شبکه‌ها پیام‌ها باید از پیوندهای معینی عبور کنند. شبکه‌های ارتباطی پویا در صورت نیاز اتصالاتی بین دو یا چند گره می‌سازند تا پیام‌ها از این پیوندها عبور کنند. توپولوژی خطی، حلقوی، مش^۱، درختی و فوق مکعبی^۲ که طرح آن‌ها در شکل ۲-۵ مشخص است، نمونه‌هایی از شبکه‌های ارتباطی ایستا هستند و شبکه‌های تک‌باس^۳، چند باس^۴، کراس بار^۵ و چند لایه‌ای^۶ نمونه‌هایی از شبکه‌های ارتباطی پویا هستند [20].



شکل ۲-۴: انواع شبکه‌های ارتباطی ایستا

۲-۴-۱ مدل برنامه نویسی موازی

مدل‌های تجریدی به علت طبیعت آرمانگرایی آن ممکن است در دنیای واقعی مناسب به‌نظر نرسد. اما ماشین‌های تجریدی در مطالعه الگوریتم‌های موازی توزیع شده و ارزیابی کارایی پیش‌بینی شده، مستقل از ماشین‌های موازی بسیار مفیدند.

¹ Mesh
² Hyper Cube
³ Single Bus
⁴ Multiple Bus
⁵ Crossbar
⁶ Multistage

بدیهی است که اگر اجرای یک الگوریتم روی یک سیستم تجریدی رضایت‌بخش نباشد، پیاده‌سازی آن روی سیستم واقعی بی‌معنی است. با وجود این که مدل‌های تجریدی برخی ملاحظات عملی در سامانه‌های واقعی موازی توزیع‌شده را در نظر نمی‌گیرند، دشواری یافتن محدودیت‌های اجرایی و برآوردهای پیچیدگی را کاهش می‌دهند. بنابراین الگوریتم‌های موازی همواره طبق یک مدل انتخابی طراحی شده و سپس تلاش می‌شود تا مدل به یک برنامه قابل اجرا تبدیل گردد. برای اجرایی کردن یک مدل در دنیای واقعی مجموعه‌ای از زبان‌ها، کامپایلرها، کتابخانه‌ها، سامانه‌های ارتباطی وردی و خروجی موازی به کار گرفته می‌شود. تاکنون مدل‌های موازی مختلفی نظیر حافظه مشترک، داده-موازی، رشته‌ها، انتقال پیام و غیره ارائه شده است که در ادامه دو نوع رایج آن شرح داده می‌شود [21].

۲-۴-۲ مدل حافظه مشترک

در مدل حافظه مشترک، یک برنامه موازی به قسمت‌های مختلفی تقسیم شده‌اند که به هر قسمت وظیفه^۱ گفته می‌شود. اجرای هر وظیفه به یک پردازنده محول می‌شود و تمام پردازنده‌ها بر روی داده‌های ذخیره شده در حافظه مشترک عمل می‌کنند. از مکانیزم‌های همزمانی مختلفی نظیر قفل‌ها (روش همزمانی هستند که سیستم عامل به کار می‌گیرد تا مانع استفاده همزمان چندین وظیفه از یک منبع گردد) برای کنترل دستی همزمان پردازنده‌ها به خانه‌های حافظه استفاده می‌شود. در تحلیل الگوریتم‌های موازی نوشته‌شده در این مدل، معیارهایی نظیر زمان اجرا، تعداد پردازنده‌هایی که الگوریتم برای حل مسأله به کار می‌برد و هزینه الگوریتم موازی که حاصل ضرب زمان اجرا در تعداد پردازنده‌ها است مورد توجه قرار می‌گیرد.

¹ Task

یکی از مدل‌هایی که در سیستم حافظه مشترک به‌طور گسترده مورد استفاده قرار می‌گیرد، مدل ماشین دسترسی تصادفی موازی (PRAM) است. یک PRAM متشکل از یک واحد کنترل و یک حافظه سراسری است که برای p پردازنده مشترک است. علاوه بر حافظه سراسری که ارتباط پردازنده‌ها از طریق آن برقرار می‌شود، هر پردازنده به منظور کاهش مراجعات به حافظه مشترک و تسریع پردازش، حافظه اختصاصی خود را دارد. شکل ۲-۶ دیاگرامی را نشان می‌دهد که اجزای این مدل را شرح می‌دهد. در این مدل فرض می‌شود که به تعداد دلخواه پردازنده در اختیار هست و هیچ دو پردازنده‌ای مستقیماً به یکدیگر متصل نیستند و ارتباطات تنها از طریق خواندن و نوشتن در حافظه مشترک انجام می‌شود. حالت‌های مختلفی برای عملیات خواندن و نوشتن در حافظه مشترک وجود دارد که بر این اساس، می‌توان PRAM را به کلاس‌های زیر تقسیم نمود:

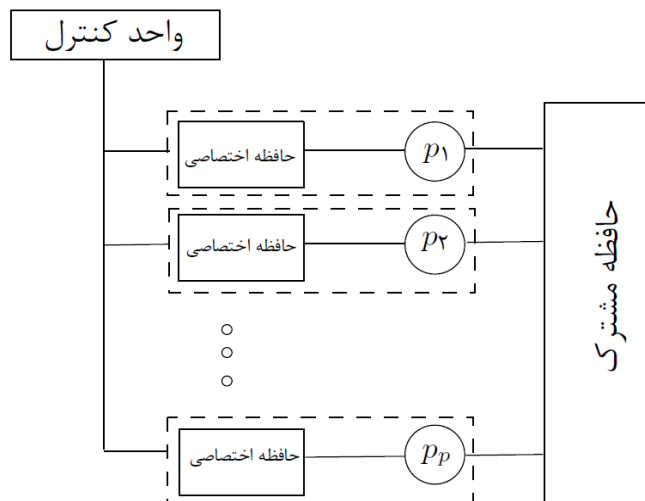
۱. خواندن انحصاری، نوشتن انحصاری (EREW)^۱: دسترسی‌های خواندن و نوشتن در یک مکان حافظه، انحصاری است.
۲. خواندن انحصاری، نوشتن همزمان (ERCW)^۲: چندپردازنده اجازه نوشتن همزمان را دارند ولی خواندن از حافظه انحصاری است.
۳. خواندن همزمان، نوشتن انحصاری (CREW)^۳: اجازه دسترسی‌های همزمان خواندن داده می‌شود ولی دسترسی نوشتن انحصاری است.
۴. خواندن همزمان، نوشتن انحصاری (CRCW)^۴: دسترسی خواندن و نوشتن همزمان مجاز است.

^۱ Exclusive-Read Exclusive-Write

^۲ Exclusive-Read Concurrent-Write

^۳ Concurrent-Read Exclusive-Write

^۴ Concurrent-Read Concurrent-Write



شکل ۲-۵: مدل PRAM برای محاسبات موازی

۲-۴-۳ مدل انتقال پیام

مدل انتقال پیام شامل مجموعه‌ای از پردازنده‌ها است که هر یک تنها حافظه محلی مختص به خود را دارند و پردازنده‌ها با ارسال و دریافت پیام با یکدیگر ارتباط برقرار می‌کنند. انتقال داده بین پردازنده‌ها نیازمند عملیات متقابل بین پردازنده‌ها است به این معنی که هر عمل ارسال متناظر با یک عمل دریافت داده است. این مدل به دلیل مزیت گوناگونی که دارد به‌طور گسترده در زمینه محاسبات موازی مورد استفاده قرار می‌گیرد. برخی از این مزیت‌ها عبارت‌اند از:

- سازگاری با سخت‌افزار: این مدل برای استفاده در ابرکامپیوترها و کلاسترها که شامل پردازنده‌های مجزایی هستند که توسط شبکه‌های ارتباطی به یکدیگر متصل‌اند، مناسب است.
- ساختار تابعی: مدل انتقال پیام یک مجموعه کاملی از توابع را ارائه می‌دهد که امکان کنترل‌هایی را فراهم می‌کند که در مدل‌های دیگر نظیر مدل داده موازی وجود ندارد.

- کارایی: استفاده مؤثر از پردازنده‌های مدرن امروزی، نیاز به مدیریت قوی بر سلسله مراتب حافظه، به ویژه حافظه‌های کش دارد. این مدل با اعطای ابزار صریح کنترل به برنامه‌نویس، امکان مدیریت داده‌ها را فراهم می‌آورد.

مشکل اصلی این روش این است که مسئولیت آنچه اتفاق می‌افتد بر عهده برنامه‌نویس است. یعنی برنامه‌نویس باید صریحاً با فراخوانی توابع موجود، داده‌ها را بین پردازنده‌ها توزیع کرده و همزمانی و ارتباط بین پردازنده‌ها را مدیریت کند.

مدل‌های مختلفی از انتقال پیام نظیر MPI, BSP, CGM¹ و غیره ارائه شده است که هر یک کتابخانه مربوط به خود را دارند.

۲-۵ الگوریتم‌های موازی

بسیاری از الگوریتم‌های به‌منظور استفاده از سخت‌افزار موازی باید دوباره طراحی شوند. برنامه‌هایی که در یک سیستم تک پردازنده درست کار می‌کند ممکن است در یک محیط موازی هرگز اجرا نشود و خروجی غیر قابل قبولی در بر داشته باشد. این بدان علت است که چند کپی از یک برنامه ممکن است با هم تداخل کنند. (برای نمونه تداخل در اختصاص دهی یک مقدار به یک پارامتر در یک محل از حافظه). بنابراین نیاز اصلی یک سیستم موازی، برنامه‌نویسی دقیق مختص به خود است.

طراحی و توسعه برنامه‌های موازی اغلب یک فرآیند دستی محسوب می‌شوند. برنامه‌نویس مسئول تعیین و اجرای واقعی موازی‌سازی است. توسعه دستی کدهای موازی اغلب فرآیندی زمان‌بر، پیچیده، تکراری و مستعد ایجاد خطا است. در سال‌های اخیر بسیاری از سامانه‌های

¹ Message Passing Interface

نرم‌افزاری جهت برنامه‌نویسی کامپیوترهای موازی طراحی شده‌اند که به برنامه‌نویس در تبدیل برنامه سریال به موازی کمک می‌کند. این دستگاه‌ها هم در سطح سیستم‌عامل و هم در سطح زبان‌های برنامه‌نویسی وجود دارند. آن‌ها باید ساز و کاری جهت تقسیم یک مسأله به چند وظیفه و تخصیص این وظایف به پردازنده داشته باشند. چنین سازوکارهایی می‌تواند شامل موازی‌سازی ضمنی و یا موازی‌سازی صریح باشد.

در روش موازی‌سازی ضمنی، سیستم (می‌تواند کامپایلر یا برنامه دیگری باشد) به‌طور خودکار مسأله را به چند وظیفه تقسیم کرده و هر یک را به پردازنده‌ای اختصاص می‌دهد اما در روش موازی‌سازی صریح، برنامه‌نویس شخصاً مسأله را به چندوظیفه تقسیم کرده و هر یک را به پردازنده‌ای ارجاع می‌دهد. روش موازی‌سازی ضمنی محدود به زیر مجموعه‌ای از کدها (اغلب حلقه‌های تکرار) است و انعطاف‌پذیری کمتری نسبت به موازی‌سازی صریح دارد. همچنین ممکن است نتایج اشتباهی تولید کند و کارایی را کاهش دهد. از این رو اغلب برنامه‌نویسی‌های موازی به‌صورت صریح انجام می‌شود.

۲-۵-۱ طراحی الگوریتم موازی

اولین گام در طراحی این الگوریتم‌های موازی این است که برنامه‌نویس یاد بگیرد چگونه موازی فکر کند. قسمت‌هایی از مسأله که قابلیت موازی‌سازی دارند را تشخیص داده و پس از انتخاب مدل، بدون توجه به ملزومات اجرایی، ذهن خود را صرفاً بر روی ارائه بهترین الگوریتم موازی متمرکز کند.

در حل یک مسأله به روش موازی به نکات مختلفی باید توجه نمود. قبل از هر چیز باید مشخص شود که آیا مسأله قابلیت موازی‌سازی دارد؟ به‌عنوان مثال مسأله ساخت دنبال فیبوناتچی

به علت ماهیت وابستگی داده‌ها، یک مسأله غیر موازی است. پس از این مرحله باید نقاط اصلی مسأله که بیشترین محاسبات در آن قسمت انجام می‌شود، شناسایی شده و حتی‌المقدور بر روی موازی‌سازی این نقاط اصلی^۱ برنامه تمرکز شود. همچنین تنگناهای^۲ مسأله باید مشخص شود یعنی نقاطی که به علت وابستگی داده‌ها و یا نیاز به عملیات ورودی - خروجی، عمل موازی‌سازی متوقف شود.

پس از این نوبت به تجزیه مسأله به بخش‌های مختلف است که هر بخش می‌تواند تحت عنوان یک وظیفه به یک پردازنده محول شود. دو روش اساسی برای تقسیم وظایف محاسباتی بین پردازنده‌ها وجود دارد: تجزیه دامنه‌ای و تجزیه تابعی. در تجزیه دامنه‌ای داده‌های مسأله تقسیم شده و هر پردازنده دستورات یکسانی را بر روی داده‌های مربوط به خود اجرا می‌کند. در تجزیه تابعی دستورات محاسباتی که باید اجرا شوند بین پردازنده‌ها تقسیم می‌شوند. پس از تجزیه مسأله به وظایف مختلف، چنانچه نیاز به ارتباط بین وظایف باشد، باید از رویه‌های همزمانی و عملیات ارتباطی بین پردازنده‌ها استفاده گردد.

۲-۶ معیار ارزیابی سیستم موازی

یکی از معیارهای ارزیابی سیستم موازی ضریب تسريع^۳ یا $S(p)$ است که به صورت زیر تعریف می‌شود: نسبت زمان مورد نیاز برای حل یک مسأله توسط یک پردازنده که با $T_{sequential}$ نمایش داده می‌شود، به زمان مورد نیاز برای حل همان مسأله توسط یک سیستم موازی که از p پردازنده موازی تشکیل شده است. زمان سیستم موازی با $T_{parallel}$ نمایش داده می‌شود.

¹ Hotspot

² Bottlenecks

³ Speed Up

$$S(p) = \frac{T_{sequential}}{T_{parallel}} \quad (1-2)$$

چنانچه $T_{parallel} = O(\frac{T_{sequential}}{p})$ باشد، ضریب تسریع برابر p است و سیستم موازی بهینه است. در عمل، دستیابی به افزایش خطی سرعت (افزایش متناسب با تعداد پردازنده‌ها) بسیار مشکل است. این مشکل از طبیعت ترتیبی بسیاری از الگوریتم‌ها است به طوری که قسمت‌هایی از یک الگوریتم قابل موازی‌سازی و قسمت‌هایی غیر قابل موازی‌سازی هستند. بر طبق قانون آمدال^۱، میزان تسریع یک الگوریتم موازی نسبت به یک الگوریتم ترتیبی به تعداد پردازنده مورد استفاده بستگی ندارد بلکه به قسمتی از الگوریتم که قابل موازی‌سازی نیست وابسته است. اگر F برابر کسری از کل الگوریتم باشد که قابل موازی‌سازی نیست، و حتماً باید به صورت ترتیبی اجرا شود، ضریب تسریع، طبق قانون آمدال به صورت زیر تعریف می‌شود:

$$S(p) = \frac{1}{F + \frac{1-F}{p}} \quad (2-2)$$

فرض کنید ده درصد یک الگوریتم قابلیت موازی‌سازی ندارد یعنی $F = 10\%$ اما بقیه الگوریتم به طور موازی توسط ۲۰ پردازنده اجرا می‌شود. در این حالت سرعت اجرای برنامه (نسبت به زمانی که روی یک پردازنده اجرا می‌شود) ۲۰ برابر نمی‌شود بلکه مطابق قانون آمدال ضریب تسریع تقریباً ۷ برابر می‌شود.

$$S(20) = \frac{1}{0.1 + \frac{1-0.1}{20}} \simeq 7.0$$

معیار دیگری که می‌تواند در ارزیابی سیستم‌های موازی به کار گرفته شود، کارایی^۲ یا $E(p)$ است که برابر با نسبت هزینه اجرای یک الگوریتم در سیستم ترتیبی به هزینه اجرای همان

¹ Amdhal's law

² Efficiency

الگوریتم در یک سیستم موازی است. هزینه اجرای الگوریتم برابر حاصل ضرب زمان اجرا در تعداد پردازنده‌های مورد استفاده است.

$$E(p) = \frac{1 \times T_{\text{sequential}}}{p \times T_{\text{parallel}}} = \frac{S(p)}{p} \quad (3-2)$$

۷-۲ یک نمونه الگوریتم موازی

در این بخش برای مسأله ساده محاسبه جمع n عدد، الگوریتمی موازی در مدل حافظه مشترک ارائه می‌شود. این الگوریتم‌ها لزوماً بهینه نیستند و صرفاً برای نمایش تعامل پردازنده‌ها در مدل بیان شده‌اند.

الگوریتم محاسبه حاصل جمع آرایه‌ای از اعداد در مدل EREW PRAM

فرض کنید آرایه $A[1 \dots n]$ در حافظه مشترک ذخیره شده است. هدف محاسبه حاصل جمع این اعداد به کمک $\frac{n}{2}$ پردازنده است. از آنجایی که نیازی به انجام چند عملیات هم‌زمان خواندن و نوشتن روی یک مکان حافظه نیست، حالت EREW انتخاب می‌شود. جمع کل در نهایت در مکان $A[n]$ ذخیره می‌شود. برای سادگی فرض می‌شود n توان صحیحی از ۲ در نظر گرفته می‌شود. الگوریتم در $\log n$ تکرار به شرح زیر عمل می‌کند. در تکرار اول تمامی پردازنده‌ها فعال هستند. در تکرار دوم تنها نیمی از پردازنده‌ها فعال خواهند بود و بقیه تکرارها هم به همین ترتیب خواهد بود. جزئیات در الگوریتم ۱ تشریح شده‌اند.

ورودی: آرایه $A[1 \dots n]$ که در حافظه مشترک ذخیره شده است.

خروجی: جمع عناصر آرایه A

برای مقادیر $i = 1$ تا $i = \log n$ محاسبات زیر را تکرار کن:

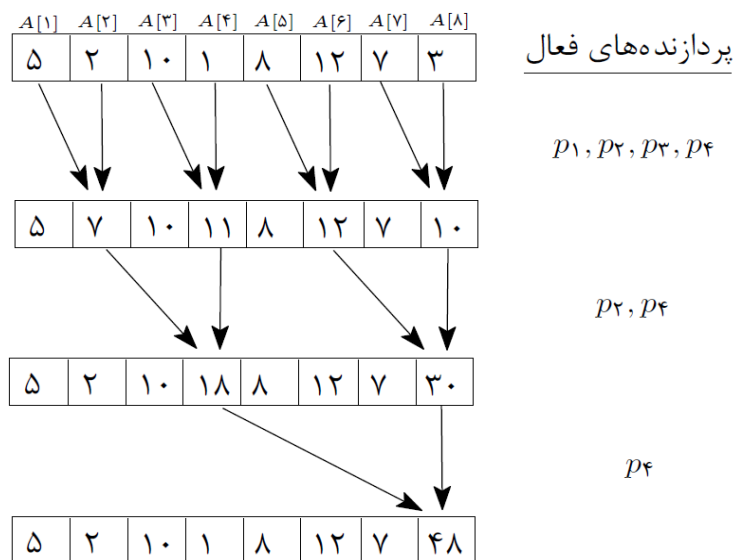
۱. برای همه پردازنده‌ها p_j که $1 \leq j \leq \frac{n}{2}$ است به صورت موازی بررسی کن:

اگر شرط $2j \bmod 2^i = 0$ برقرار است، آنگاه جایگذاری زیر را انجام بده:

$$A[2j - 2^{i-1}] + A[2j] \rightarrow A[2j]$$

شکل ۷-۲ نحوه اجرای الگوریتم را روی یک آرایه با اندازه ۸ نشان می‌دهد. برای جمع این ۸ عنصر سه تکرار به شرح زیر مورد نیاز است. در اولین تکرار، پردازنده‌های p_1, p_2, p_3, p_4 به ترتیب مقادیر ذخیره شده در مکان‌های ۱، ۳، ۵ و ۷ را با اعداد ذخیره شده در کان‌های ۲، ۴، ۶ و ۸ جمع می‌کنند. در تکرار دوم پردازنده‌های p_2, p_4 به ترتیب مقادیر ذخیره شده در مکان‌های ۲ و ۶ را با اعداد ذخیره شده در مکان‌های ۴ و ۸ جمع می‌کنند. در نهایت، در تکرار سوم پردازنده p_4 مقدار ذخیره شده در مکان ۴ را با مقدار مکان ۸ جمع می‌کند و مکان ۸ مقدار جمع تمام آرایه‌ها خواهد بود.

تحلیل پیچیدگی الگوریتم: دستور شماره الگوریتم به تعداد $\log n$ بار اجرا می‌شود و هر تکرار زمان پیچیدگی ثابتی دارد. از این رو زمان اجرای این الگوریتم $\log n$ می‌باشد. معیارهای پیچیدگی الگوریتم به صورت زیر خلاصه می‌شود:



شکل ۶-۲: مثالی از الگوریتم Sum_EREW با ۸ ورودی

زمان اجرا: $T(n) = O(\log n)$ تعداد پردازنده‌ها: $P = \frac{n}{2}$

هزینه اجرا: $C(n) = O(n \log n)$ تسریع: $S(n) = n / \log n$

کارایی: $E(n) = 1 / (\log n)$

که الگوریتم ترتیبی مناسب می‌تواند حاصل جمع لیستی از n عنصر را در $O(n)$ محاسبه کند، بنابراین الگوریتم موازی ارائه شده زمان اجرا را کاهش می‌دهد ولی دارای هزینه بهینه نیست و تنها برای زمانی که تعداد پردازنده‌های موجود از $O(n)$ باشد مناسب است. به چنین سیستم‌های موازی که تعداد پردازنده‌های موجود $O(n)$ باشد، مناسب است. به چنین سیستم‌های موازی که تعداد پردازنده‌ها تقریباً از مرتبه ورودی است و هر پردازنده به طور میانگین، تقریباً $O(1)$ داده پردازش می‌کند، به اصطلاح پردازش دانه‌ریز^۱ گفته می‌شود.

^۱ Fine Grained

فصل سوم:

پردازنده گرافیکی

۳-۱ مقدمه

در این فصل به طور مختصر توضیحاتی در مورد معماری و روش برنامه‌نویسی پردازنده‌های گرافیکی داده می‌شود. پردازنده‌های گرافیکی مانند پردازنده‌های مرکزی برای انجام محاسبات ایجاد شدند اما بر خلاف پردازنده مرکزی که برای انجام محاسبات عمومی سیستم در نظر گرفته شده، پردازنده گرافیکی به طور اختصاصی برای پردازش و ایجاد تصاویر خلق شده است و این امر شامل پردازش حجم عظیمی از اطلاعات در زمان کوتاهی می‌شود که این دستگاه دقیقاً متناسب با همین نیاز طراحی شده است. این نوع معماری و طراحی می‌تواند به محاسبات مهندسی سرعت دهد و همین موضوع باعث شده تا پردازنده گرافیکی در حیطه بررسی این پایان نامه قرار گیرد. برای بدست آوردن یک افزایش سرعت مناسب در حل مسائل عددی روی پردازنده گرافیکی نیاز هست تا با معماری این پردازنده‌ها آشنا بود تا بتوان بهترین و مناسب‌ترین روش برنامه نویسی را پیاده‌سازی کرد. آشنایی با انواع حافظه، برنامه نویسی در کودا، نخ‌ها، بلوک‌ها و غیره از اهداف این فصل هستند.

۳-۲ معماری پردازنده گرافیکی

در انتهای سال ۲۰۰۶ شرکت انویدیا یک معماری جدیدی از پردازنده‌های گرافیکی ارائه کرد (تراشه G80). هم‌زمان با این معماری جدید میان‌افزار کودا هم پدید آمد و معرفی شد و یک چهارچوب^۱ طراحی شد تا بتوان مستقیماً روی پردازنده‌های گرافیکی برنامه‌نویسی کرد. میان‌افزار کودا بر پایه زبان سی^۲ طراحی و نوشته شده است و شامل یک کامپایلر و کتابخانه‌هایی برای اجرا و همکاری با پردازنده مرکزی است.

¹ Framework

² C Language

شرکت انویدیا هزمان با ارتقاء نسل و معماری پردازنده‌های خود، میان‌افزار کودا را هم بسط و گسترش می‌دهد. همانطور که اولین نسل معماری پردازنده‌های گرافیکی دارای محدودیت‌هایی مانند:

- عدم انجام محاسبات با دقت مضاعف^۱
- نداشتن توانایی در ارتباط با دیگر قسمت‌های سیستم و در نتیجه کاربرد نداشتن توابع ورودی - خروجی مانند *printf*

بود، کودا نیز در نسخه‌های اولیه خود بسیار ابتدایی و با توابع کتابخانه‌ای بسیار محدود بود. اما با پیشرفت و توسعه سخت‌افزاری، نسل‌های مختلف معماری پردازنده، با عنوان قابلیت محاسباتی^۲ شناخته می‌شوند. اما تا این لحظه تا نسخه ۷ میان‌افزار کودا منتشر شده است که دارای توانایی‌های بسیاری است که توضیحاتی در مورد توانایی‌های جدید در نسخه‌های اخیر در فصل‌های بعدی داده می‌شود. در جدول ۱ قابلیت محاسباتی تراشه‌های پردازنده گرافیکی و کارت‌های متناظر آن‌ها آورده شده‌اند. کارت‌های معرفی شده در جدول ۱ پرچم‌داران هر نسل از قابلیت محاسباتی هستند [22].

جدول ۱: قابلیت محاسباتی و تراشه و کارت‌های متناظر آن‌ها

	1.0	1.2	1.3	2.0	2.1	3.0	3.5	5.0	5.2
تراشه	G80	GT215	GT200	GF100 GF110	GF106	GK104	GK110	GM107, GM108	GM 200
کارت	8800 GTX	GT 240	GTX 285	GTX 480	GTS 450	GTX 680	Tesla K20	GeForce GTX 750	GTX 950

¹ Double percision

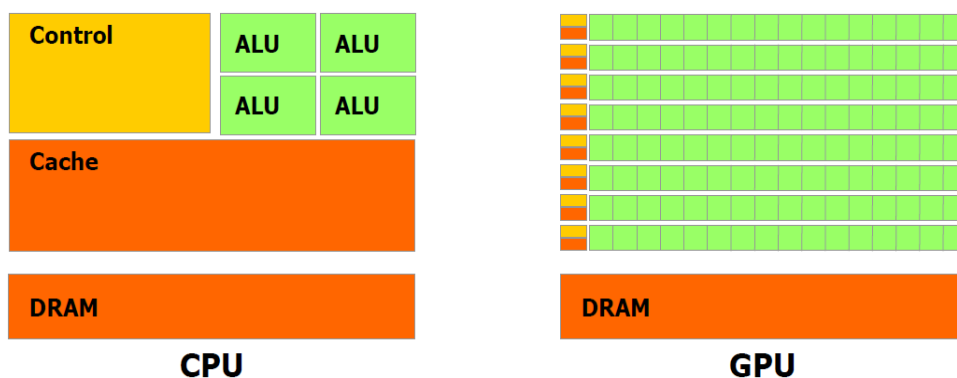
² Compute Capablity

پردازنده‌های گرافیکی از نظر ریز معماری^۱ هم دسته‌بندی جداگانه‌ای دارند که در جدول ۲ مشاهده می‌شوند [22].

جدول ۲: ریز معماری‌های مختلف و قابلیت‌های محاسباتی متناظر با آن‌ها

نام ریز معماری	Tesla	Fermi	Kepler	Maxwell
قابلیت محاسباتی	1 – 1.2	2 , 2.1	3 - 3.7	5.0 – 5.3

اگر مقایسه‌ای بین پردازنده مرکزی و گرافیکی انجام شود، اولین تفاوتی که جلب توجه می‌کند تفاوت در تعداد هسته‌های پردازشی آن‌ها است. در شکل ۳-۱ به صورت شماتیک ساختار پردازنده مرکزی و گرافیکی نمایش داده شده‌اند. در شکل ۳-۱ خانه‌های سبز رنگ نشان‌دهنده واحد پردازش عددی و منطقی هستند که در پردازنده گرافیکی به تعداد بسیار بالا دیده می‌شوند (ویژگی بارز پردازنده گرافیکی). اما از نظر حافظه کش و واحد کنترل از پردازنده مرکزی ضعیف‌تر است.



شکل ۳-۱: مقایسه ساختار معماری پردازنده مرکزی و گرافیکی

¹ Micro Architecture

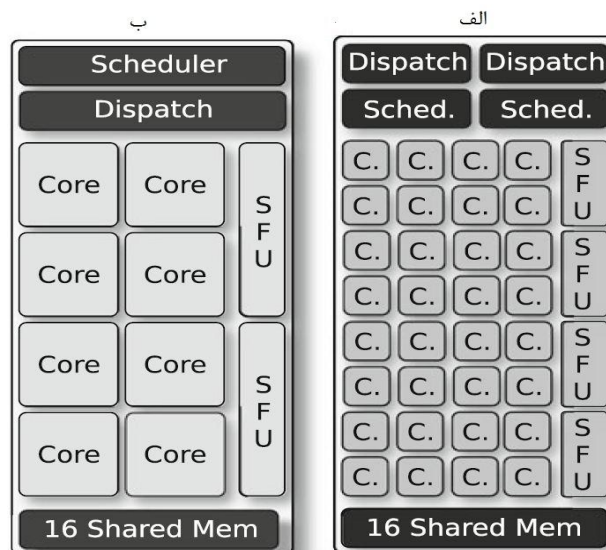
لازم به ذکر است که ^۱ALU در پردازنده مرکزی در واقع واحدهایی برای اجرا و نظارت بر عملیات منطقی و محاسبات رقمی پردازنده هستند.

۳-۳ اجزای سختافزاری و نرمافزاری پردازنده گرافیکی

برای بررسی بیشتر، مهم‌ترین اجزای سختافزاری و نرمافزاری پردازنده گرافیکی، در ادامه از نظر گذرانده می‌شوند.

۱-۳-۳ چندپردازنده

در نگاهی دقیق‌تر و موشکافانه‌تر به یک چندپردازنده ^۲گرافیکی، شکلی مانند شکل ۲-۳ خواهد بود. در این شکل در قسمت الف اجزای یک چندپردازنده از نسل تسلا و فرمی در کنار هم نشان داده شده‌اند.



شکل ۲-۳: شکل شماتیک یک چندپردازنده : الف) معماری فرمی ب) معماری تسلا

^۱ Arithmetic Logic Unit

^۲ Multi-Processor

در شکل ۲-۳ دو چندپردازنده از دو نسل مختلف دیده می‌شوند که چندپردازنده الف از نسل فرمی با ۳۲ هسته و چندپردازنده ب از نسل تسلا با ۸ پردازنده جریان^۱ یا هسته کودا^۲ هستند. چندپردازنده‌های جریانی از پایه‌ای‌ترین المان‌های سخت‌افزاری پردازش موازی با پردازنده گرافیکی هستند. این هسته‌های پردازشی شباهت‌های زیادی با هسته‌های پردازنده مرکزی دارند. این هسته-ها می‌توانند کلاک مخصوص به خود را داشته باشند و با کلاک اسمی کل پردازنده گرافیکی تفاوت داشته باشد.

در نتیجه یک چندپردازنده، مجموعه‌ای از هسته‌های پردازشی به همراه حافظه اشتراکی و یک یا چند واحد دستور^۳ هستند. در ریز معماری‌های مختلف تعداد هسته‌های پردازشی مختلف هستند (شکل ۲-۳). هر چندپردازنده یک سری ثبات دارد که به هسته‌های پردازشی اختصاص داده شدند که نخ‌های پردازشی^۴ (مجموعه‌ای از دستورات- یک واحد نرم‌افزاری) بتوانند روی هر هسته اجرا شوند. هر چندپردازنده در نسل تسلا به‌طور همزمان می‌تواند ۷۶۸ نخ را به شکل همزمان اجرا کند و در کل می‌تواند ۱۲,۲۸۸ نخ را همزمان اجرا کند. این عدد برای معماری فرمی ۱۵۳۶ و برای معماری کیپلر ۲۰۴۸ عدد است. همانطور که گفته شد چندپردازنده‌ها چند المان سخت‌افزاری دیگر دارند که در مورد آن‌ها صحبت خواهد شد.

۲-۳-۳ ریزمان

هر چندپردازنده نخ‌های مربوط به خود را به شکل یک دستورالعمل- چند داده اجرا می‌کند. بنابراین در هر چندپردازنده گروهی از نخ‌ها همیشه و همزمان با هم اجرا می‌شوند که به آن‌ها

¹ Streaming Processor

² CUDA Core

³ Instruction Unit

⁴ Thread

ریسمان^۱ می‌گویند. استراتژی‌های مختلفی برای اجرای ریسمان روی یک چندپردازنده وجود دارد، برای مثال در تراشه مدل G80 که ۸ هسته پردازشی و یک واحد دستور دارد، تنها یک ریسمان قابلیت اجرا دارد. اما از نسل تراشه GF100 به بعد دو ریسمان همزمان می‌توانند اجرا شوند.

۳-۳-۳ انشعاب

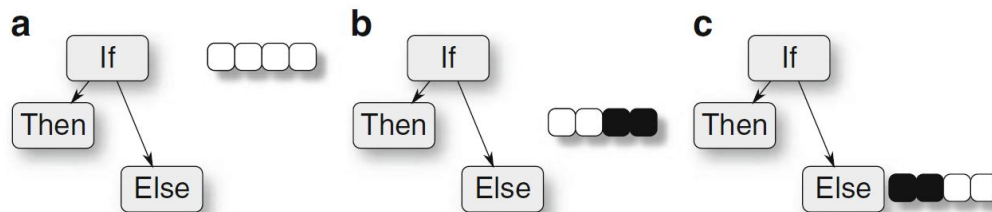
همان‌طور که ذکر شد در یک چندپردازنده، تعدادی هسته زیر نظر یک واحد دستور قرار دارند. اجرای دستورات غیر یکسان در یک زمان روی این هسته‌ها امکان‌پذیر نیست. برای اجرای چنین برنامه‌ای که در محاسبات بسیار پیش می‌آید، هسته‌ها مجبور هستند تا منشعب شوند (برنامه ۱)، گروهی از هسته‌ها قسمت if را انجام دهند درحالی‌که بقیه هسته‌ها معطل هستند و در قسمت بعدی هسته‌های گروه if معطل و هسته‌های گروه else مشغول محاسبات می‌شوند. (شکل ۳-۳)

```
If ( threadId/2==0) {//then part}  
Else {// else part}
```

آشکار است که این انشعاب^۲ باعث معطل ماندن گروهی از نخ‌ها و در نتیجه کاهش راندمان شود. فقط برای هسته‌های پردازشی که داخل یک چندپردازنده هستند انشعاب تعریف می‌شود نه برای هسته‌های دو چندپردازنده جدا از هم [22].

¹ Warp

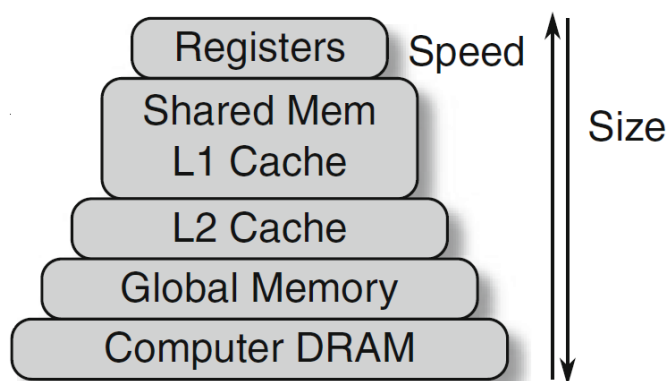
² Divergence



شکل ۳-۳: انشعاب ایجاد شده در هسته‌های پردازشی و نخ‌ها توسط دستور شرطی if [22]

۳-۳-۴ انواع حافظه در پردازنده گرافیکی

با توجه به تعداد بسیار زیاد هسته‌های پردازشی و نوع معماری خاص آن‌ها، به تناسب به سلسله مراتبی از حافظه برای پاسخگویی به این نوع معماری نیاز است. در شکل ۳-۴ سلسله مراتب حافظه در پردازنده‌های گرافیکی نشان داده شده است. در این شکل از بالا به پایین سرعت پاسخگویی کم و مقدار حافظه زیاد می‌شود. در مورد سلسله مراتب این حافظه‌ها بیشتر توضیح داده خواهد شد.



شکل ۳-۴: سلسله مراتب حافظه در کارت گرافیک [22]

۳-۳-۴-۱ حافظه نهان یا کش

این حافظه یک حافظه بسیار سریع و با مقدار بسیار کم است که روی تراشه پردازنده نزدیک به هسته‌های پردازشی تعبیه شده، که دستور بعدی که قرار است پردازنده پردازش کند به‌صورت موقت در خود جای می‌دهد. کاربرد حافظه نهان طی توسعه پردازش موازی با پردازنده گرافیکی بسیار تغییر کرده است. برای مثال در اولین نسل این پردازنده‌ها این حافظه‌ها تنها خواندنی بوده‌اند، اما در نسل فرمی این قابلیت به خواندنی و نوشتنی ارتقا پیدا کرد. در نسل فرمی به بعد حافظه نهان به دو بخش لایه ۱ و لایه ۲ تبدیل شد. در مورد مقدار حافظه هم برای مثال در نسل معماری فرمی برای هر چندپردازنده حافظه نهان لایه ۱ تنها ۴۸ کیلوبایت فضا دارد که بین هسته‌ها این مقدار تقسیم می‌شود. همچنین لایه ۲ نیز حداکثر مقدار ۷۶۸ کیلوبایت دارد.

۳-۳-۴-۲ حافظه مشترک

هر چندپردازنده به یک حافظه به نام حافظه مشترک دسترسی دارد که درواقع بخشی از لایه اول حافظه نهان است. تمام نخ‌های واقع در یک بلوک یک چندپردازنده می‌توانند مقادیر را از این حافظه بخوانند و بنویسند. اما نخ‌های یک چندپردازنده دیگر به حافظه دیگر چندپردازنده‌ها دسترسی ندارند. این حافظه در قسمت‌هایی به نام بانک دسته بندی شده‌اند و سرعت ورود و خروج اطلاعات در آن بسیار زیاد است. یکی از نکاتی که در استفاده از این حافظه باید دقت کرد، عدم تداخل و عدم استفاده همزمان دو هسته از یک بانک حافظه است که با نام تداخل بانک^۱ مطرح می‌شود و باعث افت راندمان و سرعت پردازش می‌شود.

¹ Bank Conflict

۳-۳-۴ ثبات‌ها و حافظه محلی

هر نخ یک حافظه محلی و مخصوص به خود دارد. حافظه محلی روی حافظه سراسری است پس حافظه محلی همان خواص حافظه سراسری را دارد. ثبات‌ها^۱ روی تراشه پردازنده قرار دارند و دارای بیشترین سرعت در پاسخگویی هستند. متغیرها وقتی در برنامه نویسی با کودا تعریف می‌شوند روی حافظه سراسری یا محلی قرار می‌گیرند. اما کودا به شکل خودکار بعضی از این متغیرها را با توجه به مقدار آن‌ها به ثبات‌ها انتقال می‌دهد و کاربر کنترلی روی این فرآیند ندارد. بسیار واضح است که تعداد حافظه ثبات در پردازنده گرافیکی اهمیت بیشتری نسبت به پردازنده‌های معمولی دارد. نبود یا کمبود این حافظه سبب مراجعات بیشتر به حافظه سراسری می‌شود و این خود باعث کاهش راندمان و سرعت پردازش است. در جدول ۳ تعداد ثبات‌هایی که یک نخ می‌تواند استفاده کند و تعداد ثباتی که یک چندپردازنده می‌تواند استفاده کند در پردازنده‌های نسل‌های مختلف نشان داده شده است.

جدول ۳: پیشرفت و تغییرات ثبات‌ها در طی زمان و در تراشه‌های گرافیکی از نسل‌های مختلف

GK110	GK104	GF1xx	GT200	G8x-G9x	
255	63	63	128	128	تعداد ثبات برای هر نخ
64000	64000	32000	16000	8000	تعداد ثبات در هر چندپردازنده

۳-۳-۴ حافظه سراسری

بزرگترین مقدار حافظه از نظر ظرفیت، مقدار ظرفیت حافظه سراسری است. همچنین این

¹ Registers

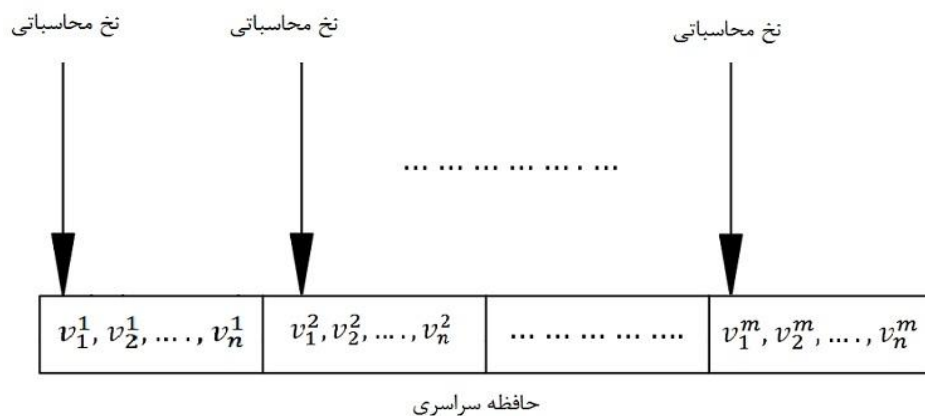
حافظه پهنای باند وسیعی دارد و مقادیر موضعی و سراسری روی این حافظه ذخیره سازی می‌شوند. تمام هسته‌ها، چندپردازنده‌ها و طبیعتاً تمام نخ‌های محاسباتی به این حافظه دسترسی دارند و اجازه خواندن و نوشتن روی این حافظه را نیز دارا هستند. یکی از نکاتی که در دسترسی نخ‌ها به حافظه سراسری باید مد نظر داشت، این مسأله است که نخ‌های موازی در هر لحظه به یک منطقه از مکان حافظه دسترسی داشته باشند. به علت بالا بودن تعداد نخ‌ها و امکان دسترسی همه نخ‌ها به حافظه سراسری، اگر هر نخ بخواهد به قسمت‌های مختلف حافظه از نظر آدرس و محل قرارگیری، دسترسی داشته باشد، با توجه به محدود بودن پهنای باند، سرعت پردازش کاهش پیدا می‌کند. به این نوع دسترسی به حافظه سراسری دسترسی غیر هم‌مکان^۱ می‌گویند (شکل ۳-۵). اما روش مناسب برای استفاده از حافظه سراسری دسترسی هم‌مکان^۲ است یعنی در هر لحظه تمام نخ‌ها به محدوده مشخصی از حافظه که در کنار هم قرار دارند رجوع کنند و مقدار مورد نظر را فراخوانی یا ذخیره کنند (شکل ۳-۶). در شکل‌های ۳-۵ و ۳-۶ به عنوان نمونه یک آرایه دو بعدی (برای مثال یک ماتریس که به صورت خطی ذخیره شده است) در حافظه سراسری نمایش داده شده و نوع دسترسی نخ‌ها نیز مشخص هستند. در حالت دسترسی غیر هم‌مکان هر نخ به مقادیر ابتدای هر سطر ماتریس دسترسی دارند. اما در دسترسی هم‌مکان نخ‌ها به مقادیر سطر اول رجوع می‌کنند و این مقادیر در کنار یکدیگر قرار دارند، به همین دلیل با فراخوانی قسمتی از حافظه نیاز کل نخ‌ها بر طرف می‌شود. دسترسی هم‌مکان بهترین حالت برای بالا نگه داشتن سرعت پردازش است که در ادامه تاثیر این مسأله روی سرعت بررسی شده است.

از دیگر ویژگی‌های حافظه سراسری امکان ارتباط این حافظه با پردازنده مرکزی و حافظه آن است. حافظه‌های مشترک، ثابت و نهان امکان ارتباط مستقیم با میزبان را ندارند و تنها حافظه‌ای

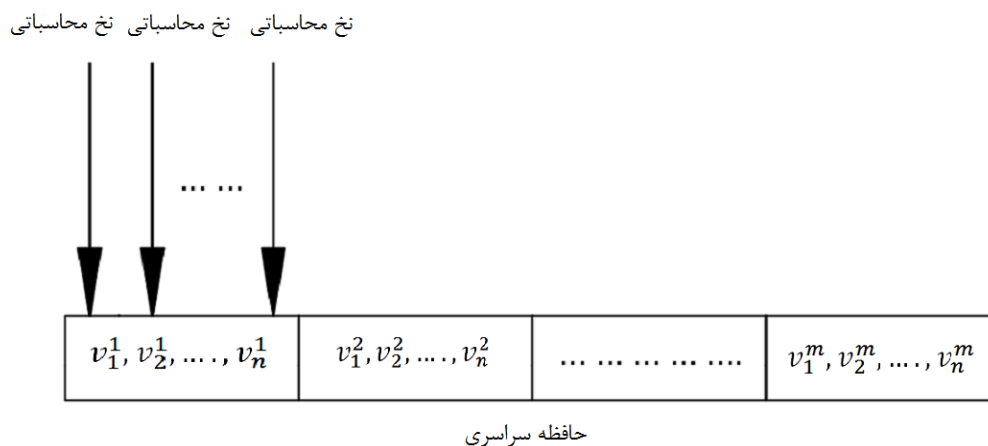
^۱ UnCoalesced Memory Access

^۲ Coalesced Memory Access

که چنین ویژگی دارد همین حافظه سراسری است. پس در محاسبات اگر نیاز بود تا داده‌ای برای پردازش موازی از میزبان به دستگاه برای انجام پردازش موازی منتقل شود، این اطلاعات روی حافظه سراسری ذخیره خواهند شد و در صورت نیاز از آنجا به دیگر حافظه‌ها منتقل شوند.



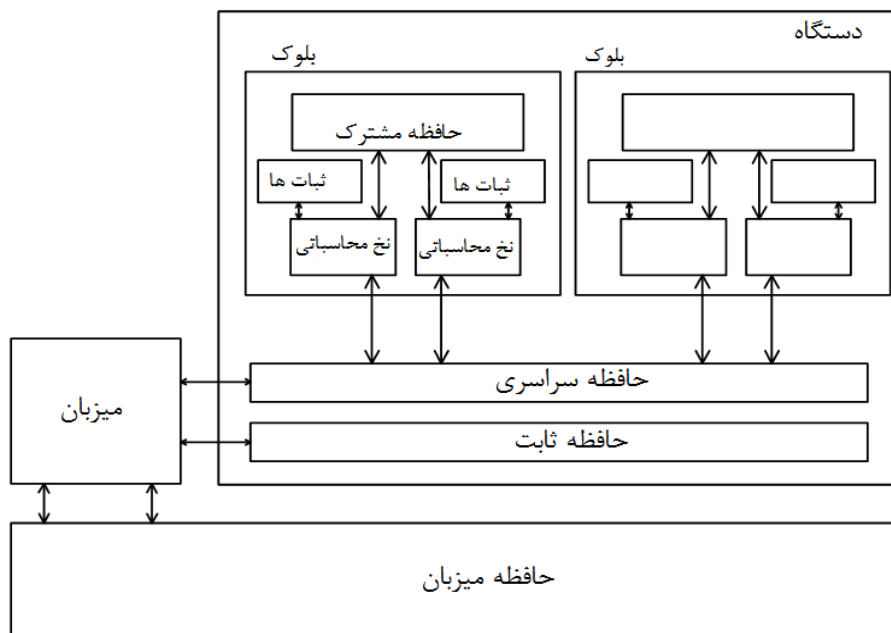
شکل ۳-۵: دسترسی غیر هم‌مکان نخ‌های محاسباتی به حافظه سراسری



شکل ۳-۶: دسترسی مناسب و هم‌مکان نخ‌های محاسباتی به حافظه سراسری

در انتها در شکل ۳-۷ یک شمای کلی از نوع برقراری ارتباط بین نخ‌های محاسباتی و انواع

حافظه نمایش داده شده است. همچنین رابطه بین میزبان و پردازنده گرافیکی یا همان دستگاه نیز نشان داده شده‌اند.



شکل ۳-۷: رابطه بین انواع حافظه و نخ‌های محاسباتی و ارتباط کلی بین میزبان و پردازنده مرکزی

۳-۳-۵ ملاحظات نرم‌افزاری و برنامه‌نویسی روی پردازنده‌های

گرافیکی

تاکنون در مورد هسته‌های پردازشی و انواع حافظه در پردازنده‌های گرافیکی توضیحاتی داده شده است. اما در پردازش موازی موضوعات دیگری نیز تاثیر گذار هستند و آن انتخاب و سازماندهی کمیت‌های نرم‌افزاری هستند.

گفته شد که کوچکترین واحد نرم‌افزاری، یک سری وظایف و دستوراتی هستند که به یک هسته سپرده می‌شوند تا آن‌ها را انجام دهد و به آن‌ها نخ یا رشته می‌گویند. این نخ‌ها در دسته-

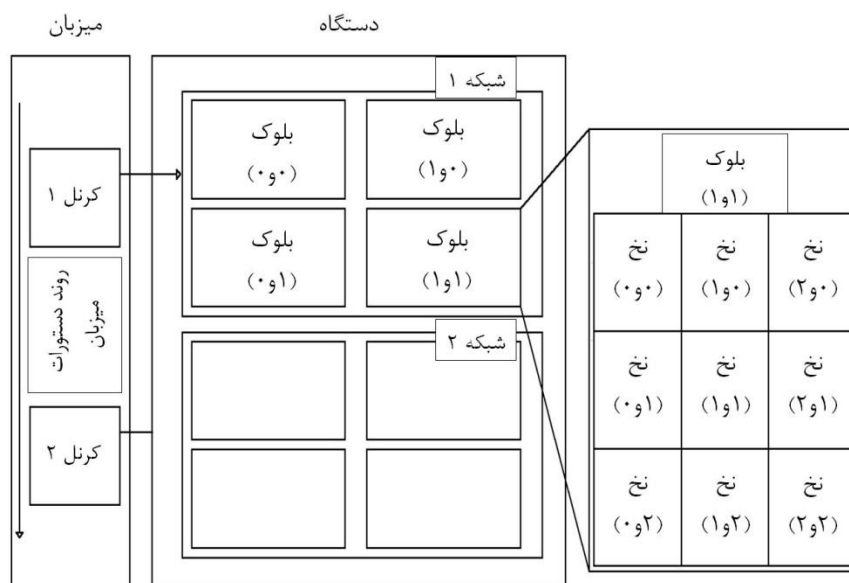
هایی تقسیم بندی می‌شوند به نام بلوک^۱، که می‌توانند یک، دو یا سه بعدی باشند. بلوک‌ها نیز در دسته‌های بزرگتری دسته‌بندی می‌شوند با نام شبکه که بزرگترین واحد تقسیم‌بندی نخ‌ها به حساب می‌آیند (شکل ۳-۸). روند کلی برنامه‌نویسی با پردازنده‌های گرافیکی به این صورت است که برنامه ابتدا روی میزبان شروع به کار می‌کند، سپس در هر قسمتی از برنامه که الگوریتم قابلیت اجرای همزمان یا موازی را داشت، پردازنده مرکزی با فراخوانی پردازنده گرافیکی و انتقال اطلاعات به آن این قسمت از پردازش را به عهده هسته‌های پرتعداد و قدرتمند پردازنده گرافیکی می‌گذارد.

فراخوانی پردازنده گرافیکی توسط یک کرنل^۲ صورت می‌گیرد. کرنل‌ها همانند توابع در زبان سی هستند که با فراخوانی آن‌ها پردازنده گرافیکی دستورات داده‌شده در کرنل را انجام و در صورت نیاز اطلاعات را به میزبان برمی‌گرداند، سپس برنامه روند عادی خود را طی می‌کند و پردازنده گرافیکی از چرخه پردازش خارج می‌شود تا دوباره در صورت لزوم توسط یک کرنل دیگر فراخوانی شود. فراخوانی با کرنل‌ها علاوه بر این که نیاز به متغیرهای اعلان شده‌ای در ورودی خود دارد اطلاعات مربوط به ابعاد بلوک‌ها و شبکه و تعداد بعد آن‌ها باید برای کرنل و پردازنده گرافیکی باید مشخص باشد. با توجه به اطلاعات ورودی در مورد ابعاد و تعداد نخ‌ها و بلوک‌ها در کرنل، پردازنده گرافیکی با دسته بندی هسته‌ها شروع به پردازش می‌کند. در شکل ۳-۸ مشخص است که هنگام فراخوانی کرنل نخ در بلوک‌ها و شبکه طبقه بندی شده‌اند. اما در مسائل مختلف انتخاب و با توجه به نوع معماری پردازنده گرافیکی استفاده شده، تعداد نخ‌ها و بلوک‌ها و ابعاد آن‌ها با محدودیت‌هایی روبرو است. برای مثال در معماری فرمی تعداد نخ‌ها در یک بلوک به صورت $64 \times 64 \times 1024 \times 1024$ نمایش داده می‌شود یعنی در دو بعد حداکثر تعداد نخ‌ها ۱۰۲۴ عدد است و در بعد سوم حداکثر ۶۴ نخ در یک بلوک قابل استفاده خواهد بود و در مورد تعداد بلوک‌ها در یک

¹ Block

² Kernel

شبکه، در هر بعد شبکه تعداد ۶۵۵۳۵ عدد بلوک قابل دسترسی و استفاده هستند.



شکل ۳-۸: ساختار اجرای برنامه با کمک پردازنده گرافیکی و تقسیم‌بندی نخ‌ها در هنگام فراخوانی کرنل‌ها

نمایش و مقداره‌ی به تعداد نخ‌ها و بلوک‌ها در فراخوانی یک کرنل به صورت زیر است:

$kernel \lll blocks\ per\ grid , threads\ per\ blok \ggg (input\ variable1 , \dots)$

تفاوت فراخوانی توابع در کودا و زبان سی فقط در علامت «»,» است. عدد اول تعداد بلوک در یک شبکه و عدد دوم تعداد نخ در یک بلوک را نمایش می‌دهد. همان‌طور که ذکر شد انتخاب تعداد نخ در بلوک و تعداد بلوک‌ها می‌تواند نقش مؤثری در بازدهی پردازش ایجاد کند. وقتی به یک چندپردازنده یک یا چند بلوک برای پردازش سپرده می‌شود، چندپردازنده آن‌ها را به ریسمان تبدیل می‌کند با توجه به این که تعداد نخ در ریسمان ۳۲ است بهتر است تعداد نخ در هر

بلوک ضربی از ۳۲ باشد برای مثال ۱۲۸ یا ۲۵۶. اما تعداد بلوک‌ها در هر شبکه می‌تواند مقادیر مختلفی داشته باشد. در نهایت خلاصه‌ای از محدودیت نخ‌ها و بلوک‌ها در پردازنده‌ها با قابلیت‌های پردازشی متفاوت در جدول ۴ نمایش داده شده‌اند.

جدول ۴: محدودیت پردازنده‌ها با قابلیت محاسباتی مختلف از نظر تعداد نخ و بلوک

قابلیت محاسباتی				
3-3.5	2.x	1.2-1.3	1-1.1	
2	2	2	2	حداکثر ابعاد بلوک‌ها
$2^{31} - 1$	2^{16}	2^{16}	2^{16}	حداکثر اندازه در راستای x
2^{16}	2^{16}	2^{16}	2^{16}	حداکثر اندازه در راستای z,y
3	3	3	3	حداکثر ابعاد نخ‌ها
64	64	64	64	حداکثر ابعاد نخ‌ها در راستای z
32	32	32	32	ابعاد ریسمان
1024	1024	512	512	حداکثر تعداد نخ‌ها در بلوک
2048	1536	1024	768	حداکثر تعداد نخ‌ها در چندپردازنده
16	16	8	8	حداکثر تعداد بلوک‌ها در چندپردازنده

فصل چهارم:

شبیه‌سازی جریان با معادلات

سه‌قطری و پیاده‌سازی آن‌ها روی

پردازنده گرافیکی

۴-۱ مقدمه

دستگاه‌های خطی در گسسته‌سازی بسیاری از مسائل مهندسی پدیدار می‌شوند، مخصوصاً آن‌هایی که با روش‌هایی مانند تفاضل محدود^۱ یا حجم محدود^۲ گسسته می‌شوند. برای حل دستگاهی مانند $Ax = b$ می‌توان با معکوس کردن ماتریس ضرایب (A) و ضرب آن در طرفین معادله بردار مجهولات را به دست آورد. اما چنین عملیاتی از نظر هزینه محاسباتی بسیار سنگین و وقت‌گیر است. مخصوصاً در مسائل غیرخطی مانند حل معادلات ناویر استوکس که برای حل باید با تکرار به همگرایی رسید، بارها و بارها وارون کردن یک ماتریس بسیار نامطلوب است.

اما خوشبختانه روش‌های دیگری نیز برای حل موجود هستند که از لحاظ پیچیدگی دارای روندی ساده‌تر و هزینه محاسباتی مطلوب هستند. روش‌هایی مانند گرادیان مزدوج، روش حذفی گوس، روش فاکتورگیری LU و غیره که هرکدام نقاط ضعف قوت مخصوص به خود را دارند.

در بین دستگاه معادلات، دستگاه معادلات سه‌قطری یکی از پرکاربردترین انواع هستند که در روش جهت متناوب ضمنی^۳ [23]، حل گر طیفی پواسون^۴ [24]، تقریب $Spline$ درجه سه^۵ و روش‌های پیش‌شرط‌سازی^۶ برای حل‌های تکراری و غیره ظاهر می‌شوند. یک روش بسیار مؤثر و سریع در حل چنین دستگاه‌هایی الگوریتم توماس^۷ است که بر پایه روش حذفی گوس می‌باشد. اما این الگوریتم یک الگوریتم ترتیبی است و برای پردازش موازی مناسب نیست.

اما در پردازش موازی هنگام برخورد با این دسته از معادلات نیاز به الگوریتم‌هایی است که

¹ Finite Difference

² Finite Volume

³ Alternating Direction Implicite Method (ADI)

⁴ Spectral Poisson

⁵ Cunic Spline Approximation

⁶ Preconditioner

⁷ Thomas Algorithm

قابلیت اجرای موازی را داشته باشند و بتوانند کارایی پردازش را بالا نگه دارند بنابراین برای حل دستگاه معادله سه قطری با رویکرد موازی جایگزین‌هایی پیشنهاد شده‌اند. دو الگوریتم کاهش متناوب [8] و الگوریتم کاهش متناوب موازی [8] دو الگوریتم موازی پرکاربرد برای حل معادلات سه قطری به صورت موازی هستند. در ادامه طریقه به کارگیری و پیاده‌سازی روش‌هایی که به معادلات سه قطری منجر می‌شوند، بررسی می‌شوند.

۴-۲ دستگاه معادلات سه قطری

یک حل گر دستگاه معادلات سه قطری، الگوریتمی است که بردار مجهولات x را در $Ax = d$ می‌یابد، که A ماتریس ضرایب با سه قطر غیر صفر و d برداری با مقادیر معلوم است. در ماتریس ضرایب نیز سه قطر اصلی، قطر بالا و قطر پایین به ترتیب با b ، c و a نمایش داده شده‌اند (شکل ۴-۱). الگوریتم‌های متفاوتی برای حل چنین دستگاه معادلاتی وجود دارد اما به این دلیل که الگوریتم توماس یک الگوریتم بسیار سریع است برای به چالش کشیدن توانایی و سرعت پردازنده‌های گرافیکی در حل مسائل، انتخاب شده است. همان‌طور که گفته شد الگوریتم توماس حالت خاصی از روش حذفی گوس است. در این تحقیق الگوریتم توماس رایج و شناخته شده به نام الگوریتم توماس کلاسیک ذکر می‌شود تا از رهیافت توماس با رویکرد موازی که در ادامه معرفی می‌شود تمایز داده شود.

$$\begin{bmatrix} b_1 & c_1 & 0 & 0 \\ a_2 & b_2 & c_2 & 0 \\ 0 & a_3 & b_3 & c_3 \\ 0 & 0 & a_4 & b_4 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \end{bmatrix}$$

شکل ۴-۱: نمایش ماتریس سه قطری

۴-۲-۱ الگوریتم توماس کلاسیک

یکی از سریع ترین و پرکاربردترین الگوریتم های حل دستگاه معادلات الگوریتم توماس است که به اختصار در اینجا معرفی می شود. الگوریتم توماس شامل دو مرحله است: (الف) کاهش پیشرو (ب) جایگزینی پس رو. در مرحله کاهش پیشرو قطر پایین قطر اصلی به صورت ترتیبی حذف می شوند و با این عمل روی قطر بالا و سمت راست معادله تغییراتی ایجاد می شود که در معادلات (۱-۴) و (۲-۴) مشخص هستند. در مرحله جایگزینی پس رو مطابق معادله (۳-۴)، الگوریتم با استفاده از قطر اصلی و قطر بالا از آخرین سطر شروع به بدست آوردن بردار مجهولات می کند تا تمام جواب ها محاسبه شوند.

$$c'_1 = \frac{c_1}{b_1}, c'_i = \frac{c_i}{b_i - c'_{i-1}a_i}, i = 2, 3, \dots, n-1 \quad (1-4)$$

$$d'_1 = \frac{d_1}{b_1}, d'_i = \frac{d_i - d'_{i-1}a_i}{b_i - c'_i a_i}, i = 2, 3, \dots, n \quad (2-4)$$

$$x_n = d'_n, x_i = d'_i - c'_i x_{i+1}, i = n-1, n-2, \dots, 1 \quad (3-4)$$

۴-۲-۲ الگوریتم کاهش متناوب

الگوریتم کاهش متناوب نیز مانند الگوریتم توماس یک الگوریتم دو مرحله ای است: (الف) کاهش پیشرو (ب) جایگزینی پس رو. در مرحله پیشرو، الگوریتم اندازه دستگاه را در هر مرحله به نصف کاهش می دهد و این روند را ادامه می دهد تا جایی که به دو معادله مستقل برسد. در هر مرحله مقادیر قطر اصلی و قطرهای بالا و پایین و سمت راست معادله طبق معادلات (۴-۴) تا (۶-۴) تغییر می کنند. پس از آن که با استفاده از معادلات (۴-۴) تا (۶-۴)، دستگاهی به اندازه ۲ در ۲

حاصل شد، این دستگاه دو معادله‌ای با روش‌های مرسوم حل می‌شود. در ادامه با استفاده از معادله (۷-۵) می‌توان تمام مجهولات دیگر را بدست‌آورد. این مرحله جایگزینی پس‌رو نام دارد.

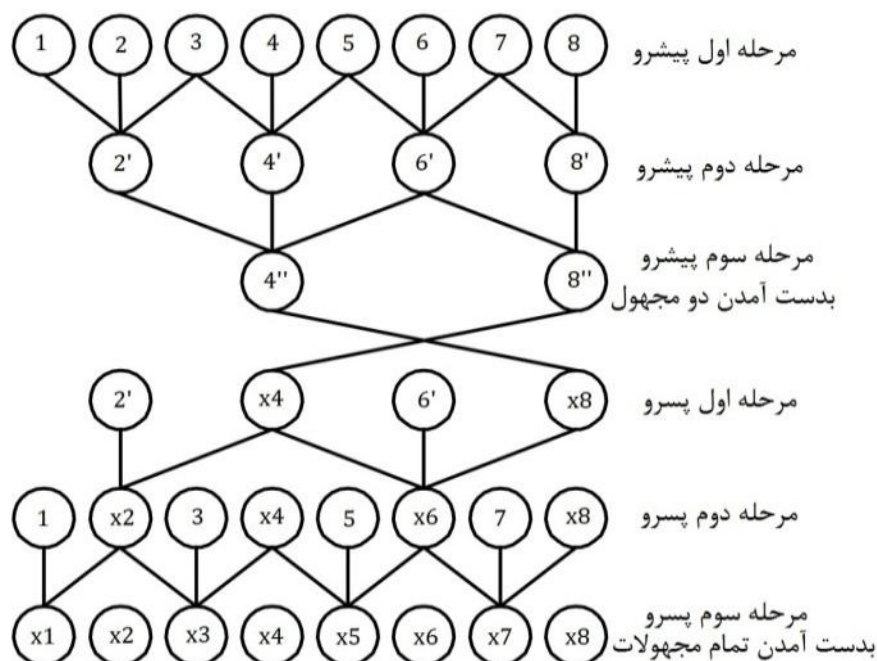
$$a'_i = -a_{i-1}k_1, b'_i = b_i - c_{i-1}k_1 - a_{i+1}k_2 \quad (۴-۴)$$

$$c'_i = -c_{i+1}k_2, d'_i = d_i - d_{i-1}k_1 - d_{i+1}k_2 \quad (۵-۴)$$

$$k_1 = \frac{a_i}{b_{i-1}}, k_2 = \frac{c_i}{b_{i+1}} \quad (۶-۴)$$

$$x_i = \frac{d'_i - a'_i x_{i-1} - c'_i x_{i+1}}{b'_i} \quad (۷-۴)$$

شکل ۴-۲ نشان دهنده الگوی الگوریتم کاهش متناوب برای یک دستگاه به اندازه ۸ در ۸ است. با سه بار اجرای الگوریتم روی دستگاه معادلات، با نصف شدن اندازه دستگاه در هر مرحله، در مرحله سوم دو معادله با دو مجهول به دست می‌آید. نکته مهم در این الگوریتم این است که در هر مرحله تعدادی از معادلات بدون تغییر باقی می‌مانند. برای مثال در شکل ۵-۳، معادلات اول، سوم، پنجم و هفتم در تمام مراحل بدون تغییر باقی می‌مانند. در مرحله دوم نیز ضرایب معادلات دوم و چهارم که در مرحله اول ضرایب آن‌ها تغییر کردند، در فرآیند مرحله سوم بدون تغییر باقی می‌مانند.

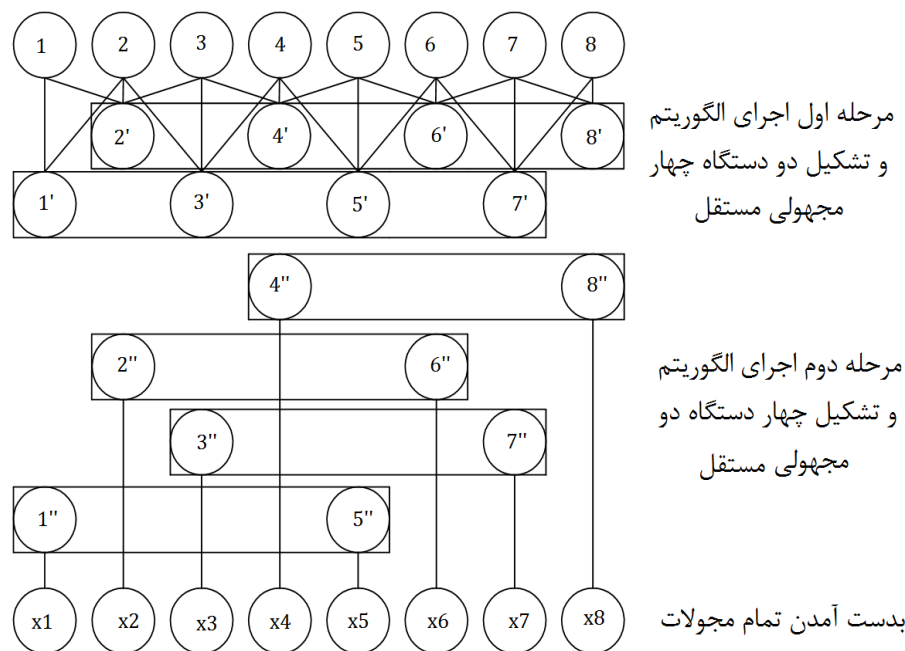


شکل ۴-۲: الگوریتم کاهش متناوب برای یک دستگاه ۸ معادله و ۸ مجهول

۴-۲-۳ الگوریتم کاهش متناوب موازی

الگوریتم کاهش متناوب موازی، یک الگوریتم بهبود یافته بر پایه الگوریتم کاهش متناوب است. اما بر عکس الگوریتم کاهش متناوب فقط دارای یک مرحله است و آن مرحله کاهش پیشرو است. تفاوت دیگر این دو الگوریتم این است که در الگوریتم کاهش متناوب موازی در هر مرحله ضرایب تمام معادلات تغییر می کنند. برای مثال در شکل ۴-۳ مراحل الگوریتم کاهش متناوب موازی برای یک دستگاه ۸ معادله ۸ مجهول به صورت شماتیک نشان داده شده اند. در هر مرحله از الگوریتم کاهش متناوب موازی دستگاه معادلات به دو دستگاه کاملاً مستقل تبدیل می شوند. به منظور درک بهتر این موضوع برای یک دستگاه ۴ در ۴، در شکل ۴-۴ و یک مرحله اعمال الگوریتم کاهش متناوب موازی نشان داده شده است. همان طور که در شکل ۴-۴ دیده می شود، پس از

اجرای یک مرحله سطرهای اول و سوم با یکدیگر و سطرهای دوم و چهارم ماتریس ضرایب با یکدیگر تشکیل دو دستگاه مستقل از هم 2×2 را می‌دهند که به راحتی قابل حل کردن می‌باشند. در شکل ۴-۴ فقط ماتریس ضرایب نشان داده شده است و با توجه به معادله (۴-۵) تغییراتی روی بردار سمت راست معادله نیز صورت می‌گیرد. مرحله پیشرو در الگوریتم کاهش متناوب موازی بر پایه الگوریتم کاهش متناوب است و از معادلات (۴-۴) تا (۶-۴) پیروی می‌کند. اما به جای مرحله پس‌رو که به دست آوردن مجهولات با استفاده از معادله (۷-۴) است، الگوریتم کاهش متناوب موازی به تعداد مشخصی دستگاه دو معادله دو مجهول می‌رسد که به صورت موازی این دستگاه‌های مستقل قابل حل هستند. یکی از برتری‌های الگوریتم کاهش متناوب موازی بر الگوریتم کاهش متناوب در همین امر است که در هر لحظه تمام نخ‌های محاسباتی در حال انجام وظایف خود هستند.



شکل ۴-۳: مراحل الگوریتم کاهش متناوب موازی برای یک دستگاه هشت مجهولی

اما در الگوریتم کاهش متناوب در بیشتر مراحل، تعدادی از نخ‌های محاسباتی معطل می‌مانند و همین موضوع در عملکرد این الگوریتم تأثیر منفی دارد. بهترین حالت در پردازش موازی این است که در روند اجرای برنامه تمام نخ‌ها درگیر محاسبات باشند.

$$\begin{array}{c}
 \begin{array}{l} 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{bmatrix} b_1 & c_1 & 0 & 0 \\ a_2 & b_2 & c_2 & 0 \\ 0 & a_3 & b_3 & c_3 \\ 0 & 0 & a_4 & b_4 \end{bmatrix} \\
 \Downarrow \\
 \begin{array}{l} 1' \\ 2' \\ 3' \\ 4' \end{array} \begin{bmatrix} b'_1 & 0 & c'_1 & 0 \\ 0 & b'_2 & 0 & c'_2 \\ a'_3 & 0 & b'_3 & 0 \\ 0 & a'_4 & 0 & b'_4 \end{bmatrix} \\
 \Downarrow \\
 \begin{array}{l} \text{دستگاه مستقل اول} \\ \text{دستگاه مستقل دوم} \end{array} \begin{array}{l} \begin{bmatrix} b'_1 & c'_1 \\ a'_3 & b'_3 \end{bmatrix} \\ \begin{bmatrix} b'_2 & c'_2 \\ a'_4 & b'_4 \end{bmatrix} \end{array}
 \end{array}$$

شکل ۴-۴: یک مرحله از مرحله پیشرو از الگوریتم کاهش متناوب موازی روی یک دستگاه با چهار مجهول که منجر به ایجاد دو دستگاه مستقل می‌شود

۴-۲-۴ الگوریتم توماس موازی

رهیافت توماس موازی از این موضوع نشأت می‌گیرد که در مسائل با بیش از یک بعد، هر سطر از شبکه عددی می‌تواند یک دستگاه معادلات سه‌قطری تشکیل دهد که هر سطر یا هر دستگاه مستقل از دستگاه‌های دیگر می‌توانند حل شوند. در واقع رهیافت توماس موازی همان الگوریتم توماس کلاسیک است که به صورت جداگانه برای تعداد زیادی از دستگاه معادلات به‌طور موازی به کار برده می‌شود. در این روش وظیفه حل هر سطر یا هر دستگاه معادله سه‌قطری به یک نخ

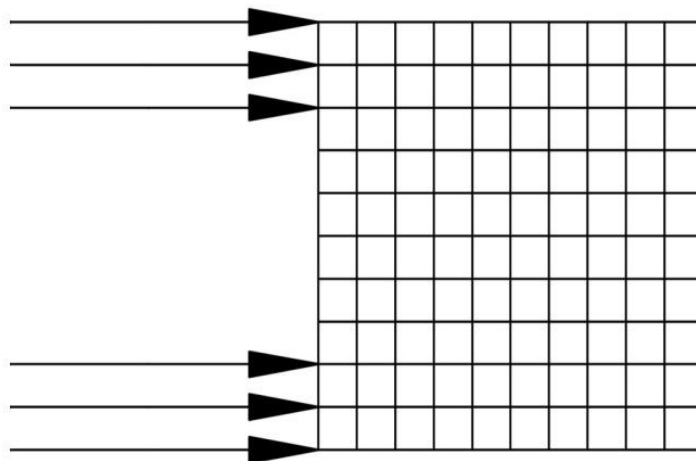
محاسباتی سپرده می‌شود. همان‌طور که قبلاً اشاره شد مزیت این روش این است که محدودیتی از نظر اندازه دستگاه وجود ندارد اما در مورد الگوریتم‌های کاهش متناوب و کاهش متناوب موازی حتماً اندازه دستگاه معادلات باید توانی از دو باشد تا این دو الگوریتم قادر به حل آنها باشند. در هر مسأله‌ای در نظر گرفتن چنین شرطی آسان نیست و در بسیاری از موارد همین محدودیت استفاده از این الگوریتم‌ها را ناممکن می‌سازد. بنابراین رهیافت توماس موازی می‌تواند جایگزین مناسبی برای این الگوریتم‌ها در مواجهه با مسائل گوناگون باشد.

همان‌طور که در شکل ۵-۵ مشخص است هر نخ محاسباتی می‌تواند یک دستگاه معادله را توسط الگوریتم توماس کلاسیک و مستقل از دستگاه‌های دیگر حل نماید و تعداد نخ‌های محاسباتی با توجه به ابعاد مسأله تعیین می‌شوند. هر نخ محاسباتی با استفاده از معادلات (۴-۱) تا (۴-۳) می‌تواند دستگاه مربوط به خود را حل نماید. مزیت این روش این است که از سرعت الگوریتم توماس بهره می‌برد و این بهره زمانی که چندین نخ به‌طور هم‌زمان شروع به حل با این الگوریتم می‌کنند، چند برابر می‌شود. دیگر مزیت این روش حل این است که پیچیدگی‌های دیگر الگوریتم‌های موازی را برای اجرا روی پردازنده گرافیکی ندارد. عملکرد این روش هرچه تعداد دستگاه معادلات بیشتر باشند، بیشتر می‌شود. پس از آن که تمام نخ‌ها محاسبات مربوط به خود را انجام دادند، در واقع دامنه محاسباتی یک‌بار حل شده است.

برای بررسی موثر بودن روش توماس موازی در پردازش موازی و قیاس سرعت حل و دقت حل آن روی پردازنده گرافیکی، در ادامه یک مسأله شبیه‌سازی جریان مانند جریان درون حفره با در پوش متحرک^۱ توسط سه الگوریتم موازی معرفی شده کاهش متناوب، کاهش متناوب موازی و توماس موازی برای حل این مسأله در نظر گرفته شده‌اند و نتایج با حل توسط الگوریتم توماس

¹ Lid-Driven Cavity Flow

کلاسیک روی پردازنده مرکزی قیاس شده‌اند و در نهایت تمام این نتایج با نتایج مرجع [25] مقایسه شده‌اند.



شکل ۴-۵: ساختار نخ‌های محاسباتی برای اجرای رهیافت توماس موازی روی یک شبکه عددی

۴-۳ شبیه سازی جریان درون حفره با استفاده از پیاده‌سازی معادلات

سه قطری روی پردازنده گرافیکی

یکی از پر استفاده‌ترین مسأله‌های شبیه‌سازی برای صحت‌سنجی و بررسی روش‌های جدید، مسأله جریان درون حفره است. تاکنون در مورد جریان درون حفره تحقیقات بسیاری منتشر شده و اثر هندسه‌های مختلف، سیالات مختلف، روش‌های مختلف حل و غیره روی این مسأله بررسی شده‌اند. برای مثال در مرجع [25] جیا و همکاران که یک مرجع شناخته شده برای این مسأله است، توسط روش چند شبکه‌ای به حل جریان درون حفره پرداختند. در این تحقیق تا رینولدز ۱۰۰۰۰ جریان را با استفاده از روش چند شبکه‌ای شبیه‌سازی و حل کرده‌اند. یا در مرجع [25]

ارتورک، بحث مفصلی در مورد تحقیقات منتشر شده در مورد جریان درون حفره در سال‌های اخیر انجام داده است یا پوچین پان [26] اثرات عبارت‌های غیر خطی را در حل مسأله وابسته به زمان توسط روش تابع جریان - تاوایی بررسی کرده است. همچنین برای نمونه کاربر پردازنده گرافیکی در حل مسأله جریان درون حفره، نانو کاردوس و بیکودو در سال ۲۰۰۹ [27] با روش تابع جریان- تاوایی^۱ جریان گذرا درون حفره را با استفاده از پردازنده گرافیکی گرافیکی تا رینولدز ۳۰۰۰ حل و بررسی کرده‌اند و به افزایش سرعت حداکثری ۱۵ در مقایسه یک پردازنده گرافیکی با یک پردازنده مرکزی و افزایش سرعت ۴ در مقایسه یک پردازنده گرافیکی با یک پردازنده مرکزی موازی شده در یک شبکه ۱۰۲۴×۱۰۲۴ رسیده‌اند. نمونه‌های دیگری از مطالعات روی جریان درون حفره در مراجع [28, 29, 30] قابل دسترس و مطالعه هستند.

در این پایان نامه نیز برای بررسی سرعت الگوریتم پیشنهادی توماس موازی و مقایسه با الگوریتم‌های موجود با استفاده از روش تابع جریان- تاوایی به حل جریان درون حفره به عنوان یک محک سرعت، پرداخته می‌شود تا قابلیت‌های الگوریتم توماس موازی بیشتر نمایان شود.

۴-۴ شبیه‌سازی جریان با روش تابع جریان - تاوایی

برای شبیه‌سازی جریان لزج تراکم ناپذیر باید معادلات ناویر استوکس (معادلات (۴-۸) الی (۴-۱۰)) روی شبکه محاسباتی گسسته شوند، دستگاه معادلات مربوطه تشکیل و سپس با اعمال روش مناسب حل شوند.

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \quad (۴-۸)$$

¹ Stream Function - Vorticity

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} + \frac{1}{\rho} \frac{\partial p}{\partial x} = v \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) \quad (9-4)$$

$$\frac{\partial v}{\partial t} + u \frac{\partial v}{\partial x} + v \frac{\partial v}{\partial y} + \frac{1}{\rho} \frac{\partial p}{\partial y} = v \left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right) \quad (10-4)$$

اما سروکار داشتن با معادلات (۸-۴) تا (۱۰-۴) به دلیل کوپل بودن با یکدیگر دشواری‌های مخصوص به خود را دارد. ولی در بعضی مسائل می‌توان از تعاریف و ابزارهای قدرتمندی چون تابع جریان و تاوایی برای ساده‌سازی معادلات ناویر استوکس استفاده کرد. تابع جریان طبق معادله (۴-۴) و تاوایی با رابطه (۴-۱۲) تعریف می‌شوند.

$$u = \frac{\partial \psi}{\partial y}, \quad v = -\frac{\partial \psi}{\partial x} \quad (11-4)$$

$$\omega = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} \quad (12-4)$$

اگر از معادله (۹-۴) نسبت به y و از معادله (۱۰-۴) نسبت به x مشتق گرفته شود (معادلات (۱۳-۴) و (۱۴-۴)) و این دو معادله از هم کم شوند، پس از یک‌سری ساده‌سازی معادله (۱۵-۴) با نام معادله پخشی-انتقالی تاوایی به دست می‌آید.

$$\frac{\partial}{\partial y} \left(\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} + \frac{1}{\rho} \frac{\partial p}{\partial x} = v \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) \right) \quad (13-4)$$

$$\frac{\partial}{\partial x} \left(\frac{\partial v}{\partial t} + u \frac{\partial v}{\partial x} + v \frac{\partial v}{\partial y} + \frac{1}{\rho} \frac{\partial p}{\partial y} = v \left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right) \right) \quad (14-4)$$

$$\frac{\partial \omega}{\partial t} + u \frac{\partial \omega}{\partial x} + v \frac{\partial \omega}{\partial y} = v \left(\frac{\partial^2 \omega}{\partial x^2} + \frac{\partial^2 \omega}{\partial y^2} \right) \quad (15-4)$$

همچنین اگر روابط (۱۱-۴) درون معادله (۱۲-۴) جایگزین شوند، رابطه بین تابع جریان و

تاوایی نیز به دست می آید (معادله ۴-۱۶).

$$\frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2} = -\omega \quad (۴-۱۶)$$

معادلات (۴-۱۵) و (۴-۱۶) معادلات اساسی روش تابع جریان - تاوایی هستند. این روش حل برای جریان سیالات تراکم ناپذیر کاربرد دارد و ویژگی آن این است که دو معادله برای تحلیل جریان در اختیار می گذارد و تعداد معادلات را از سه به دو عدد کاهش می دهد زیرا معادله پیوستگی خود به خود توسط تعریف تابع جریان ارضا می شود. نکته مهم دیگر از بین رفتن عبارت فشار در معادلات است که سهولت بیشتری در انجام محاسبات ایجاد می کند اما اگر نیاز به محاسبه فشار در یک مسأله باشد باید به طریق زیر عمل کرد که از معادله (۴-۹) نسبت به x و از معادله (۴-۱۰) نسبت به y مشتق گرفته شود و با هم جمع شوند، پس از ساده سازی معادله پوآسون فشار به شکل معادله (۴-۱۷) یا (۴-۱۸) ظاهر می شود.

$$\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} = -2\rho \left(\left(\frac{\partial^2 \psi}{\partial x^2} \right) \left(\frac{\partial^2 \psi}{\partial y^2} \right) - \left(\frac{\partial^2 \psi}{\partial xy} \right)^2 \right) \quad (۴-۱۷)$$

$$\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} = -2\rho \left(\left(\frac{\partial u}{\partial x} \right) \left(\frac{\partial v}{\partial y} \right) - \left(\frac{\partial u}{\partial y} \right) \left(\frac{\partial v}{\partial x} \right) \right) \quad (۴-۱۸)$$

فرم بی بعد تابع جریان و تاوایی به ترتیب به فرم روابط (۴-۱۹) و (۴-۲۰) تبدیل می شوند.

$$\psi^* = \frac{\psi}{u_\infty L} \quad (۴-۱۹)$$

$$\omega^* = \frac{\omega L}{u_\infty} \quad (۴-۲۰)$$

که L یک طول مشخص در هندسه مسأله است و u_∞ سرعت جریان آزاد می باشند. با توجه به روابط (۴-۱۹) و (۴-۲۰) معادلات اساسی تابع جریان - تاوایی به فرم بی بعد به ترتیب در روابط (۴-۲۱) و (۴-۲۲) آورده شده اند. همچنین روابط بی بعد فشار نیز به شکل (۴-۲۱) و (۴-۲۲)

هستند.

$$\frac{\partial^2 \psi^*}{\partial x^{*2}} + \frac{\partial^2 \psi^*}{\partial y^{*2}} = -\omega^* \quad (۱۹-۴)$$

$$\frac{\partial \omega^*}{\partial t^*} + u^* \frac{\partial \omega^*}{\partial x^*} + v^* \frac{\partial \omega^*}{\partial y^*} = \frac{1}{Re_\infty} \left(\frac{\partial^2 \omega^*}{\partial x^{*2}} + \frac{\partial^2 \omega^*}{\partial y^{*2}} \right) \quad (۲۰-۴)$$

$$\frac{\partial^2 p^*}{\partial x^{*2}} + \frac{\partial^2 p^*}{\partial y^{*2}} = 2 \left(\left(\frac{\partial^2 \psi^*}{\partial x^{*2}} \right) \left(\frac{\partial^2 \psi^*}{\partial y^{*2}} \right) - \left(\frac{\partial^2 \psi^*}{\partial x^* \partial y^*} \right)^2 \right) \quad (۲۱-۴)$$

$$\frac{\partial^2 p^*}{\partial x^{*2}} + \frac{\partial^2 p^*}{\partial y^{*2}} = -2 \left(\left(\frac{\partial u^*}{\partial x^*} \right) \left(\frac{\partial v^*}{\partial y^*} \right) - \left(\frac{\partial u^*}{\partial y^*} \right) \left(\frac{\partial v^*}{\partial x^*} \right) \right) \quad (۲۲-۴)$$

دیگر پارامترهای بی بعد استفاده شده در روابط (۲۳-۴) و (۲۴-۴) مشخص هستند.

$$t^* = \frac{tu_\infty}{L} \quad x^* = \frac{x}{L} \quad y^* = \frac{y}{L} \quad (۲۳-۴)$$

$$u^* = \frac{u}{u_\infty} \quad v^* = \frac{v}{u_\infty} \quad p^* = \frac{p}{\rho_\infty u_\infty^2} \quad Re = \frac{\rho_\infty u_\infty L}{\mu_\infty} \quad (۲۴-۴)$$

با استفاده از معادلات (۱۹-۴) تا (۲۲-۴) می توان یک جریان دوبعدی را حل و شبیه سازی کرد. اما

این روش در مسائل سه بعدی به علت تعریف تاوایی باعث پیچیده تر شدن محاسبات می شود و

معمولا در مسئل دو بعدی از این روش استفاده می شود. پس از حل معادلات (۱۹-۴) و (۲۰-۴) و

به دست آمدن مقادیر تاوایی و تابع جریان روی یک دامنه محاسباتی، می توان به راحتی توسط

روابط (۱۱-۴) مولفه های عمودی و افقی سرعت را محاسبه کرد. و در نهایت برای محاسبه مقدار

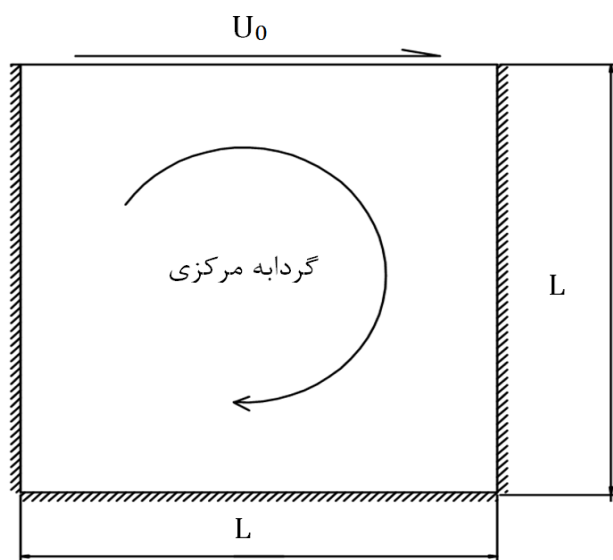
فشار یا از معادله (۲۱-۴) یا (۲۲-۴) می توان فشار را محاسبه کرد.

۴-۵ هندسه مسأله و گسسته سازی

مسأله جریان درون حفره هندسه ای مانند شکل ۴-۶ دارد. طول و عرض حفره با طول

مشخصه L و سرعت جریان آزاد و یا سرعت درپوش U_0 را داراست. در این قسمت تحلیل یک

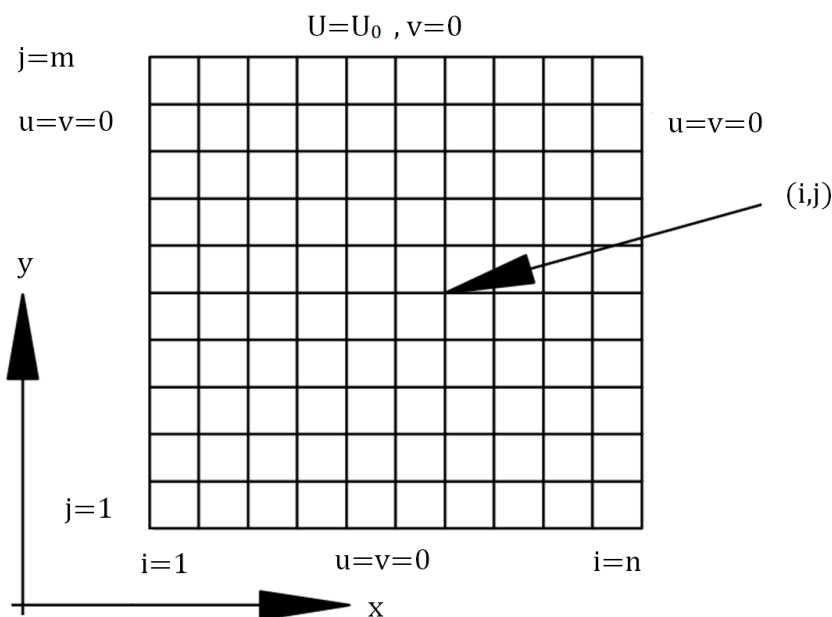
جریان پایدار مد نظر است و در معادله (۲۰-۴) عبارت مشتق زمان صفر می‌شود. ($\frac{\partial \omega^*}{\partial t^*} = 0$)



شکل ۴-۶: هندسه مسأله جریان درون حفره

در این مسأله مؤلفه‌های سرعت روی دیواره‌های ثابت صفر هستند. بنابراین با توجه به معادله (۱۱-۴) می‌توان نتیجه گرفت که تابع جریان روی این دیواره‌ها یک عدد ثابت است ($\psi = 0$). شرایط مرزی برای تاوایی نیز روی مرزهای $x = 0$ و $x = L$ و $y = 0$ به دلیل این که تابع جریان در راستای دیواره تغییر نمی‌کند از معادله (۱۹-۴) می‌توان بدست آورد. روی مرز متحرک هم شرط مرزی تاوایی با استفاده از معادله (۱۹-۴) به دست می‌آید.

برای حل معادلات، یک شبکه عددی همانند شکل ۴-۷ در نظر گرفته می‌شود و معادلات (۱۹-۴) و (۲۰-۴) با روش تفاضل محدود مرکزی گسسته می‌شوند. معادلات گسسته شده به ترتیب با (۲۵-۴) و (۲۶-۴) مشخص شده‌اند:



۷-۴: شبکه عددی تولید شده و شرایط مرزی برای مسأله جریان درون حفره

$$\frac{\psi_{i+1,j}^* - 2\psi_{i,j}^* + \psi_{i-1,j}^*}{\Delta x^{*2}} + \frac{\psi_{i,j+1}^* - 2\psi_{i,j}^* + \psi_{i,j-1}^*}{\Delta y^{*2}} = -\omega_{i,j}^* \quad (۲۵-۴)$$

$$\begin{aligned} u_{i,j}^* \frac{(\omega_{i+1,j}^* - \omega_{i-1,j}^*)}{2\Delta x^*} + v_{i,j}^* \frac{(\omega_{i,j+1}^* - \omega_{i,j-1}^*)}{2\Delta y^*} \\ = \left(\frac{1}{Re} \right) \left(\frac{\omega_{i+1,j}^* - 2\omega_{i,j}^* + \omega_{i-1,j}^*}{\Delta x^{*2}} + \frac{\omega_{i,j+1}^* - 2\omega_{i,j}^* + \omega_{i,j-1}^*}{\Delta y^{*2}} \right) \end{aligned} \quad (۲۶-۴)$$

با مرتب کردن معادلات (۲۵-۴) و (۲۶-۴) به ترتیب، معادلات (۲۷-۴) و (۲۸-۴) به دست

می‌آیند که به شکل سه قطری مرتب شده‌اند.

$$\begin{aligned} \psi_{i+1,j}^* + \psi_{i-1,j}^* - 2(1 + \beta^2)\psi_{i,j}^* \\ = -\Delta x^{*2}\omega_{i,j}^* - \beta^2(\psi_{i,j+1}^* + \psi_{i,j-1}^*) \end{aligned} \quad (۲۵-۴)$$

$$\begin{aligned} \omega_{i+1,j}^* \left(\frac{1}{Re * \Delta x^2} + \frac{u_{i,j}}{2\Delta x} \right) + \omega_{i-1,j}^* \left(\frac{-1}{Re * \Delta x^2} - \frac{u_{i,j}}{2\Delta x} \right) - 2 \left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2} \right) \omega_{i,j}^* \\ = \omega_{i,j+1}^* \left(\frac{1}{Re * \Delta y^2} + \frac{-v_{i,j}}{2\Delta y} \right) + \omega_{i,j-1}^* \left(\frac{-1}{Re * \Delta y^2} + \frac{v_{i,j}}{2\Delta y} \right) \end{aligned} \quad (۲۶-۴)$$

که $\beta = \frac{\Delta x}{\Delta y}$ و i و j اندیس‌های گره‌های شبکه محاسباتی هستند. برای این تحقیق $\beta = 1$ در نظر گرفته شده است.

پردازنده گرافیکی مورد استفاده در این تحقیق، کارت گرافیک GeForce GTX 580 با معماری فرمی و با قابلیت محاسباتی ۲ می‌باشد. این پردازنده دارای ۵۱۲ هسته پردازشی است و دارای ۴ گیگا بایت حافظه سراسری است. پردازنده مرکزی مورد استفاده نیز Intel Core i7 990x می‌باشند. این نکته قابل ذکر است که برای مقایسه از یک کارت گرافیک و پردازنده مرکزی هم‌نسل و هم‌رده استفاده شده تا مقایسه صحیح و عادلانه باشد. همچنین از نسخه ۴ میان‌افزار کودا برای برنامه‌نویسی روی پردازنده گرافیکی استفاده شده است. در برنامه‌نویسی با میان‌افزار کودا اندازه بلوک و ابعاد آن قبل از اجرای برنامه باید مشخص باشد همچنین تعداد بلوک‌های درون یک شبکه در هر راستا از ۶۵۵۳۵، و تعداد نخ‌های محاسباتی درون یک بلوک نیز در هر راستا از ۱۰۲۴ نمی‌تواند بیشتر باشد. در این مسأله اندازه شبکه محاسباتی از 64×64 تا 1024×1024 متغیر است و با توجه به ساختار کرنل و اندازه شبکه، اندازه هر بلوک از ۶۴ تا ۲۵۶ مقداردهی شده است. در پیاده‌سازی الگوریتم‌های کاهش متناوب و کاهش متناوب موازی از حافظه سراسری برای ذخیره‌سازی متغیرها استفاده شده و فقط برای ذخیره‌سازی چند متغیر فرعی از حافظه مشترک استفاده شده است. دلیل این امر هم همان‌طور که قبلاً اشاره شد، محدودیت حافظه مشترک می‌باشد. همچنین در الگوریتم توماس موازی نمی‌توان از حافظه مشترک در مسائل با ابعاد بسیار بزرگ استفاده کرد به این دلیل که حافظه مشترک دارای این خصیصه است که فقط نخ‌های محاسباتی همان بلوک به آن دسترسی دارند و اگر اندازه هر دستگاه معادله از ۱۰۲۴ بیشتر شود به ناچار تعداد بلوک‌ها، حداقل دو بلوک خواهند بود و نخ‌های محاسباتی یک بلوک به مقادیر بلوک دیگر دسترسی ندارند. این موضوع که نیمی از متغیرها به مقدار عددی نیم دیگر دسترسی نداشته

باشند در الگوریتم توماس پذیرفته نیست. اما برای مسائلی که اندازه هر دستگاه مستقل کمتر از ۱۰۲۴ باشد استفاده از حافظه مشترک برای ذخیره‌سازی متغیرها امکان‌پذیر است.

در مرجع [8] به این مطلب اشاره شده است که، اگر استفاده از پردازنده گرافیکی فقط برای شتاب‌دهی به بخشی از محاسبات استفاده شده، باید زمان انتقال اطلاعات از حافظه میزبان به حافظه دستگاه را محاسبه نمود و بعد در مورد میزان مؤثر بودن استفاده از پردازش موازی قضاوت کرد. اما اگر تمام مراحل حل از ابتدا روی پردازنده گرافیکی انجام شده باشند، در انتها برای مشاهده نتایج باید اطلاعات را از دستگاه به میزبان انتقال داد، در این صورت نیازی به در نظر گرفتن زمان انتقال نیست. در این تحقیق نیز تمام مراحل حل روی دستگاه صورت گرفته و بنابراین در پایان مراحل حل، زمان انتقال متغیرهایی نظیر سرعت، تاوایی و غیره از دستگاه به میزبان در نظر گرفته نشده است.

مسأله دیگری که در این تحقیق بررسی شده، ارزیابی و تأثیر به کار بردن مفهوم دسترسی هم‌مکان به حافظه در زمان حل است. یکی از موارد مهمی که روی عملکرد و بازده یک برنامه پردازش موازی تأثیر بسزایی دارد توجه به دسترسی هم‌مکان به حافظه سراسری است که اهمیت آن با توجه به معماری پردازنده گرافیکی و ارتباط با حافظه سراسری توضیح داده شد. در بخش بعد نتایج تحلیل جریان درون حفره توسط پردازنده گرافیکی و مرکزی ارائه شده است.

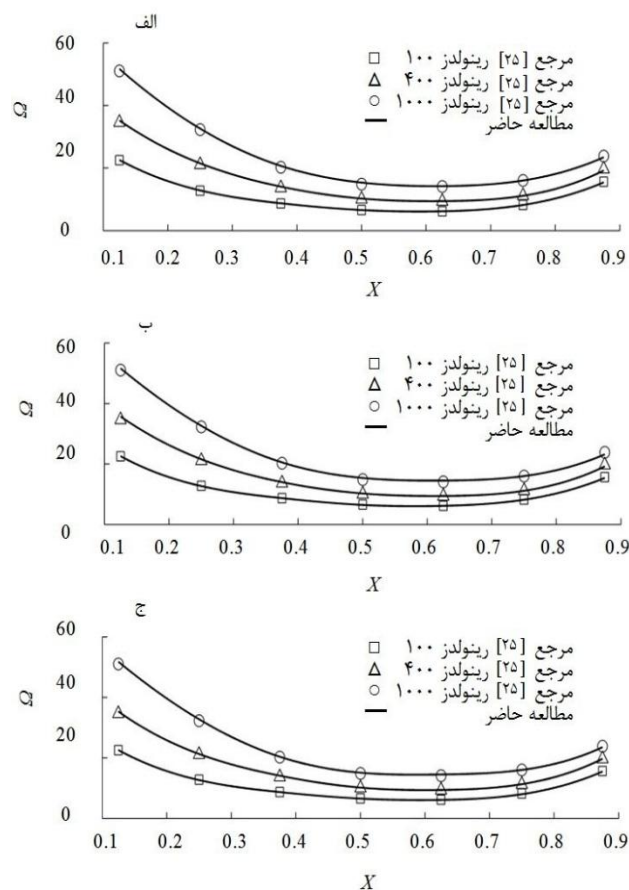
۴-۶ نتایج

در شکل شماره ۴-۸ برای نمونه به ترتیب مقادیر تاوایی در مرز متحرک حاصل از نتایج حل با رهیافت توماس موازی و الگوریتم‌های کاهش متناوب، کاهش متناوب موازی، در سه عدد رینولدز

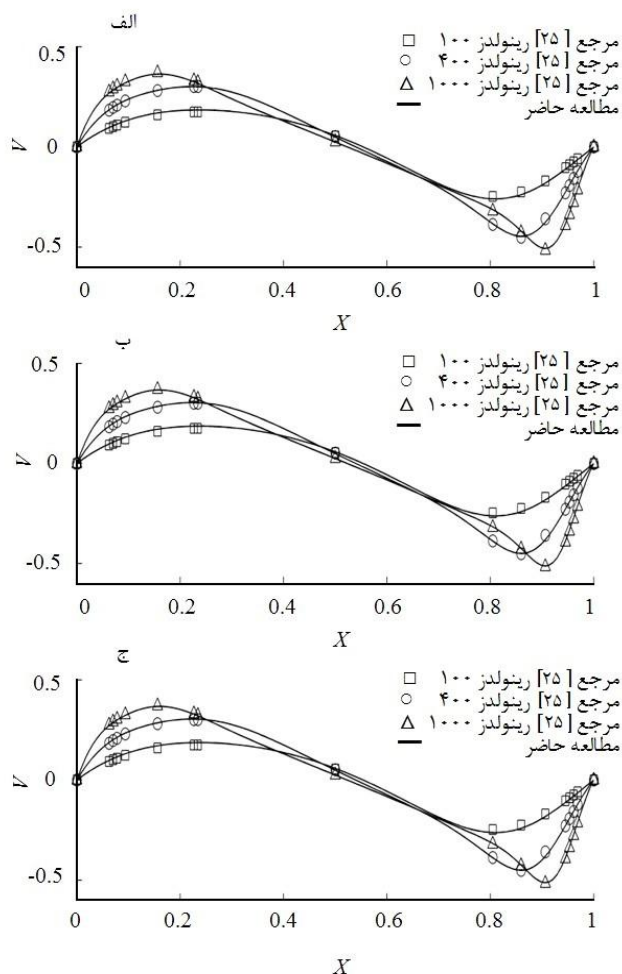
۱۰۰، ۴۰۰ و ۱۰۰۰ با مقادیر گزارش شده در مرجع [25] مقایسه شده‌اند. در این قسمت تمام نتایج به شکل بی بعد گزارش شده‌اند. در شکل ۴-۹ مؤلفه عمودی سرعت روی خط $Y = 0.5$ حاصل از حل مسأله توسط رهیافت توماس و دو الگوریتم کاهش متناوب و کاهش متناوب موازی گزارش و با مرجع [25] مقایسه شده است. به همین ترتیب برای مؤلفه افقی سرعت نیز روی خط $X = 0.5$ در شکل ۴-۱۰ مورد مقایسه قرار گرفته‌اند. در شکل‌های ۴-۸ تا ۴-۱۰ مقادیر برای نتایج حل با یک شبکه 128×128 ارائه شده‌اند.

در شکل‌های ۴-۱۱ و ۴-۱۲ به ترتیب کانتورهای تابع جریان و کانتورهای تاوایی را برای رینولدزهای ۴۰۰، ۱۰۰ و ۱۰۰۰ توسط الگوریتم توماس موازی نمایش داده شده‌اند. در کانتورهای تاوایی روی بعضی از منحنی‌ها مقادیر تاوایی نیز گزارش شده‌اند. در شکل ۴-۱۳ زمان حل مسأله جریان درون حفره توسط اجرای رهیافت توماس، الگوریتم کاهش متناوب و کاهش متناوب موازی روی پردازنده گرافیکی با زمان حل مسأله توسط پردازنده مرکزی و اجرای الگوریتم توماس کلاسیک مقایسه شده‌اند. در این مقایسه اندازه شبکه محاسباتی از 64×64 تا 1024×1024 تغییر می‌کنند. زمان‌های اندازه‌گیری شده در رینولدز ۱۰۰۰ و برای ۱۰۰۰ تکرار به دست آمده‌اند. به دلیل خطی بودن نسبت تعداد تکرار با زمان حل هم برای پردازنده مرکزی و هم برای پردازنده گرافیکی می‌توان از این تعداد تکرار برای مقایسه استفاده کرد. با توجه به شکل ۴-۱۳ همان‌طور که انتظار می‌رفت در تعداد شبکه‌های کم، پردازنده مرکزی سریع‌تر از الگوریتم‌های موازی عمل می‌کند اما از اندازه 256×256 این موضوع بر عکس می‌شود. اما در مورد رهیافت توماس موازی می‌توان مشاهده کرد که در تمام اندازه‌ها هم از الگوریتم‌های موازی و هم از الگوریتم توماس کلاسیک سریع‌تر عمل می‌کند. در شکل ۴-۱۴ که افزایش سرعت حل مسأله روی پردازنده گرافیکی نسبت به پردازنده مرکزی نمایش داده شده، مشخص است که رهیافت توماس موازی با الگوی دسترسی هم‌مکان به

حافظه، به حداکثر افزایش سرعت $38/45$ برابری دست یافته است. همچنین نسبت به الگوریتم کاهش متناوب و کاهش متناوب موازی نیز به ترتیب حداکثر افزایش سرعت $8/7$ و $7/4$ برابری را از خود نشان می دهد. این افزایش سرعت، نشانگر برتری رهیافت توماس موازی نسبت به این الگوریتم های حل دستگاه معادلات سه قطری با رویکرد موازی است.



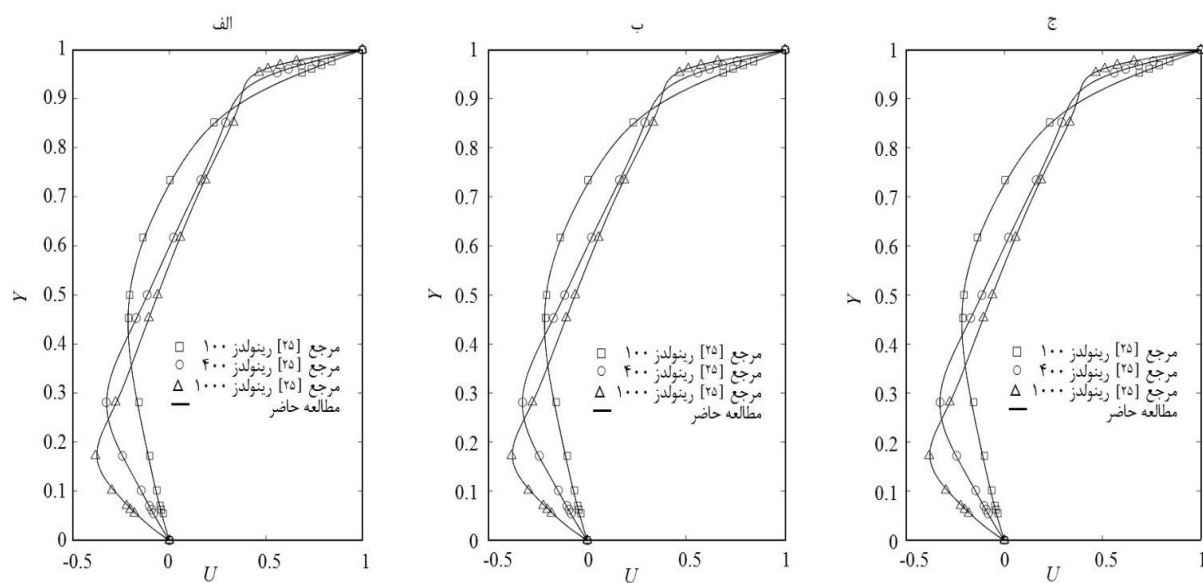
شکل ۴-۸: نمایش مقدار تاوایی در مرز متحرک برای سه عدد رینولدز 100، 400 و 1000 حاصل از نتایج:
(الف) رهیافت توماس موازی (ب) الگوریتم کاهش متناوب (ج) الگوریتم کاهش متناوب موازی



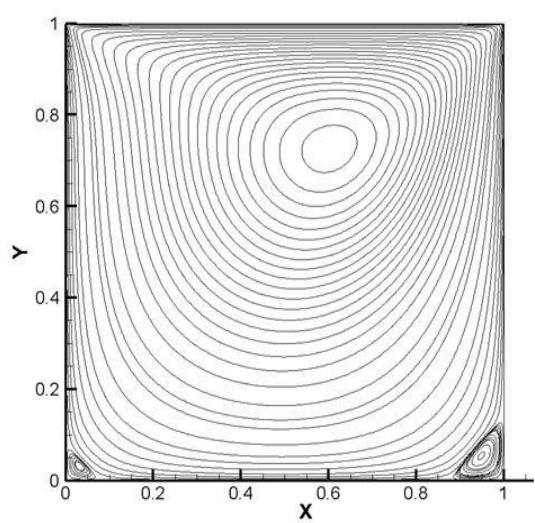
شکل ۴-۹: نمایش مؤلفه عمودی سرعت روی خط $Y = 0.5$ برای سه عدد رینولدز ۱۰۰، ۴۰۰ و ۱۰۰۰ حاصل از نتایج: (الف) رهیافت توماس موازی (ب) الگوریتم کاهش متناوب (ج) الگوریتم کاهش متناوب موازی

در مورد اهمیت دسترسی هم‌مکان به حافظه توضیح داده شد و ذکر شد که در صورت رعایت این موضوع عملکرد و سرعت پردازش به طرز چشم‌گیری افزایش پیدا می‌کند. برای بررسی تأثیر در سرعت پردازش برای الگوریتم تماس موازی به هر دو صورت برنامه‌نویسی انجام شده و در شکل ۱۶ افزایش سرعت رهیافت توماس موازی اجرا شده روی پردازنده گرافیکی، با دسترسی‌های هم‌مکان و غیر هم‌مکان نسبت به الگوریتم توماس کلاسیک اجرا شده روی پردازنده مرکزی،

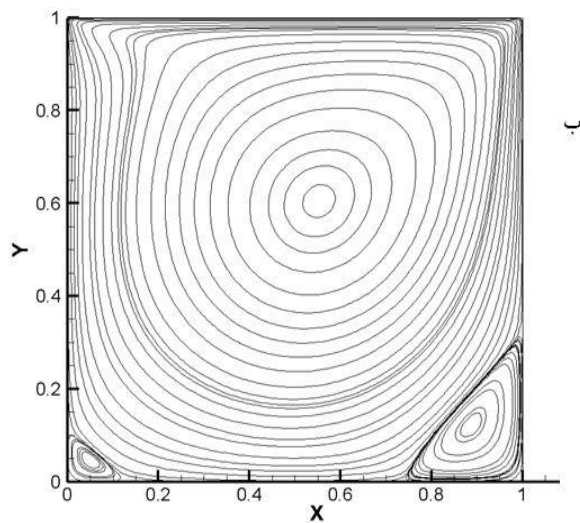
مقایسه شده‌اند. مشخص است که در هر تعداد گره محاسباتی سرعت الگوریتم با دسترسی هم‌مکان حدوداً دو برابر الگوریتم با دسترسی غیر هم‌مکان است. بنابراین رعایت این اصل تغییر بزرگی در سرعت پردازش ایجاد می‌کند. رعایت دسترسی هم‌مکان به حافظه در مورد الگوریتم‌های کاهش متناوب و کاهش متناوب موازی مطرح نیست و نمی‌توان این بهینه‌سازی را روی آنها اجرا کرد.



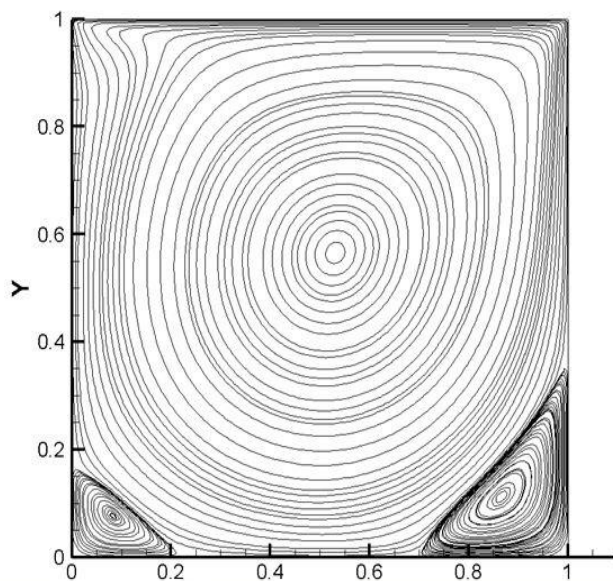
شکل ۴-۱۰: نمایش مؤلفه افقی سرعت روی خط $X = 0.5$ برای سه عدد رینولدز 100، 400 و 1000 حاصل از نتایج: (الف) رهیافت توماس موازی (ب) الگوریتم کاهش متناوب (ج) الگوریتم کاهش متناوب موازی



الف



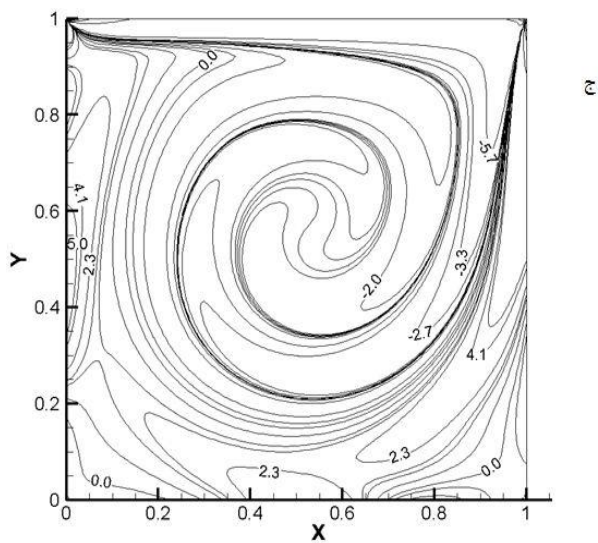
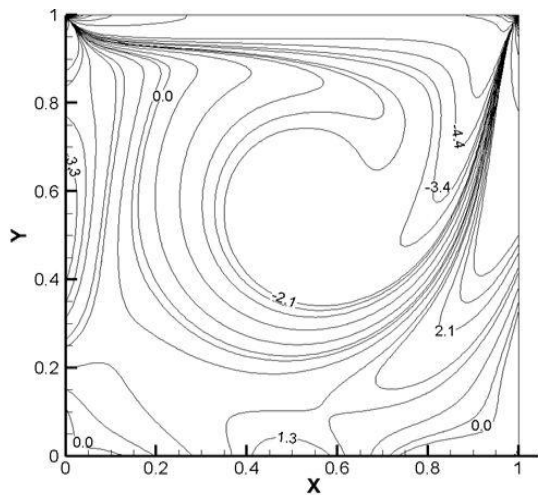
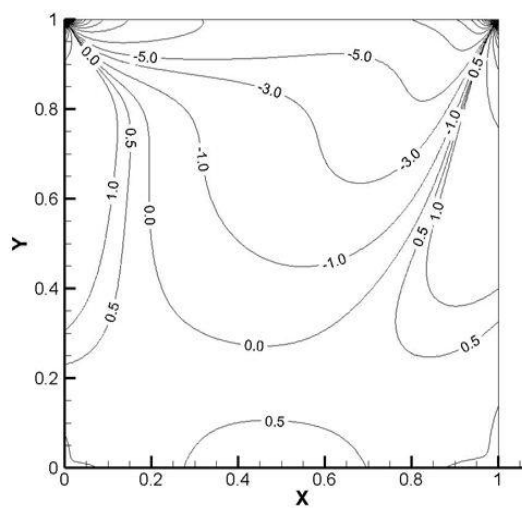
ب



ج

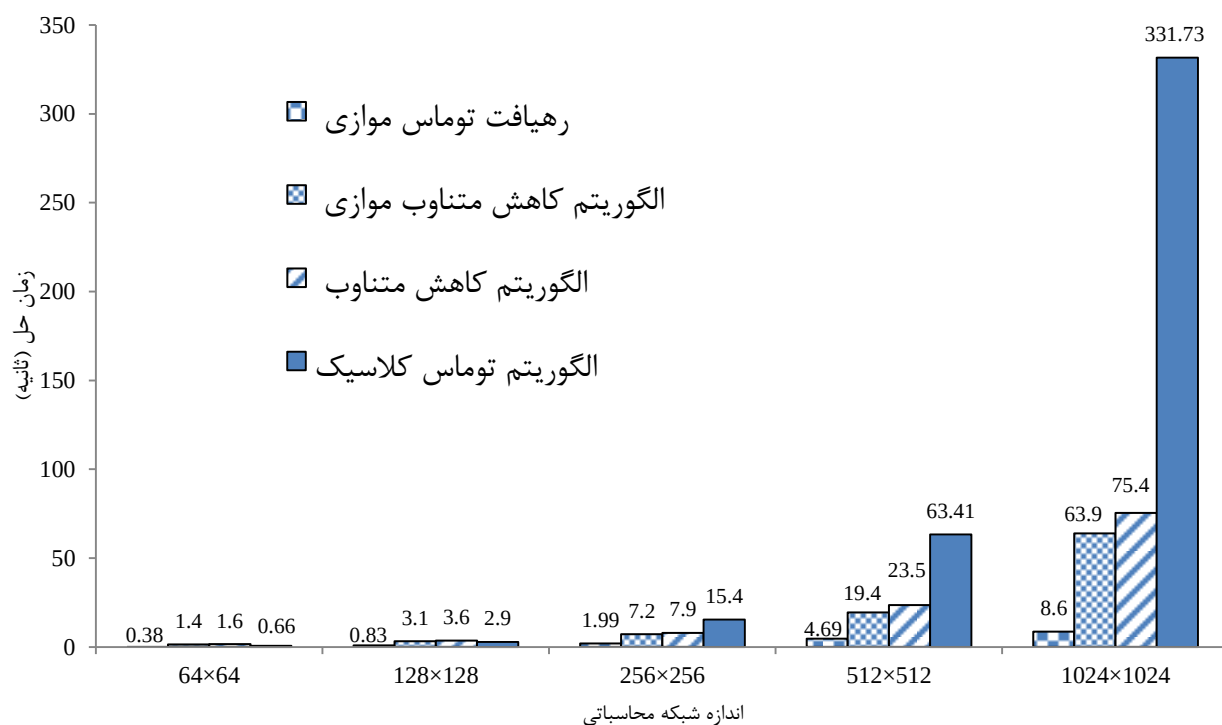
شکل ۴-۱۱: کانتورهای تابع جریان در جریان درون حفره با الگوریتم توماس موازی

برای رینولدزهای (الف) ۱۰۰ (ب) ۴۰۰ (ج) ۱۰۰۰

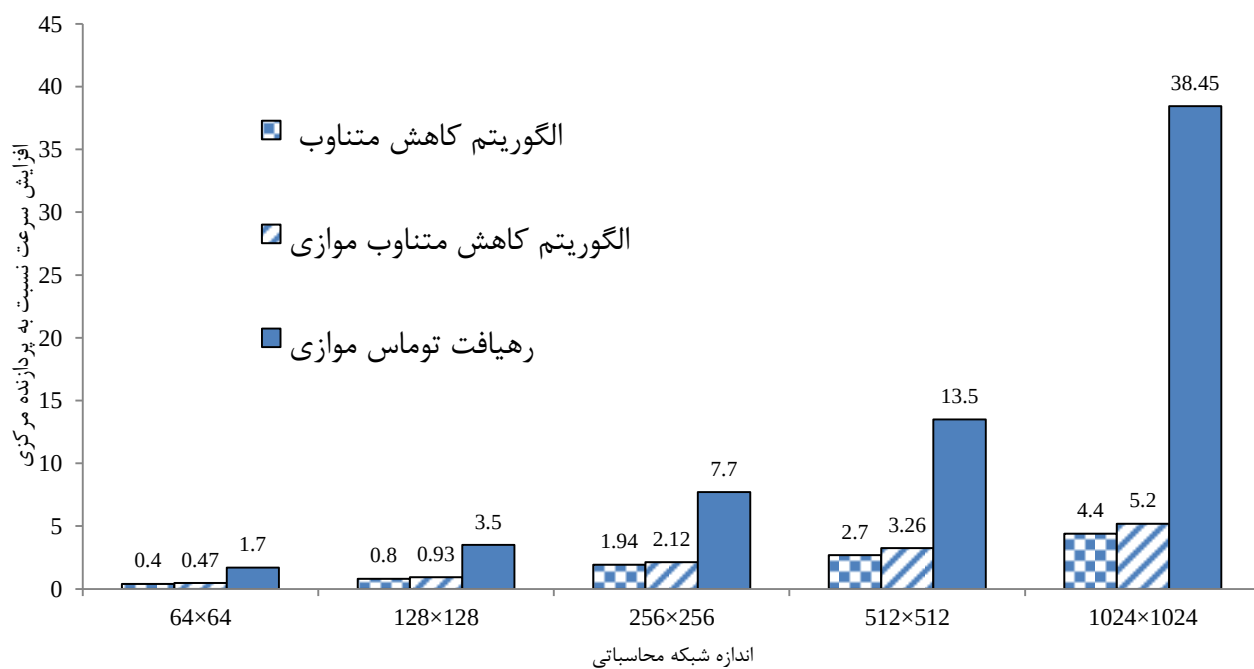


شکل ۴-۱۲: کانتورهای تاوایی در جریان درون حفره با الگوریتم توماس موازی

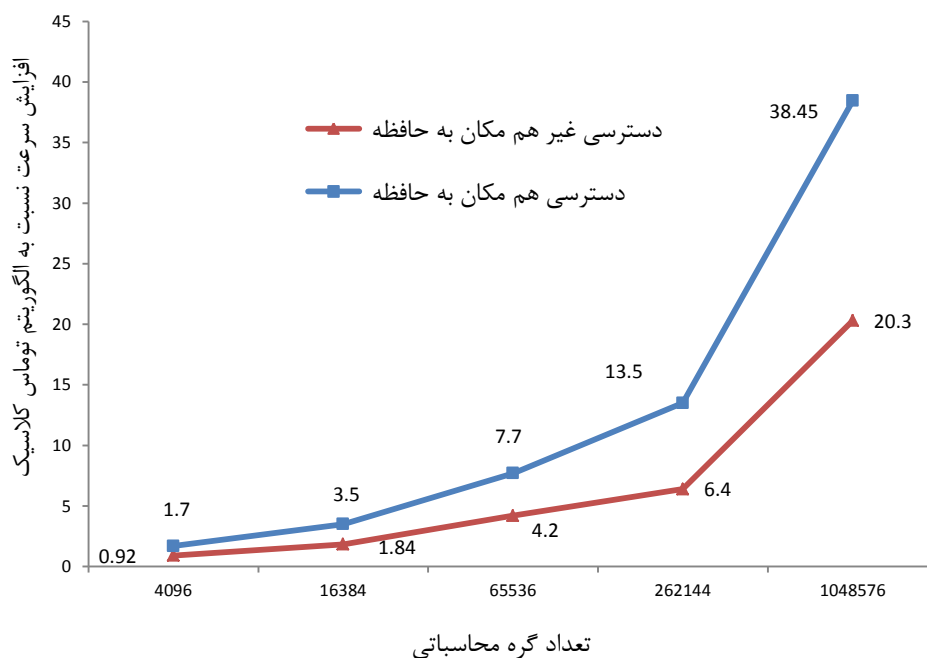
برای رینولدزهای (الف) ۱۰۰ (ب) ۴۰۰ (ج) ۱۰۰۰



شکل ۴-۱۳: نمایش زمان حل الگوریتم‌های مختلف برای رینولدز ۱۰۰۰ در طی ۱۰۰۰ تکرار



شکل ۴-۱۴: نمایش افزایش سرعت زمان حل الگوریتم‌های موازی برای رینولدز ۱۰۰۰



شکل ۴-۱۵: مقایسه افزایش سرعت پردازش موازی در برابر پردازنده مرکزی در حالت دسترسی هم‌مکان و غیرهم‌مکان به حافظه برای تعداد گره محاسباتی مختلف در رینولدز ۱۰۰۰

در این تحقیق دو الگوریتم کاهش متناوب و کاهش متناوب موازی برای اجرا روی پردازنده‌های گرافیکی معرفی شدند و همچنین بر پایه الگوریتم توماس کلاسیک، این ایده مطرح شد که می‌توان دستگاه معادلات مربوط به هر سطر محاسباتی را مستقل از دیگر دستگاه‌ها حل کرد و این نکته منجر به پیشنهاد رهیافت توماس موازی شد. با پیاده‌سازی این الگوریتم‌ها روی پردازنده گرافیکی، اجرا و حل مسأله جریان درون حفره، نتایج حاصل از حل توسط پردازنده گرافیکی با نتایج حل روی پردازنده مرکزی و مرجع [25] از تطابق قابل قبولی برخوردار بودند. افزایش سرعت زمان حل برای اندازه‌های مختلف شبکه از 64×64 تا 1024×1024 نسبت به زمان حل پردازنده مرکزی، برای الگوریتم کاهش متناوب بین $0/4$ تا $4/4$ به دست آمد و برای الگوریتم کاهش متناوب موازی از $0/47$ تا $5/2$ متغیر بود. اما رهیافت توماس موازی نیز با دو نوع رویکرد متفاوت از نوع دسترسی به حافظه سراسری مورد بررسی قرار گرفت. در زمانی که دسترسی از نوع هم‌مکان صورت گرفت

حداکثر ۳۸/۵ و در دسترسی غیر هم‌مکان تا ۲۰/۳ برابر، افزایش سرعت حاصل می‌شد. در دسترسی هم‌مکان در تمام اندازه‌های شبکه محاسباتی رهیافت توماس موازی در حل مسأله افزایش سرعت نشان داد اما هنگامی که نوع دسترسی به حافظه بهینه نبود، سرعت حل چشم‌گیر نبود و حتی در اندازه شبکه ۶۴×۶۴ کاهش سرعت نسبت به زمان حل پردازنده مرکزی ثبت شد. بنابراین مشخص شد که در پردازش با پردازنده‌های گرافیکی حتماً باید به دسترسی حافظه توجه ویژه‌ای شود تا نتایج قابل قبولی از لحاظ پاسخ‌گویی زمانی در بر داشته باشد.

مسأله دیگری که می‌تواند مورد توجه قرار گیرد این است که روش اجرای رهیافت توماس موازی به‌گونه‌ای است که برای دیگر روش‌های حل نیز کاربرد خواهد داشت. زمانی که بتوان هر سطر محاسباتی را مستقل از دیگر سطرها حل نمود این ایده به ذهن متبادر می‌شود که محاسبات هر سطر را یک نخ محاسباتی می‌تواند به عهده گیرد خواه دستگاه معادله تشکیل شده سه‌قطری باشد خواه پنج‌قطری باشد. در نتیجه این روش به کارگیری نخ‌ها وسعت و کاربرد بیشتری نسبت به الگوریتم‌های موازی معرفی شده می‌تواند داشته باشد. از دیگر مزیت‌های رهیافت توماس موازی، محدود نبودن اندازه دستگاه معادلات به 2^n ذکر شد و در بسیاری از مسائل محقق شدن این شرط امکان پذیر نخواهد بود. این توانایی می‌تواند گستره کاربرد این رهیافت را افزون کند.

فصل پنجم :

شبیه‌سازی جریان روی

استوانه با الگوریتم

توماس موازی

۵-۱ مقدمه

با توجه به اهمیت عملی و نظری بسیار بالای عبور جریان گذرنده از اجسام حجیم به خصوص اجسام با سطح مقطع گرد، این زمینه از مطالعات به میزان زیادی توجه محققان را به خود جلب کرده است. مثال‌های رایج در صنعت و استفاده از این نوع جریان، جریان در مبدل‌های حرارتی، استفاده از استوانه‌های باریک به عنوان پروب‌های اندازه‌گیری همانند سنسورها و غیره است. در طول سالیان تحقیقات قابل توجهی در مورد درباره جنبه‌های مختلفی مانند نوع جریان، نوع سیال و انتقال حرارت در این نوع جریان منتشر شده است [32, 33, 34]. بر خلاف تحقیقات بسیار در مورد جریان حول استوانه‌های دایروی، در مورد جریان عبوری از استوانه با دیگر مقاطع هندسی اندک مقطع مربعی یا مثلثی با توجه به کاربرد آن‌ها اطلاعات مشابهی وجود ندارد. با این حال در سال‌های اخیر تلاش‌هایی در حیطه جریان حول سیلندر مربعی به چشم می‌خورد.

ترکی و همکاران [35] یک مطالعه عددی جهت تحلیل میدان جریان غیر دائم و مشخصات انتقال حرارت در یک کانال افقی با یک سیلندر مربعی گرم شده ارائه دادند. مطالع برای دو نسبت انسداد $1/8$ و $1/4$ و اعداد رینولدز مختلف ارائه شده و نشان داده شده که انتقال حرارت با انسداد رابطه‌ای هر چند اندک دارد و روابطی ارائه کرده‌اند. شارما و اسواران [36] نیز مطالعه دقیقی روی جریان و انتقال حرارت روی سیلندر استوانه‌ای انجام دادند. بروئر و همکاران [37] بررسی دقیقی روی جریان آرام گذرنده از روی یک استوانه مربعی با دو روش شبکه بولتزمن و روش حجم محدود پرداختند و از رینولدز $0/5$ تا 300 را با هر دو روش بررسی و نتایج را نیز صحت سنجی کرده‌اند. یون و همکاران [38] جریان عبوری از استوانه را تحت زوایای چرخش مختلف استوانه بررسی کرده‌اند و رابطه بین عدد رینولدز، زوایای چرخش و نیروی وارد شده بر استوانه مورد بررسی قرار گرفتند. یوسفی فرد و همکاران [39] با یک روش تکراری مرتبه بالا جریان عبوری از استوانه مربعی

را شبیه‌سازی کرده‌اند و نتایج را با مرجع [38] مقایسه کرده‌اند و نتایج از تطابق خوبی برخوردار بوده است. اما همه این مطالعات که معرفی شدند در واقع بحث اصلی روی جریان گذرا بوده و یک پدیده مهم که در جریان روی اجسام حجیم مانند استوانه رخ می‌دهد، گردابه‌های فون کارمن هستند که هم از لحاظ انتقال حرارتی هم از نظر تحلیل سازه‌ای مهم هستند و در مطالعات مختلف مشخص شده که از حدود رینولدز ۶۰ به بالا این پدیده مشاهده می‌شود. تا قبل از آن گردابه‌های ثابتی پشت استوانه به وجود می‌آیند و نوسانی مشاهده نمی‌شود بنابراین مهمترین و جذاب‌ترین بخش، رینولدزهای ۶۰ به بالا در این مسأله هستند و بیشتر مورد بررسی قرار می‌گیرند.

اما تعداد محدودی از محققان پردازش موازی با پرازنده گرافیکی نیز به بررسی جریان عبوری از اجسام حجیم پرداخته‌اند از جمله مرجع [13] که پیشتر معرفی شد. در این تحقیق جریان عبوری از استوانه دایروی و ایرفویل با الگوریتم کاهش متناوب پرداخته‌اند. و همچنین کدا و لین [40] که اخیرا با روش شبکه بولتزمن جریان درون حفره، جریان درون لوله و جریان روی استوانه را شبیه‌سازی کردند.

علاوه بر روش شبکه بولتزمن، روش‌های کلاسیک دیگری که برای شبیه‌سازی این‌گونه از مسائل وابسته به زمان وجود دارد که می‌توان آن‌ها را در دو دسته طبقه‌بندی کرد، روش صریح^۱ برای گسسته‌سازی و روش ضمنی^۲. در مسائل وابسته به زمان گام زمانی^۳ حل نقش مهمی در پایداری حل دارند. در روش صریح با توجه سهولت در امر گسسته‌سازی، پیاده‌سازی و حل اما پایداری عددی شکننده‌ای دارند و در نتیجه گام‌های زمانی باید بسیار کوچک انتخاب شوند. اما روش‌های ضمنی از پایداری مناسبی برخوردارند ولی در امر پیاده‌سازی ممکن است برنامه‌نویس

¹ Explicit

² Implicit

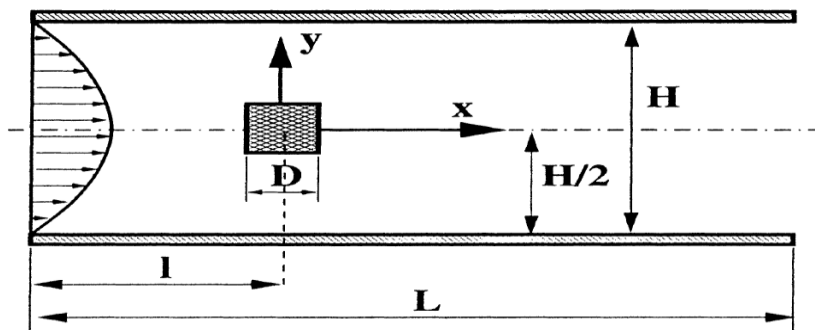
³ Time Step

دچار زحمت شود. در مسائل یک بعدی اگر از روش ضمنی برای گسسته‌سازی استفاده شود یک دستگاه سه قطری حاصل می‌شود که با روش‌های مرسوم قابل حل هستند اما اگر مسأله مورد بررسی دو یا سه بعدی باشد دستگاه معادلات حاصل پنج یا هفت قطری می‌شوند. برای حل چنین دستگاه‌هایی راه‌های مختلفی وجود دارند و یکی از آن‌ها، روش ضمنی جهت متناوب^۱ (ADI) است. در ادامه پیاده‌سازی روش ضمنی جهت متناوب و حل و شبیه‌سازی جریان روی استوانه مربعی با این روش روی پردازنده گرافیکی شرح داده می‌شود.

۵-۲ شبیه‌سازی جریان روی استوانه مربعی، معادلات حاکم و پیاده‌سازی

روی پردازنده گرافیکی

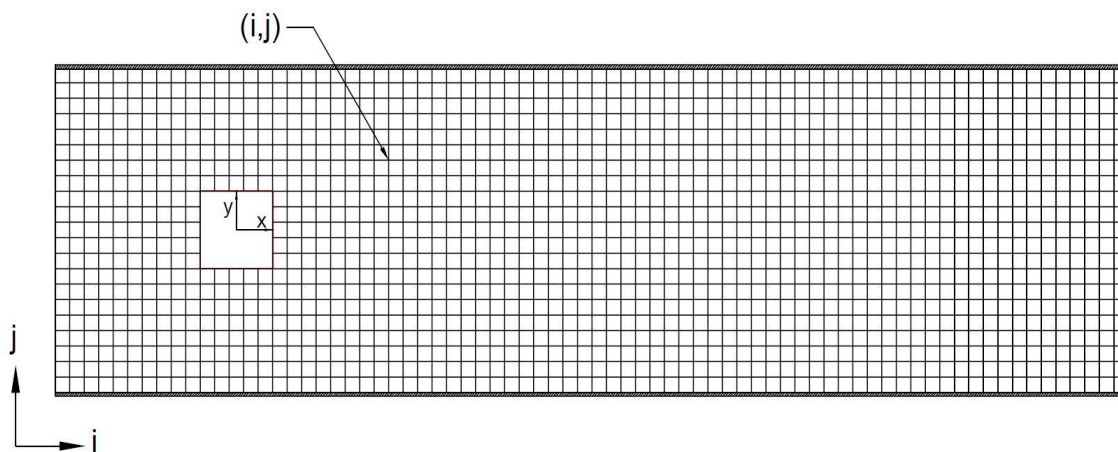
همانطور که قبلاً گفته شد جریان عبوری از استوانه مربعی یک مسأله نسبتاً پرکاربرد در صنعت است. شکل ۵-۱ شمای کلی و هندسه مسأله را نمایان می‌کند. در این شکل یک استوانه با مقطع مربع که مبدأ مختصات روی مرکز آن قرار گرفته است و توسط دو دیواره محدود شده دیده می‌شود.



شکل ۵-۱: هندسه جریان روی استوانه مربعی

¹ Alternating Direction Implicit

در شکل ۱-۵ طول کل کانال L ، عرض استوانه D ، عرض کانال H ، و طول ابتدا تا مرکز استوانه l یا طول ورودی است. همچنین نسبت عرض کانال به عرض استوانه نسبت انسداد^۱ گفته می‌شود. $(\beta = \frac{D}{H})$. برای گسسته‌سازی روش تفاضل محدود مرکزی انتخاب می‌شود و یک شبکه عددی یکنواخت مانند شکل ۲-۵ در نظر گرفته می‌شود.



شکل ۲-۵: شبکه عددی یکنواخت برای حل جریان روی استوانه مربعی

همانطور که گفته شد برای گسسته‌سازی حالت ضمنی در نظر گرفته می‌شود و برای حل دستگاه معادلات روش ضمنی جهت متناوب در نظر گرفته می‌شود. روش ضمنی جهت متناوب با توجه به تعداد بعد، مسأله و گام زمانی را به همان تعداد تقسیم می‌کند یعنی اگر مسأله دو بعدی باشد معادلات به دو بخش تقسیم می‌شوند و کل دامنه محاسباتی در یک راستا (برای مثال در راستای x) به شکل سه‌قطری یک بار حل می‌شوند و در واقع حل نصف یک گام زمانی پیشروی داشته و در مرحله دوم با حل معادلات در کل دامنه محاسباتی به شکل سه‌قطری در راستای y نصف دیگر یک گام زمانی طی می‌شود و با انجام این دو مرحله در واقع حل یک گام زمانی به جلو رفته است. خاصیت و مزیت این روش در سه‌قطری بودن معادلات است که توسط الگوریتم توماس

¹ Blockage

که یک الگوریتم سریع و بهینه است می‌تواند حل شوند و در نتیجه یک سرعت خوب در حل ایجاد می‌شود.

تلفیق این الگوریتم با الگوریتم توماس موازی که در فصل ۴ معرفی شد می‌تواند یک ابزار کارآمد و بسیار سریع در حل مسائل مختلف باشد.

۵-۲-۱ معادلات حاکم

برای شروع معادله پخششی - جابجایی تاوایی بازنویسی می‌شود (معادله (۵-۱)). در معادله (۵-۲) روش ضمنی جهت متناوب اعمال شده و مقادیر تاوایی در راستای x به صورت ضمنی و در راستای y به صورت صریح گسسته شده‌اند. با حل این معادله که به شکل سه قطری است، باعث می‌شود مقادیر تاوایی در گام زمانی $n + \frac{1}{2}$ به دست‌آید مانند لایه دوم در شکل ۵-۳.

$$\frac{\partial \omega^*}{\partial t^*} + u^* \frac{\partial \omega^*}{\partial x^*} + v^* \frac{\partial \omega^*}{\partial y^*} = \frac{1}{Re_\infty} \left(\frac{\partial^2 \omega^*}{\partial x^{*2}} + \frac{\partial^2 \omega^*}{\partial y^{*2}} \right) \quad (۵-۱)$$

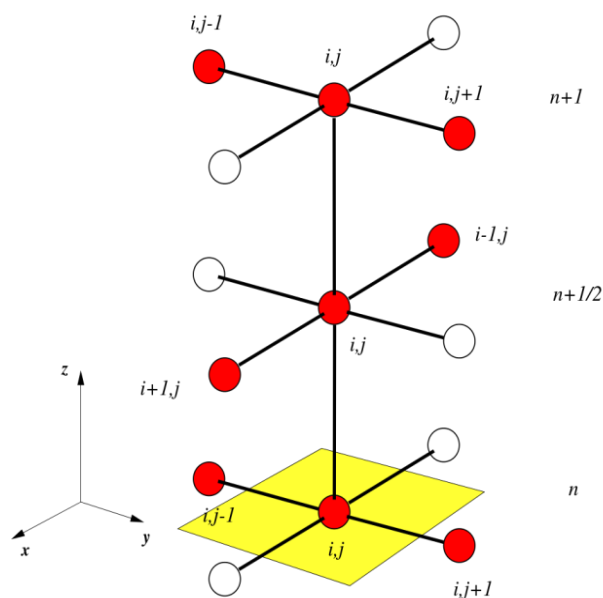
$$\begin{aligned} & \frac{\left(\omega_{i+1,j}^{*n+\frac{1}{2}} - \omega_{i-1,j}^{*n} \right)}{\frac{\Delta t^*}{2}} + u_{i,j}^{*n} \frac{\left(\omega_{i+1,j}^{*n+\frac{1}{2}} - \omega_{i-1,j}^{*n+\frac{1}{2}} \right)}{2\Delta x^*} + v_{i,j}^{*n} \frac{\left(\omega_{i,j+1}^{*n} - \omega_{i,j-1}^{*n} \right)}{2\Delta y^*} \\ &= \frac{1}{Re} \left(\frac{\omega_{i+1,j}^{*n+\frac{1}{2}} - 2\omega_{i,j}^{*n+\frac{1}{2}} + \omega_{i-1,j}^{*n+\frac{1}{2}}}{\Delta x^{*2}} \right. \\ & \quad \left. + \frac{\omega_{i,j+1}^{*n} - 2\omega_{i,j}^{*n} + \omega_{i,j-1}^{*n}}{\Delta y^{*2}} \right) \end{aligned} \quad (۵-۲)$$

$$\begin{aligned}
& \frac{\left(\omega_{i+1,j}^{*n+1} - \omega_{i-1,j}^{*n+\frac{1}{2}}\right)}{\frac{\Delta t^*}{2}} + u_{i,j}^{*n} \frac{\left(\omega_{i+1,j}^{*n+\frac{1}{2}} - \omega_{i-1,j}^{*n+\frac{1}{2}}\right)}{2\Delta x^*} + v_{i,j}^{*n} \frac{\left(\omega_{i,j+1}^{*n+1} - \omega_{i,j-1}^{*n+1}\right)}{2\Delta y^*} \\
& = \frac{1}{Re} \left(\frac{\omega_{i+1,j}^{*n+\frac{1}{2}} - 2\omega_{i,j}^{*n+\frac{1}{2}} + \omega_{i-1,j}^{*n+\frac{1}{2}}}{\Delta x^{*2}} \right. \\
& \quad \left. + \frac{\omega_{i,j+1}^{*n+1} - 2\omega_{i,j}^{*n+1} + \omega_{i,j-1}^{*n+1}}{\Delta y^{*2}} \right) \tag{۳-۵}
\end{aligned}$$

در گام دوم با استفاده از معادله (۳-۵) که بر عکس معادله (۲-۵)، مقادیر تلاوایی در راستای y به صورت ضمنی و در راستای x به صورت صریح گسسته شده‌اند و با حل این معادله در واقع کل دامنه محاسباتی یک گام زمانی به جلو حرکت کرده و تمام مقادیر در زمان $n + 1$ محاسبه شده هستند. به همین ترتیب تا زمان تعیین شده می‌توان حل را ادامه داد تا پروسه به پایان برسد. البته در مورد مقادیر تابع جریان به علت این که یک معادله بیضوی است، باید به صورت تکراری حل شود. که می‌توان هم به شکل سه‌قطری ($LSOR^1$) (معادله (۴-۵)) و هم به شکل نقطه به نقطه ($PSOR^2$) (معادله (۵-۵)) به حل پرداخت و مقادیر مورد نظر را بدست آورد. در معادله (۵-۵) پارامتر k نشان‌دهنده شماره تکرار حل و n گام زمانی حل است.

¹ Line Successive Over Relaxation

² Point Successive Over Relaxation



شکل ۳-۵: نمایش مراحل روش ضمنی جهت متغیر

$$\begin{aligned} \psi_{i+1,j}^{*k+1} + \psi_{i-1,j}^{*k+1} - 2(1 + \beta^2)\psi_{i,j}^{*k+1} \\ = -\Delta x^{*2}\omega_{i,j}^{*n+1} - \beta^2(\psi_{i,j+1}^{*k} + \psi_{i,j-1}^{*k}) \end{aligned} \quad (۴-۵)$$

$$\begin{aligned} \psi_{i,j}^{*k+1} = \frac{1}{2(1 + \beta^2)} (\Delta x^{*2}\omega_{i,j}^{*n+1} + \beta^2(\psi_{i,j+1}^{*k} + \psi_{i,j-1}^{*k+1}) + \psi_{i+1,j}^{*k} \\ + \psi_{i-1,j}^{*k+1}) \end{aligned} \quad (۵-۵)$$

با مرتب سازی معادلات (۲-۵) و (۳-۵) به شکل سه قطری، به ترتیب معادلات (۶-۵) و (۷-۵)

به دست می آیند.

$$\begin{aligned} -\frac{1}{2}\left(\frac{1}{2}c_x + d_x\right)\omega_{i-1,j}^{*n+\frac{1}{2}} + (1 + d_x)\omega_{i,j}^{*n+\frac{1}{2}} + \frac{1}{2}\left(\frac{1}{2}c_x - d_x\right)\omega_{i+1,j}^{*n+\frac{1}{2}} \\ = D_x \end{aligned} \quad (۶-۵)$$

$$\begin{aligned} -\frac{1}{2}\left(\frac{1}{2}c_y + d_y\right)\omega_{i,j-1}^{*n+1} + (1 + d_y)\omega_{i,j}^{*n+1} + \frac{1}{2}\left(\frac{1}{2}c_y - d_y\right)\omega_{i,j+1}^{*n+1} \\ = D_y \end{aligned} \quad (۷-۵)$$

در معادلات (۶-۵) و (۷-۵) داریم:

$$c_x = u \frac{\Delta t}{\Delta x} \quad , \quad c_y = v \frac{\Delta t}{\Delta y}$$

$$d_x = \frac{1}{Re} \frac{\Delta t}{\Delta x^2} \quad , \quad d_y = \frac{1}{Re} \frac{\Delta t}{\Delta y^2}$$

همچنین سمت راست معادلات (۵-۶) و (۵-۷) به شکل (۵-۸) و (۵-۹) تعریف می‌شوند.

$$D_x = \frac{1}{2} \left(\frac{1}{2} c_y + d_y \right) \omega_{i,j-1}^{*n} + (1 - d_y) \omega_{i,j}^{*n} + \frac{1}{2} \left(-\frac{1}{2} c_y - d_y \right) \omega_{i,j+1}^{*n} \quad (۵-۸)$$

$$D_y = \frac{1}{2} \left(\frac{1}{2} c_x + d_x \right) \omega_{i-1,j}^{*n+\frac{1}{2}} + (1 - d_x) \omega_{i,j}^{*n+\frac{1}{2}} + \frac{1}{2} \left(-\frac{1}{2} c_x - d_x \right) \omega_{i+1,j}^{*n+\frac{1}{2}} \quad (۵-۹)$$

و در نهایت می‌توان معادلات (۵-۶) و (۵-۷) به شکل زیر بازنویسی می‌شوند.

$$A_x \omega_{i-1,j}^{*n+\frac{1}{2}} + B_x \omega_{i,j}^{*n+\frac{1}{2}} + C_x \omega_{i+1,j}^{*n+\frac{1}{2}} = D_x \quad (۵-۱۰)$$

$$A_y \omega_{i,j-1}^{*n+1} + B_y \omega_{i,j}^{*n+1} + C_y \omega_{i,j+1}^{*n+1} = D_y \quad (۵-۱۱)$$

استراتژی حل به این شکل است که برای اجرای مرحله اول و حل معادله‌ای مانند (۵-۱۰) ناحیه

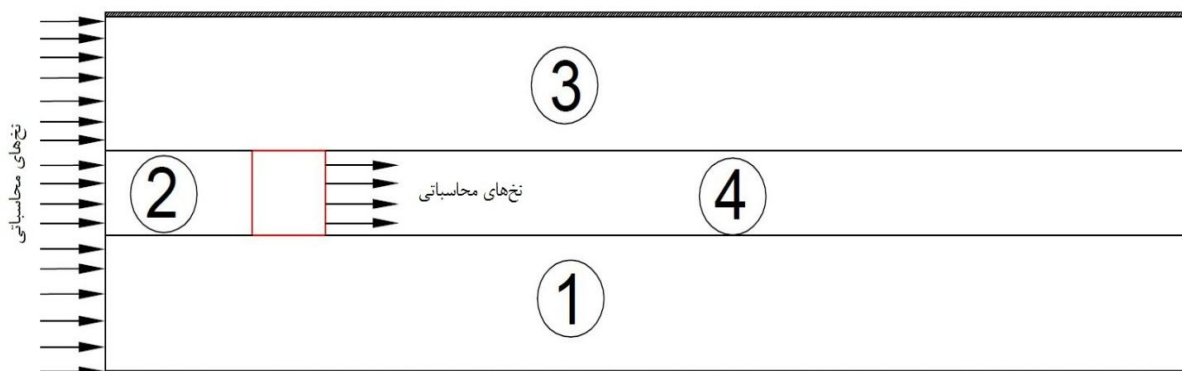
محاسباتی به چهار بخش تقسیم می‌شود و هر سطر یک دستگاه معادله سه‌قطری است که با

الگوریتم توماس حل می‌شوند. در پردازنده مرکزی هر سطر شکل (۵-۴) به ترتیب حل و پردازش

می‌شوند. برای پیاده‌سازی و حل روی پردازنده گرافیکی آرایش نخ‌های محاسباتی به شکل (۵-۴) در

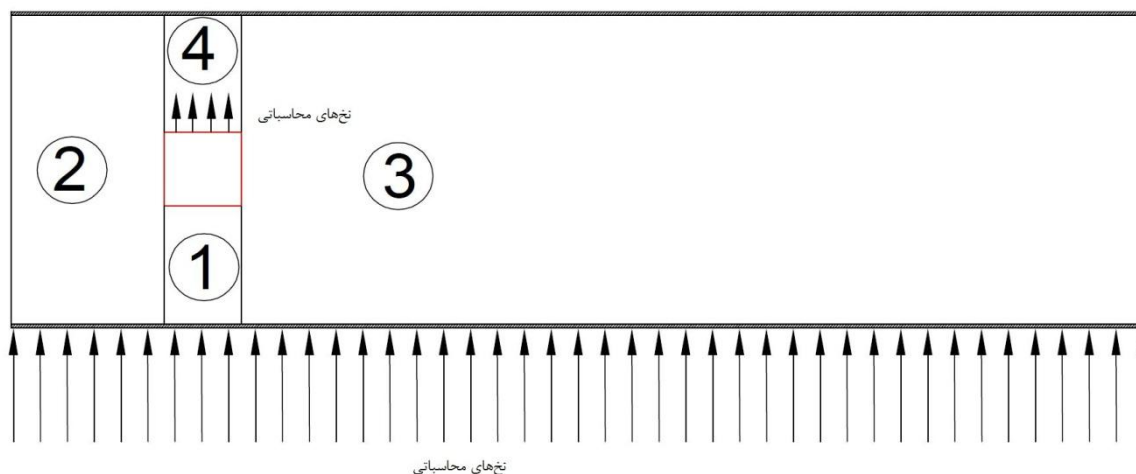
می‌آیند. هر نخ محاسباتی یک سطر محاسباتی را برای پردازش در دست می‌گیرد و هر نخ یک

دستگاه معادله سه‌قطری را با الگوریتم توماس حل می‌کند.



شکل ۴-۵: آرایش نخ‌های محاسباتی در مرحله اول اعمال روش ضمنی جهت متناوب

در مرحله دوم حل باید در راستای y و طبق معادله (۵-۱۱) دستگاه معادلات حل شوند تا یک گام زمانی کامل شود و نخ‌ها در پردازنده گرافیکی آرایشی همچون شکل ۵-۵ پیدا می‌کنند. در مرحله دوم سترسی به حافظه سراسری و متغیرها از نوع هم‌مکان خواهد بود و یک حالت ایده‌آل برای پردازش موازی با پردازنده گرافیکی است.



شکل ۵-۵: آرایش نخ‌های محاسباتی در مرحله دوم اعمال روش ضمنی جهت متناوب

۵-۲-۲ شرایط مرزی

شرایط مرزی در مسأله جریان روی استوانه به این صورت است که تابع جریان روی دیواره استوانه در رینولدزهای پایین و کمتر از ۶۰ چون جریان نوسان ندارد و مقدار دبی یکسانی از بالا و پایین استوانه می‌گذرند برابر صفر می‌شود ($\psi = 0$). اما در رینولدزهای بالای ۶۰ چون دبی عبوری از بالا و پایین استوانه یکسان نیست مقدار تابع جریان روی تمام استوانه باید در هر گام زمانی طبق معادله (۵-۵) محاسبه شود. مقدار تابع جریان در دیواره‌های کانال نیز طبق این استدلال محاسبه می‌شوند که اگر هر دیواره را یک خط جریان در نظر بگیریم تفاضل بین دو دیواره دبی ورودی به کانال خواهد بود.

مقدار تابع جریان در ورودی کانال با توضیحات مرجع [40] می‌توان از رابطه (۵-۱۲) و توضیحات زیر استفاده کرد که با توجه به این که در ورودی کانال مولفه عمودی سرعت صفر است ($v = 0$) پس در نتیجه $\frac{\partial \psi}{\partial x} = 0$. با استفاده از تفاضل محدود پیشرو می‌توان به رابطه (۵-۱۲) رسید.

$$\psi_{1,j} = \frac{1}{3}(4\psi_{2,j} - \psi_{3,j}) \quad (۵-۱۲)$$

با همین استدلال که در جروجی نیز اگر طول کانال به اندازه کافی طولانی باشد که نوسانات و گردابه‌ها ناپدید شوند و مستهلک شوند می‌توان گفت ($v = 0$) پس با استفاده از یک تفاضل محدود پسرو داریم:

$$\psi_{NX,j} = \frac{1}{3}(4\psi_{NX-1,j} - \psi_{NX-2,j}) \quad (۵-۱۳)$$

تاوایی نیز روی دیواره‌های بالا و پایین کانال به ترتیب با روابط (۵-۱۴) و (۵-۱۵) محاسبه می‌شوند. همچنین مقدار تاوایی روی دیواره‌های استوانه نیز با این روابط قابل محاسبه‌اند. برای مثال روی دیواره عمودی سمت راست استوانه اگر اندیس شمارنده در راستای x با IRW مشخص شود مقدار تاوایی از رابطه (۵-۱۶) محاسبه می‌شوند. ذکر این نکته الزامی است که با توجه به عبارت آخر رابطه (۵-۱۶) اگر مقدار تابع جریان روی یک دیواره مانند دیواره کانال، عددی ثابت باشد، این عبارت حذف یا در واقع صفر می‌شود. در مورد دیواره بالای استوانه نیز می‌توان از رابطه (۵-۱۷) بهره برد.

$$\omega_{i,1} = \frac{2(\psi_{i,1} - \psi_{i,2})}{\Delta y^2} \quad (۵-۱۴)$$

$$\omega_{i,NY} = \frac{2(\psi_{i,NY} - \psi_{i,NY-1})}{\Delta y^2} \quad (۵-۱۵)$$

$$\omega_{IRW,j} = \frac{2(\psi_{IRW,j} - \psi_{IRW+1,j})}{\Delta x^2} + \frac{\psi_{IRW,j+1} - 2\psi_{IRW,j} + \psi_{IRW,j-1}}{\Delta y^2} \quad (۵-۱۶)$$

$$\omega_{i,JUW} = \frac{2(\psi_{i,JUW} - \psi_{i,JUW+1})}{\Delta y^2} + \frac{\psi_{i+1,j} - 2\psi_{i,j} + \psi_{i-1,j}}{\Delta x^2} \quad (۵-۱۷)$$

مقدار تاوایی در مرز خروجی نیز از دو طریق قابل محاسبه است، یا با فرض $\frac{\partial \omega}{\partial x} = 0$ که منجر به رابطه (۵-۱۸) می‌شود. یا شبیه رابطه (۵-۱۶) از رابطه بین تاوایی و تابع جریان استفاده شود که رابطه (۵-۱۹) به دست می‌آید. در مرز ورودی هم تاوایی به مانند دیگر مرزها قابل محاسبه است (رابطه (۵-۲۰)).

$$\omega_{NX,j} = \frac{1}{3}(4\omega_{NX-1,j} - \omega_{NX-2,j}) \quad (۵-۱۸)$$

$$\omega_{NX,j} = \frac{2(\psi_{NX,j} - \psi_{NX-1,j})}{\Delta x^2} + \frac{\psi_{NX,j+1} - 2\psi_{NX,j} + \psi_{NX,j-1}}{\Delta y^2} \quad (۵-۱۹)$$

$$\omega_{1,j} = \frac{2(\psi_{1,j} - \psi_{2,j})}{\Delta x^2} + \frac{\psi_{1,j+1} - 2\psi_{1,j} + \psi_{1,j-1}}{\Delta y^2} \quad (۵-۲۰)$$

۵-۳ نتایج

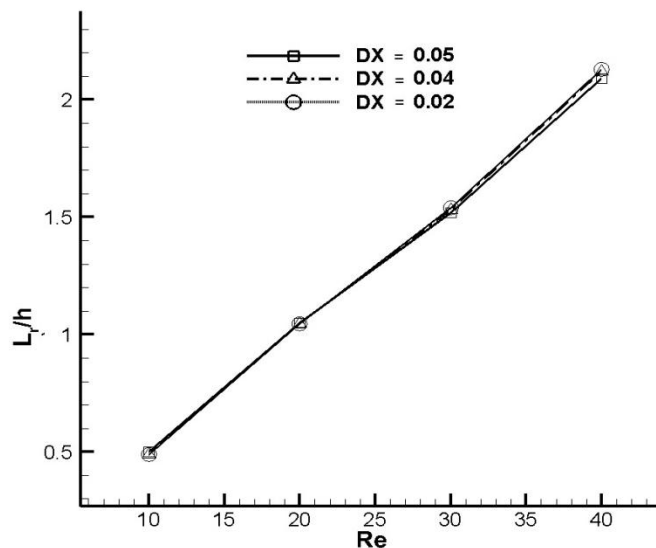
در این قسمت برای اطمینان از صحت نتایج حاصل از شبیه‌سازی جریان روی استوانه توسط الگوریتم توماس موازی روی پردازنده گرافیکی، در دو رژیم جریان بررسی می‌شوند، جریان پایا و ناپایا. در جریان پایا مواردی چون الگوی جریان، طول گردابه‌های پشت استوانه یا طول بازچرخش^۱ مورد بررسی قرار می‌گیرند. اما در جریان ناپایا مواردی چون نیروی درگ، عدد اشتروهال، الگوی جریان و غیره که اهمیت بیشتری دارند مورد مطالعه قرار می‌گیرند و در نهایت بحث‌های مربوط به بهینه‌سازی و مقایسه زمان حل با پردازنده‌های گرافیکی انجام می‌شود.

۵-۳-۱ بررسی استقلال از شبکه

با توجه به این که برای بررسی افزایش سرعت پردازش در قیاس با سرعت پردازنده مرکزی سه اندازه شبکه مختلف ۰/۰۵، ۰/۰۴ و ۰/۰۲ در نظر گرفته شده است، و برای بررسی پارامترهای مختلف با نتایج گزارش شده در مراجع مختلف و انتخاب بهترین اندازه شبکه برای بررسی، ابتدا استقلال از شبکه بررسی می‌شود که مشخص شود که آیا بین نتایج حاصل از شبکه‌های مختلف اختلافی وجود دارد یا خیر؟

در شکل ۵-۶ برای جریان پایا طول گردابه‌های ایجاد شده پشت استوانه مربعی برای سه اندازه شبکه ۰/۰۲، ۰/۰۴ و ۰/۰۵ اندازه‌گیری شده و مشخص است که نتایج این سه تفاوت زیادی ندارند و برای ادامه کار و به‌دست آوردن نتایج، اندازه شبکه ۰/۰۴ در نظر گرفته شده است.

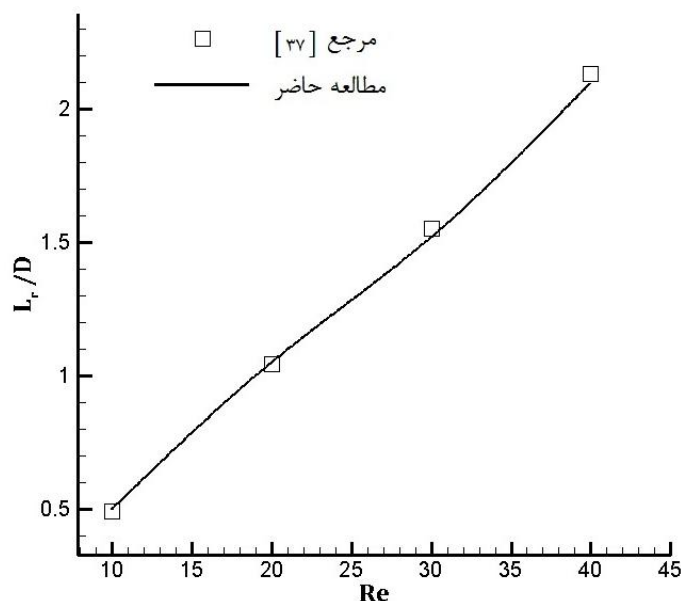
¹ Recirculation



شکل ۵-۶: بررسی طول گردابه‌های ایجاد شده پشت استوانه در جریان پایا در رینولدزهای مختلف برای بررسی استقلال از شبکه

۵-۳-۲ جریان پایا

در جریان پایا که در رینولدزهای کمتر از ۶۰ اتفاق می‌افتد، جریان الگویی ثابت داشته و گردابه چسبیده به پشت سیلندر ایجاد می‌شوند. یکی مشخصه‌هایی برای این دسته از رینولدزها می‌توان بررسی کرد طول گردابه ایجاد شده است. در شکل ۵-۷ طول گردابه در رینولدزهای ۱۰ تا ۴۰ اندازه‌گیری شده و با نتایج مرجع [37] که با دو روش متفاوت (حجم محدود و شبکه بولتزمن) مقایسه شده‌اند. همچنین در شکل ۵-۸ برای بررسی الگوی جریان و خطوط تابع جریان در رینولدزهای ۱، ۲۰ و ۴۰ نمایش داده شده‌اند. شکل‌های ۵-۷ و ۵-۸ با استفاده از یک شبکه محاسباتی یکنواخت با اندازه 200×100 به دست آمده‌اند. اما در مرجع [37] از شبکه غیریکنواخت و حدود ۶۰ گره روی سیلندر استفاده شده، طبیعتاً به علت اینکه تعداد گره‌های محاسباتی روی سیلندر با هم یکسان نیستند در رینولدزهای بالاتر کمی انحراف و عدم تطابق در شکل ۵-۷ دیده می‌شود.



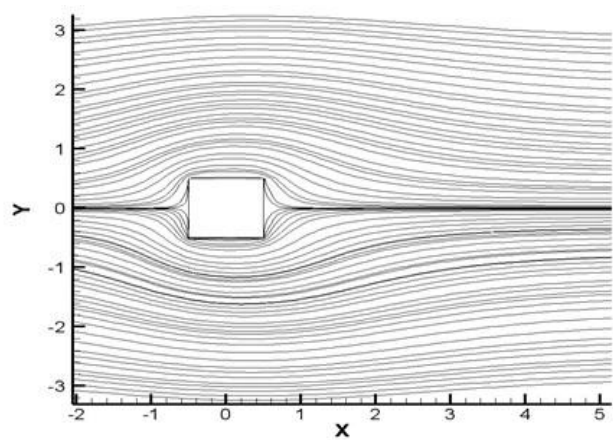
شکل ۵-۷: مقایسه طول گردابه‌های ایجاد شده در پشت سیلندر در رینولدزهای متفاوت از ۱۰ تا ۴۰ و مقایسه با نتایج گزارش شده در مرجع [37]

با توجه به شکل ۵-۸ مشاهده می‌شود که در $Re \leq 1$ یک جریان خزشی شکل می‌گیرد و هیچ‌گونه جدایشی رخ نمی‌دهد. خطوط جریان کاملاً محیط استوانه را در بر می‌گیرند. طبق مطالعات دروکویچ [33] در $Re \cong 5$ جدایش رخ می‌دهد و گردابه‌هایی ایجاد می‌شوند که با افزایش رینولدز بر طول آن‌ها افزوده می‌شود. در جریان‌های خزشی^۱ با بالا رفتن رینولدز تا قبل از جدایش نیروی درگ ویسکوز^۲ هم افزایش می‌یابد تا زمانی که گردابه ایجاد شود و بعد از آن نیروی درگ فشاری^۳ با افزایش رینولدز افزایش می‌یابد.

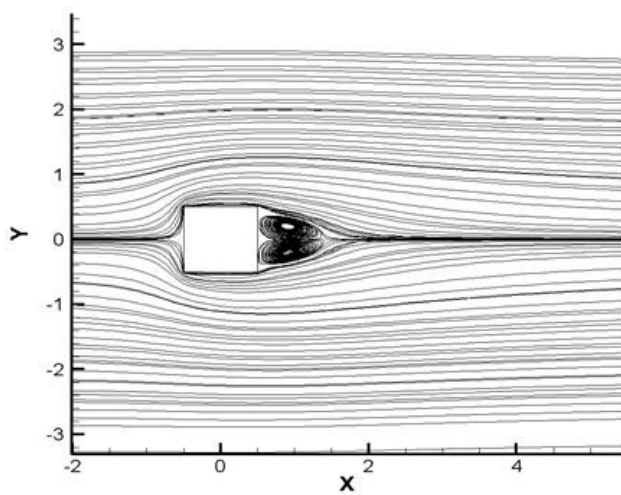
¹ Creep Flow

² Viscous Drag Force

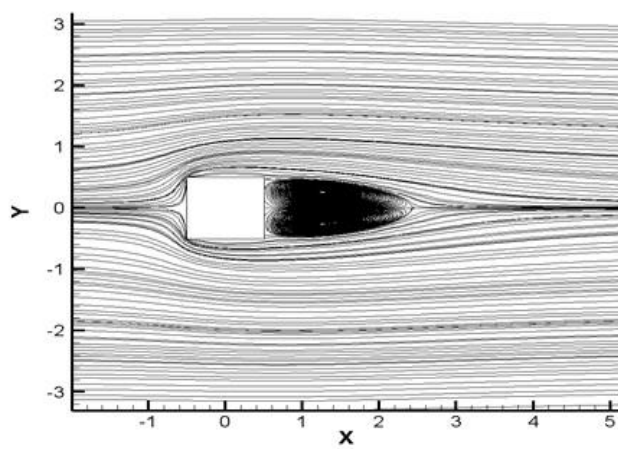
³ Pressure Drag Force



الف



ب



ج

شکل ۵-۸: خطوط تابع جریان برای رینولدزهای: الف) ۱ (ب) ۳۰ (ج) ۵۰

۵-۳-۳ جریان ناپایا

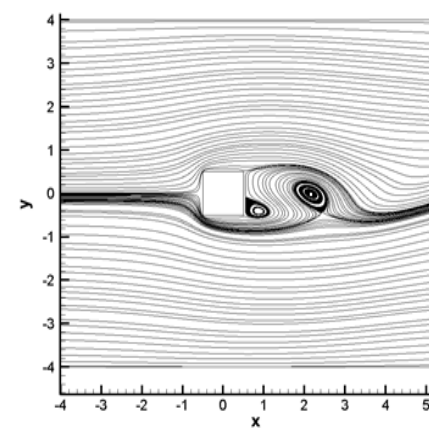
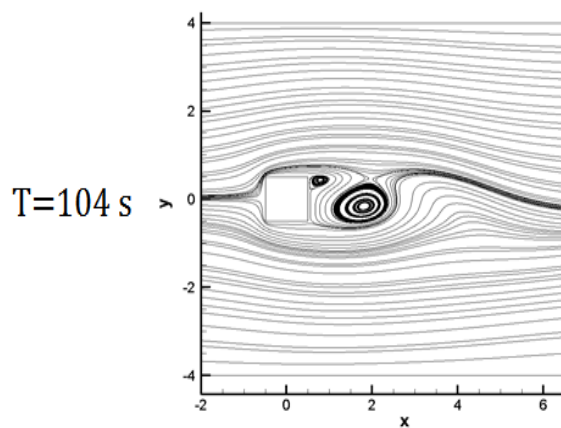
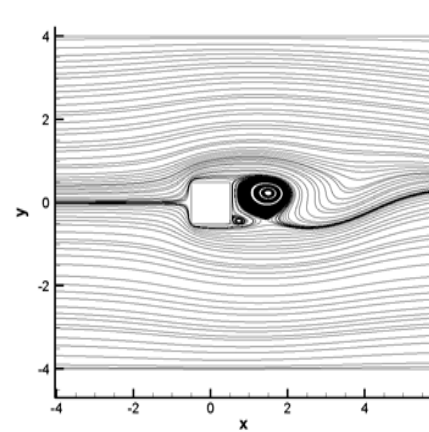
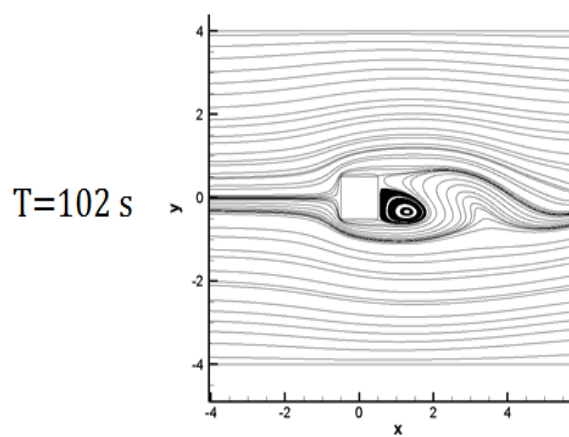
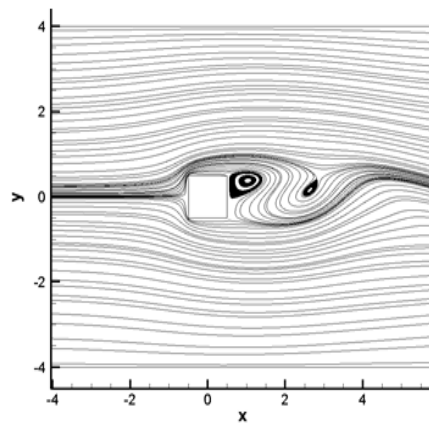
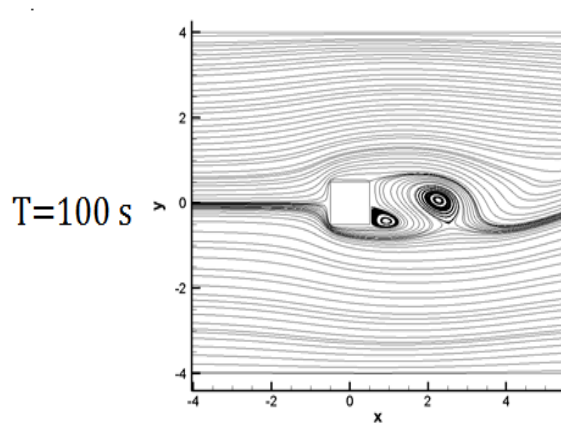
در رژیم جریان ناپایا همان طور که گفته شد گردابه‌های پشت سیلندر شروع به نوسان و جدایش می‌کنند. پارامترهای متفاوتی برای صحت سنجی حل وجود دارد. یکی الگوی جریان که در شکل‌های ۵-۹ و ۵-۱۰ الگو و خطوط تابع جریان برای رینولدزهای ۱۰۰ و ۲۰۰ در طی ۱۰ ثانیه و به فاصله‌های ۲ ثانیه‌ای به نمایش در آمده‌اند. در شکل ۵-۹ مشاهده می‌شود که در حدود ۹/۵ ثانیه یکبار الگوی جریان تکرار می‌شود. برای شکل ۵-۱۰ که نتایج جریان با رینولدز ۲۰۰ هستند این مقدار به حدود ۶ ثانیه می‌رسد و گردابه‌ها سریعتر نوسان می‌کنند که این امر قابل پیش‌بینی نیز بود. برای بررسی نوسانات معمولاً از عدد بی بعد اشتروهل^۱ استفاده می‌شود که طبق رابطه (۵-۲۱) محاسبه می‌شود:

$$St = \frac{fD}{u_{max}} \quad (۵-۲۱)$$

که فرکانس نوسانات گردابه‌ها f و طول استوانه D هستند.

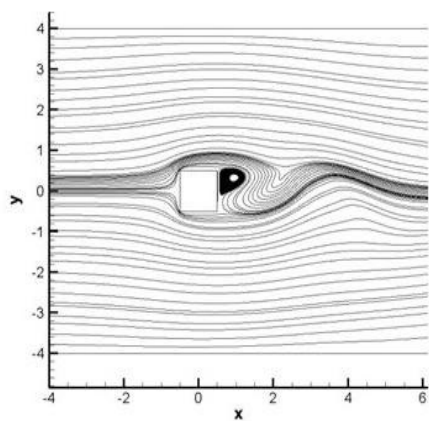
برای رینولدزهای ۱۵۰، ۱۰۰ و ۲۰۰ عدد اشتروهل برای ضریب انسداد ۰/۱ به دست آمده و با مرجع [39] مقایسه شده‌اند. همچنین پروفیل‌های سرعت افقی و عمودی در مقاطع $x^* = 0, 4, 8$ در $y^* = 0$ ترسیم شده‌اند و با نتایج مرجع [37] مقایسه شده‌اند. در راستای صحت سنجی مقادیر سرعت در شکل‌های ۵-۱۱ الف و ۵-۱۱ ب به ترتیب پروفیل مولفه‌های عمودی و افقی سرعت روی خط مرکزی $y^* = 0$ ترسیم شده است.

¹ Strouhal Number

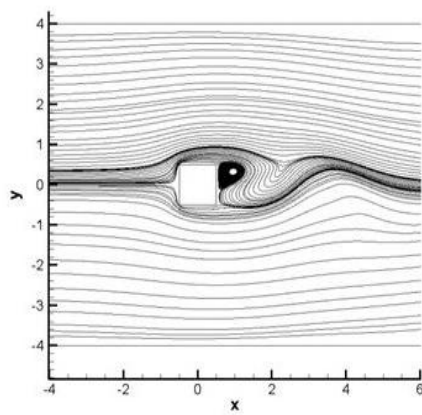


شکل ۵-۹: خطوط تابع جریان طی ۱۰ ثانیه برای رینولدز ۱۰۰

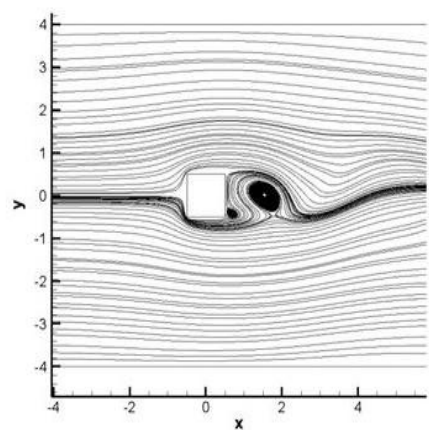
T=100s



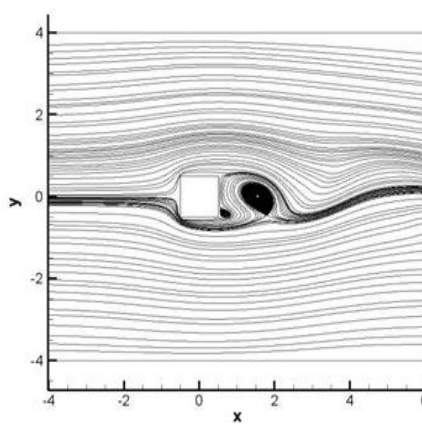
T=106s



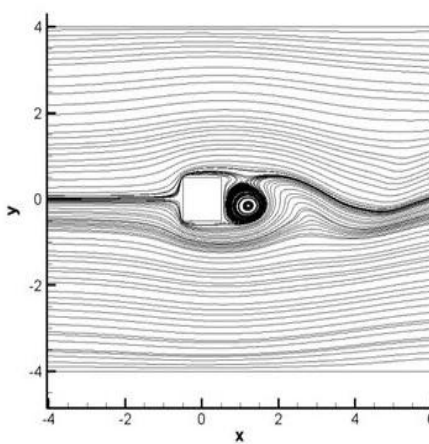
T=102s



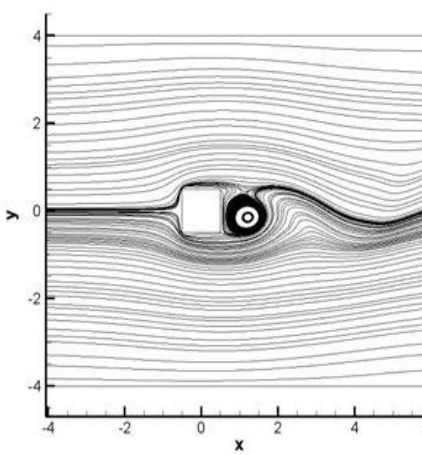
T=108s



T=104s



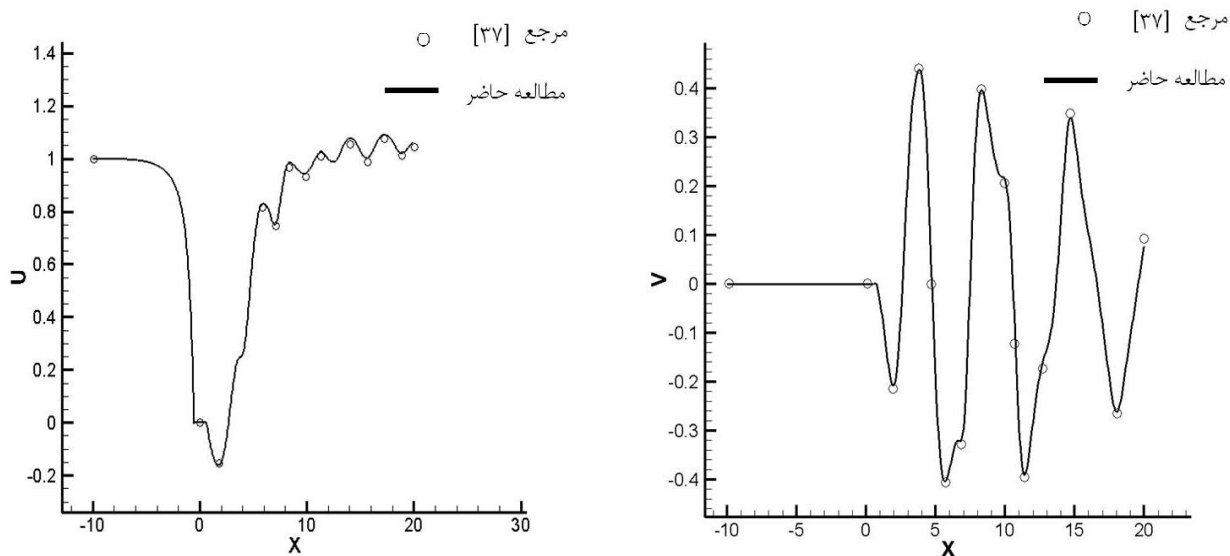
T=110s



شکل ۵-۱۰: خطوط تابع جریان طی ۱۰ ثانیه برای رینولدز ۲۰۰

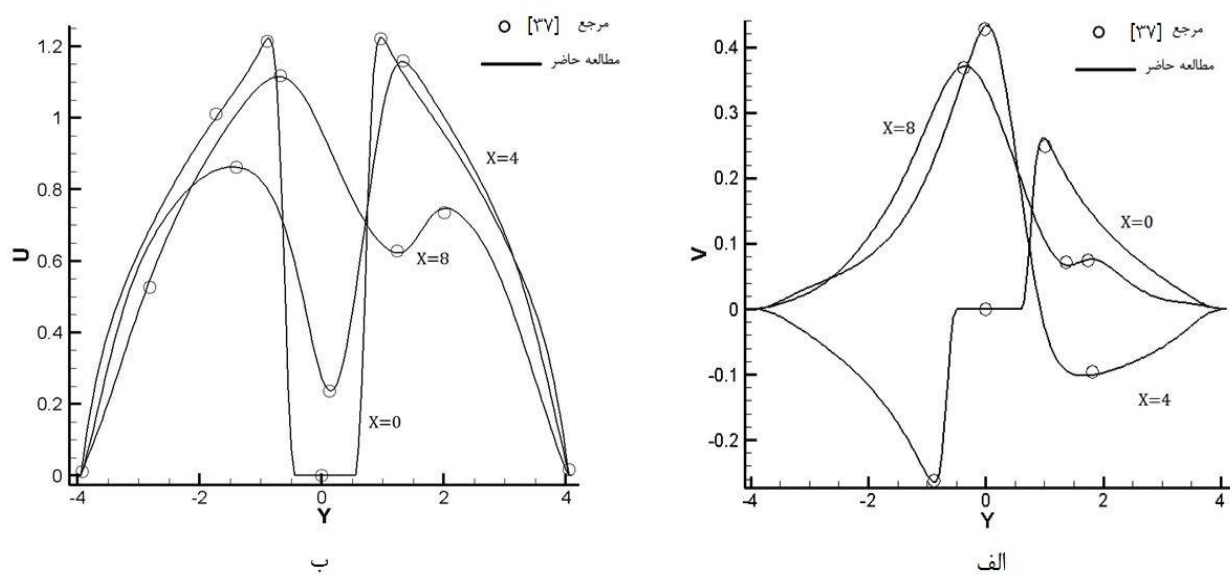
جدول ۵-۱: مقایسه مقادیر اشتروهال با مقادیر مرجع [39] با ضریب انسداد ۱/۱۰

عدد رینولدز	عدد اشتروهال مطالعه حاضر	عدد اشتروهال مرجع [39]
۱۰۰	۰/۱۴۵	۰/۱۵
۱۵۰	۰/۱۵۷	۰/۱۶۵
۲۰۰	۰/۱۶	۰/۱۶۸



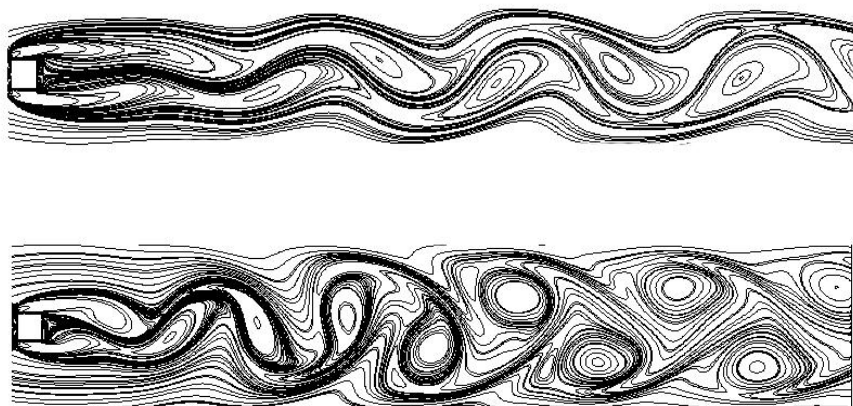
شکل ۵-۱۱: مولفه‌های سرعت الف) عمودی و ب) افقی در طول خط $Y = 0$ و در لحظه‌ای که مولفه عمودی سرعت در مکان $X = 10.5$ صفر می‌شود و پروفیل تغییر علامت می‌دهد.

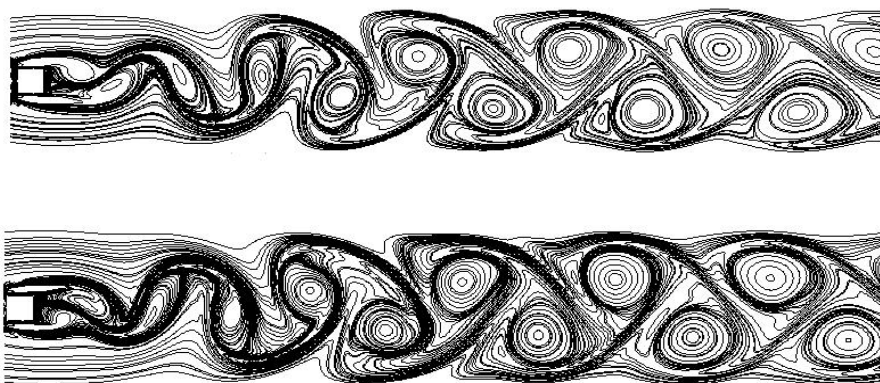
برای ترسیم پروفیل‌های سرعت در شکل ۵-۱۱ لحظه $t = 105.5$ مورد بررسی قرار گرفته است. در این لحظه در $X = 10.5$ مولفه عمودی سرعت، مقدار صفر دارد. در قسمت الف شکل ۵-۱۲ مولفه عمودی سرعت در سه مقطع و در قسمت ب مولفه افقی در سه مقطع ترسیم شده‌اند.



شکل ۵-۱۲: پروفیل‌های سرعت الف) عمودی و ب) افقی در سه مقطع عرضی مختلف $X = 0, 4, 8$ که در لحظه $t = 105.5$ به نمایش در آمده‌اند.

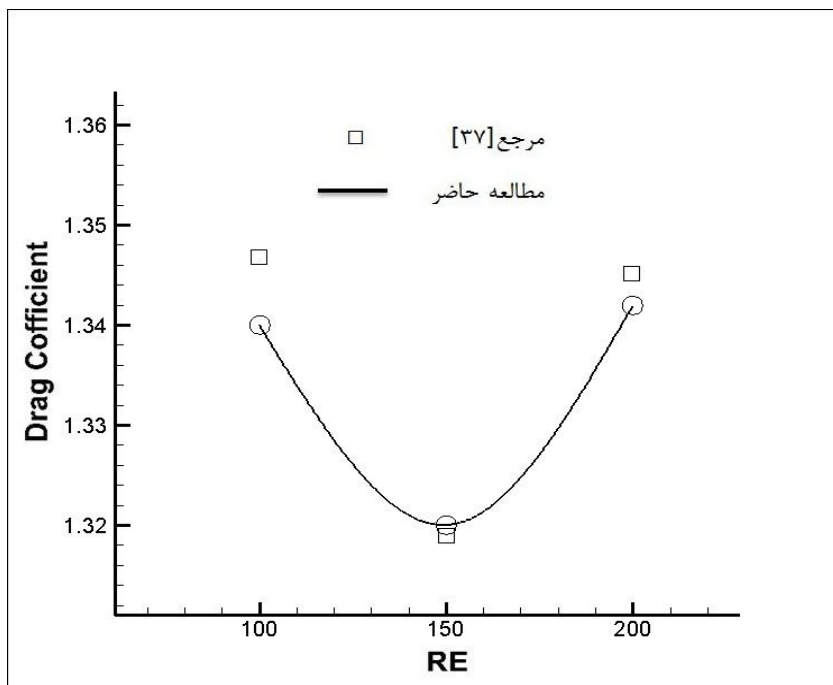
همچنین در شکل ۵-۱۳ به ترتیب در رینولدزهای ۷۵، ۱۰۰، ۱۵۰ و ۲۰۰ خطوط هم‌تراز تاوایی در لحظه $t = 100s$ رسم شده و نمایش داده شده‌اند





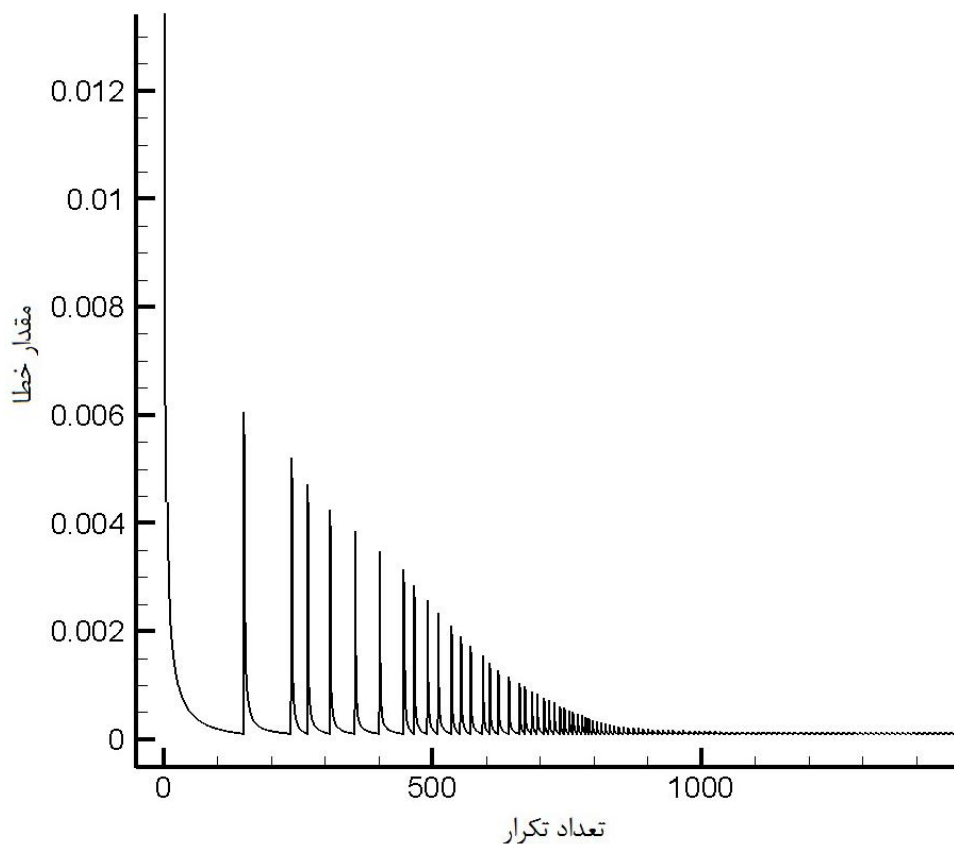
شکل ۵-۱۳: خطوط همتراز تاوایی به ترتیب از بالا به پایین برای رینولدزهای ۷۵، ۱۰۰، ۱۵۰ و ۲۰۰ در لحظه $t = 100 s$

همچنین مقدار ضریب درگ به دست آمده با مقادیر مرجع [۳۷] مقایسه شده‌اند که در شکل ۵-۱۴ می‌توان مشاهده کرد.

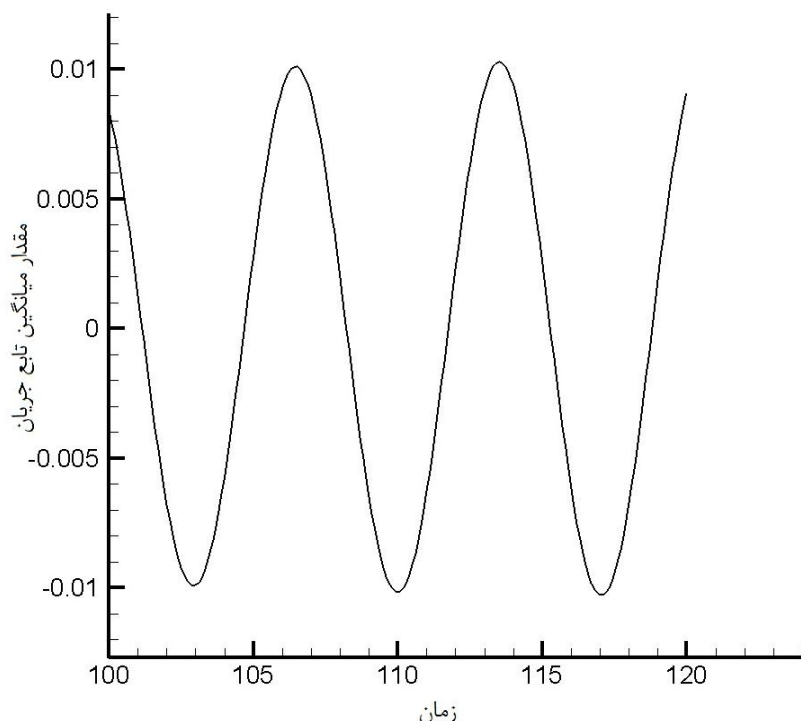


شکل ۵-۱۴: ضریب درگ برای سه رینولدز مختلف

در نهایت در شکل ۵-۱۵ می‌توان مقدار خطا در هر گام زمانی را مشاهده کرد که از گام زمانی دوم به بعد برای چند گام زمانی متوالی ترسیم شده‌اند. پرش‌های موجود مربوط به رفتن از گام قبلی به گام زمانی بعدی است. همچنین مقدار میانگین تابع جریان با استفاده از میانگین‌گیری مقدار تابع جریان روی کل استوانه انجام شده‌است و در شکل ۵-۱۶ می‌توان مشاهده کرد که این مقدار نیز به شکل نوسانی ظاهر می‌شود.



شکل ۵-۱۵: روند کاهش خطا از گام زمانی دوم به بعد در رینولدز ۱۰۰



شکل ۵-۱۶: تغییرات مقدار میانگین تابع جریان روی استوانه برای رینولدز ۲۰۰ در طی زمان ۱۰۰ تا ۱۲۰

۵-۳-۴ بررسی پیاده‌سازی روی پردازنده گرافیکی

پس از بررسی‌هایی که در فصل ۴ انجام شد، مشخص شد که رهیافت پیشنهادی این پایان‌نامه با نام الگوریتم توماس موازی می‌تواند عملکرد بهینه‌ای در پردازش موازی داشته‌باشد و حتی در مواردی بهترین انتخاب برای حل گرهای سه‌قطری می‌تواند باشد. در ادامه ملاحظاتی که در پیاده‌سازی روی پردازنده گرافیکی انجام شده از نظر می‌گذرند.

توضیح داده شد که در روش *ADI* برای یک مسأله دو بعدی، گام زمانی به دو قسمت تقسیم می‌شود و در هر زیرگام در یک جهت مکانی معادلات به شکل ضمنی گسسته می‌شوند و می‌توان

به صورت سه قطری به حل مسأله پرداخت. هنگامی که حل گر در راستای x معادلات را حل می-کند. دسترسی به حافظه از نوع هم مکان نیست (شکل ۵-۴). اما در مرحله دوم دسترسی ها هم مکان هستند و از نوع مطلوب هستند (شکل ۵-۵). از طرفی دیگر برای مثال در یک شبکه بندی 800×160 در زیرگام اول تنها ۱۸۰ نخ محاسباتی (در هر ۴ ناحیه) به کار گرفته می شوند، ولی در زیرگام دوم علاوه بر دسترسی هم مکان تعداد ۸۲۰ نخ محاسباتی درگیر محاسبات هستند و این یک مزیت است.

با توجه به بحثی که شد در هنگام حل معادلات تابع جریان و فشار که معادلاتی بیضوی هستند و روش حل آن ها تکراری است، پس می توان گفت بهترین حالت آرایش نخ های محاسباتی برای این دسته از معادلات با روش $LSOR$ ، مانند شکل ۵-۵ است که علاوه بر دسترسی هم مکان تعداد نخ بیشتری درگیر محاسبات هستند.

ساختار برنامه نوشته شده برای پردازنده گرافیکی به ترتیب زیر است:

۱- اختصاص مقدار اولیه به پارامترهای فیزیکی مانند سرعت و تاوایی در سرتاسر فضای محاسباتی.

۲- تعیین و اعمال شرط های مرزی فشار، تاوایی، تابع جریان و سرعت.

۳- حل معادله گسسته شده تاوایی در راستای x به صورت ضمنی و با رهیافت توماس موازی و انجام یک زیر گام زمانی به اندازه $n/2$

۴- حل معادله گسسته شده تاوایی در راستای y به صورت ضمنی و با رهیافت توماس موازی انجام یک زیر گام زمانی دیگر به اندازه $n/2$

۵- حل معادله تابع جریان به روش تکراری

۶- محاسبه مقادیر سرعت در تمام فضای محاسباتی

۷- محاسبه فشار با حل معادله فشار به روش تکراری

۸- رفتن به مرحله دوم و تکرار این حلقه تا رسیدن به زمان حل مورد نظر.

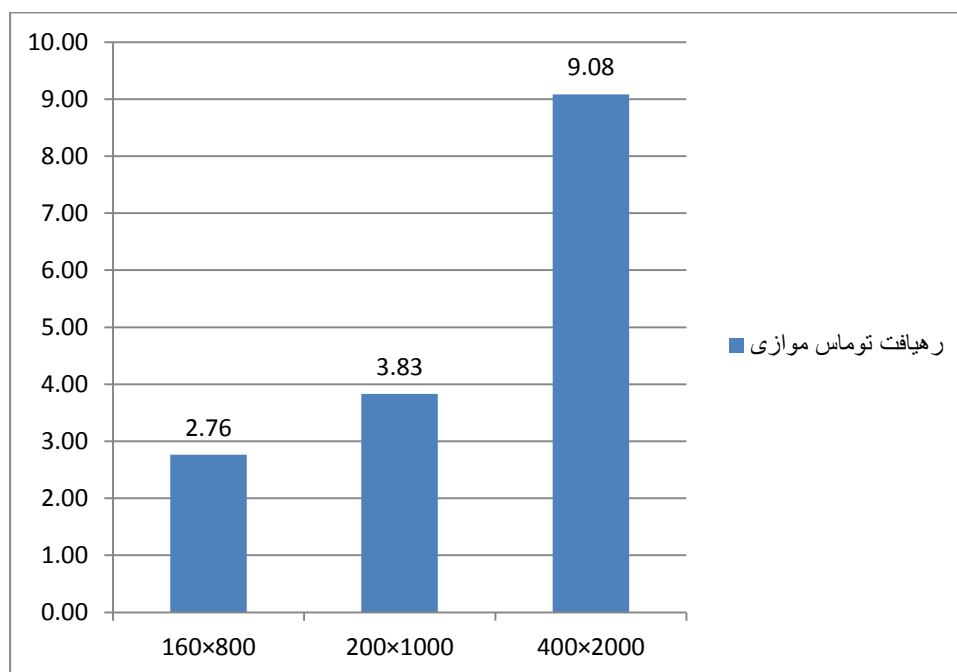
با توجه به سبک برنامه‌نویسی در پردازنده‌های گرافیکی هر کدام از بخش‌های نامبرده شده باید توسط یک کرنل به سرانجام برسند. این امر بدیهی است که هرچه بیشتر بتوان نخ‌های محاسباتی بیشتری را به کار گرفت اصولاً سرعت پردازش بیشتری بدست خواهد آمد. حداکثر نخ-های محاسباتی استفاده شده در رهیافت توماس موازی در جدول ۵-۲ در سه اندازه مختلف شبکه گزارش شده‌اند.

جدول ۵-۲: مقایسه حداکثر نخ‌های محاسباتی فعال در رهیافت توماس موازی در اندازه شبکه‌های مختلف

اندازه شبکه	ابعاد شبکه	حداکثر تعداد نخ محاسباتی در رهیافت توماس موازی
۰/۰۵	۸۰۰×۱۶۰	۸۲۰
۰/۰۴	۱۰۰۰×۲۰۰	۱۰۲۵
۰/۰۲	۲۰۰۰×۴۰۰	۲۰۵۰

با توجه به مسائل ذکر شده در ادامه نتایج و زمان حل برای ۵۰۰ گام زمانی پردازنده گرافیکی و پردازنده مرکزی آورده شده‌اند. ذکر این نکته ضروری است که بهترین و سریع‌ترین روش پردازش

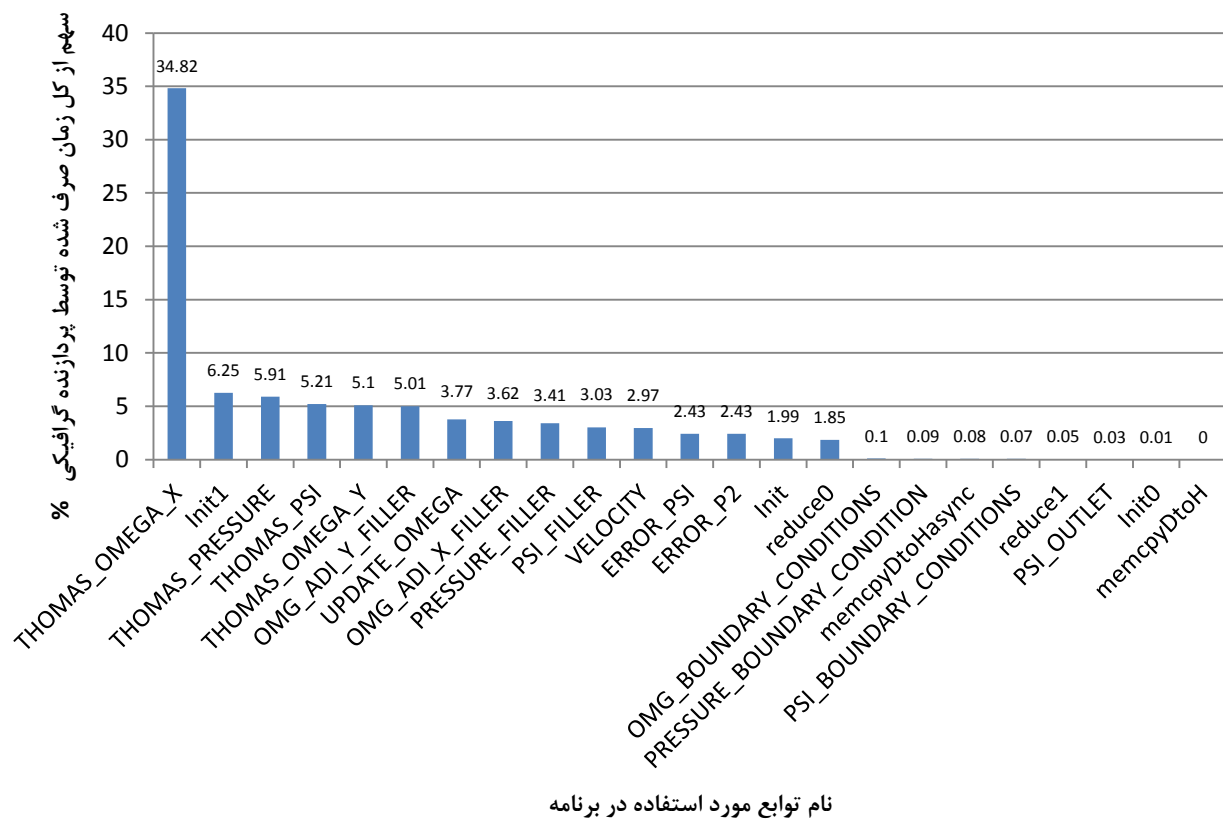
با پردازنده مرکزی با حالت بهینه در پردازش موازی و همچنین دسترسی هم‌مکان به حافظه سراسری در روش‌های معرفی شده مورد قیاس قرار گرفته‌اند. زمان پردازش ۵۰۰ گام زمانی را برای رینولدز ۱۰۰ در جریان عبوری از استوانه برای شبکه‌های محاسباتی با اندازه‌های مختلف اندازه‌گیری شده و با زمان حل روی پردازنده مرکزی مقایسه شده‌اند. در شکل ۵-۱۷ افزایش سرعت پردازش موازی در مقایسه با پردازنده مرکزی نمایش داده شده است.



شکل ۵-۱۷: افزایش سرعت در رهیافت توماس موازی نسبت به زمان حل پردازنده مرکزی

در نهایت برای داشتن دید بهتر نسبت به آنچه در پردازنده گرافیکی می‌گذرد در شکل ۵-۱۸ سهم هر کرنل یا تابع نوشته شده در یک‌بار اجرای تمام برنامه روی پردازنده گرافیکی نمایش داده شده است. به‌طور مشخص تابعی که در جهت محور طولی دستگاه‌های سه‌قطری را حل می‌کند، بیشترین سهم و بیشترین زمان حل را به خود اختصاص دهد زیرا با تعداد نخ نسبتاً کم باید

دستگاه‌هایی با اندازه بزرگتر را نسبت به تابع دیگر که در راستای محور عرضی عمل می‌کند، را حل و پردازش کند. اما مشخصاً این نمودارها برای یکبار اجرای توابع رسم شده‌اند و در روند حل تعداد تکرار این توابع متفاوت خواهند بود.



شکل ۵-۱۸: سهم زمانی توابع مختلف اجرا شده روی پردازنده گرافیکی از یکبار اجرای کل برنامه

فصل ششم:

نتیجه‌گیری

۶-۱ مروری بر مطالعه حاضر و نتیجه گیری

در این مطالعه پردازش و شبیه‌سازی جریان سیالات با استفاده از پردازنده گرافیکی مورد بحث و بررسی قرار گرفته است. همان‌طور که گفته شد استفاده از قدرت پردازنده گرافیکی بعد از سال ۲۰۰۶ رواج بیشتری پیدا کرد و پژوهشگران در این عرصه، تحقیقات بسیاری انجام داده‌اند، مخصوصاً در زمینه دینامیک سیالات محاسباتی (CFD) که بر پایه و مبنای محاسبات عددی بنا شده است. در این علم بسیار پیش آمده که برای بررسی و شبیه‌سازی یک پدیده علمی و مهندسی کامپیوترهای معمولی دچار مشکل شده و زمان بسیاری صرف می‌شود تا یک مسأله به سرانجام برسد. در راستای حل این مشکل ابرکامپیوترها و همچنین پردازش موازی راه حل این مشکل هستند.

از طرفی بسیاری از الگوریتم‌های موجود برای حل مسائل مختلف ترتیبی هستند که برای استفاده از آن‌ها باید جایگزینی ارائه شود که موازی پذیر باشد. یکی از الگوریتم‌های بسیار پرکاربرد، الگوریتم توماس است که جایگزین‌های برای این الگوریتم در پردازش موازی پیشنهاد شده‌اند که در این پایان‌نامه معرفی شده‌اند از جمله الگوریتم‌های CR و PCR . اما این الگوریتم‌ها یک محدودیت دارند و آن این است که حتماً اندازه دستگاه ورودی به این الگوریتم‌ها باید توانی از عدد ۲ باشد. این محدودیت در مسائل گوناگون می‌تواند مشکل‌ساز شود و در بعضی مسائل حل را غیر ممکن می‌کند.

در این مطالعه برای حل مشکل الگوریتم‌های موجود و محدودیت‌های آن‌ها یک رهیافت به نام رهیافت توماس موازی ارائه شد که بر مبنای الگوریتم توماس است. برای ارزیابی عملکرد این رهیافت، مسأله جریان درون حفره در نظر گرفته شد تا دقت و سرعت رهیافت توماس موازی با

الگوریتم‌های کاهش متناوب و کاهش متناوب موازی روی پردازنده گرافیکی و همچنین الگوریتم توماس کلاسیک روی پردازنده مرکزی مورد قیاس قرار بگیرند. نتایج حاصله نشان داد در شرایط برابر از نظر سرعت حل رهیافت توماس موازی تا $20/3$ برابر سریع‌تر از پردازنده مرکزی قادر به حل مسأله است. اما یک موضوع مهم در پردازش توسط پردازنده‌های گرافیکی در استفاده از حافظه دستگاه، دسترسی مناسب به مقادیر ذخیره شده است که با نام دسترسی هم‌مکان شناخته می‌شود. در بررسی‌های انجام شده نشان داده شد که با بهینه کردن دسترسی به حافظه می‌توان به افزایش سرعت حدود دو برابری رسید و ضریب تسریع پردازنده گرافیکی به حدود $38/45$ می‌رسد.

پس از اطمینان از عملکرد رهیافت توماس موازی، یک مسأله وابسته به زمان در نظر گرفته شد که با حل دستگاه معادلات سه‌قطری جریان شبیه‌سازی شود. برای این امر جریان عبوری از استوانه مربعی در نظر گرفته شد که این مسأله با توجه به محل قرار گیری استوانه یک مسأله نامتقارن از لحاظ شبکه محاسباتی است و با گسسته‌سازی معادلات و با اعمال یک روش ضمنی، دستگاه معادلات سه‌قطری حاصله را به علت نابرابر بودن، نمی‌توان با الگوریتم‌های مرسوم موازی برای دستگاه معادلات سه‌قطری پردازش کرد. در نتیجه رهیافت توماس موازی برای حل چنین مسأله‌ای بسیار توانا است. برای حل مسأله ناپایای عبوری از استوانه روش ADI انتخاب شد که یک روش بسیار پرکاربرد است. با بررسی جریان عبوری از استوانه با پردازنده گرافیکی و مرکزی و با رهیافت توماس موازی و توماس کلاسیک مشخص شد که رهیافت توماس موازی در یک شبکه با اندازه 2000×400 به ضریب تسریع حدود ۹ دست پیدا می‌کند. با بررسی حداکثر نخ‌های استفاده شده در زمان حل مشخص شد که رهیافت توماس موازی حداکثر ۲۰۵۰ نخ را برای استفاده آماده پردازش می‌کند و این تعداد نخ محاسباتی در معادلات بیضوی با روش‌های حل تکراری شاید مناسب نباشد. بنابراین برای حل این دسته از معادلات اگر راه‌کار دیگری ارائه شود شاید سرعت

حل بیشتری هم حاصل شود.

در جمع بندی و بررسی نقاط ضعف و قوت رهیافت توماس موازی می توان گفت که در مسائلی که گسسته سازی منجر به معادلات سه قطری می شوند مخصوصاً در مسائل وابسته به زمان مانند معادله هدایت حرارتی، استفاده از روش توماس موازی می تواند راه گشا باشد. اما در مسائلی که روش های تکراری برای حل نیاز باشند، شاید بهتر باشد به دیگر روش ها هم فکر شود. در نهایت رهیافت توماس موازی با بهره گیری از الگوریتم توماس که بسیار سریع و قابل درک تر نسبت به الگوریتم های موجود موازی مانند کاهش متناوب و کاهش متناوب موازی، برای حل دستگاه معادلات سه قطری است، می تواند در گسترش استفاده از پردازش موازی در تحلیل مسائل مختلف مهندسی، بسیار راه گشا باشد و محققان را تشویق و ترغیب کند که با توجه آسان تر بودن و سریع تر بودن استفاده از چنین حل گری در قیاس با حل گرهای موازی موجود، از پردازنده های گرافیکی که پتانسیل و قدرت پردازشی بسیار بالایی دارند بهره بیشتری ببرند.

۶-۲ پیشنهادات

در راستای ادامه تحقیقات در زمینه پردازش موازی با پردازنده های گرافیکی، برای تحقیقات پیش رو می توان بررسی عملکرد چند دستگاه کارت گرافیک با چند پردازنده مرکزی یا یک پردازنده مرکزی با چندین هسته سهمیم در پردازش را بررسی کرد. همچنین در مورد رهیافت توماس موازی می توان عملکرد این رهیافت را در مسائلی که معادلات بیضوی و حل های تکراری ندارند بررسی کرد. برای مثال معادله هدایت حرارتی گذرا می تواند مثال خوبی برای چنین مطالعه ای باشد. همچنین بررسی مسائلی که منجر به ایجاد دستگاه معادلاتی با تعداد قطر بیشتر مانند ۵ یا ۷ قطر می شوند با ایده رهیافت توماس موازی (که هر سطر محاسباتی مستقل از سطرهای دیگر می تواند

به حل دامنه محاسباتی بپردازد)، می‌توانند بحث جذابی برای مطالعات آینده باشند.

- [١] P. C. Shigeyoshi Tsutsui, Massively Parallel Evolutionary Computation on GPGPUs, Dortmund, Germany: Springer, 2013.
- [٢] D. M. E. Scott Larsen, "Fast Matrix Multiplies using Graphics Hardware", in Proceedings of the 2001 ACM/IEEE conference on Supercomputing, Denver, 2001.
- [٣] N. K. G. M. H. D. M. N. Galoppo, "LU-GPU: Efficient algorithms for solving dense linear systems on graphics hardware", in Proceedings of Supercomputing 2005, Seattle, 2005.
- [٤] M. G. Nathan Bell, "Efficient Sparse Matrix-Vector Multiplication on CUDA", NVIDIA Corporation., 2008.
- [٥] M. M. B. ., R. Bordawekar, "Optimizing sparse matrix-vector multiplication on gpus using compile-time and run-time strategies", IBM, 2008.
- [٦] Y. S. Ruipeng Li, "Gpu-accelerated preconditioned iterative linear solvers", The Journal of Supercomputing, vol. 2, no. 63, pp. 443-466, 2013.
- [٧] J. M. C. S. P. a. J. N. Jeroen Molemaker, "Low viscosity", in In Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, 2008.
- [٨] J. C. J. D. Yao Zhang, "Fast Tridiagonal Solvers on the GPU", in Proceedings of the 15th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, Bangalore, 2009.
- [٩] R. S. Dominik G  ddeke, "Cyclic Reduction Tridiagonal Solvers on GPUs Applied to Mixed-Precision Multigrid", IEEE Transactions on Parallel and Distributed Systems, vol. 22, no. 1, pp. 22-32, 2011.
- [١٠] Y. Z. J. D. O. Andrew Davidson, "An Auto-tuned Method for Solving Large Tridiagonal Systems on the GPU", in Proceedings of the 25th IEEE International Parallel and Distributed Processing Symposium, Anchorage, Alaska, 2011.

- [١١] D. Egloff, "High performance finite difference PDE solvers on GPUs", QuantAlea GmbH, Zurich, 2010.
- [١٢] L.-W. C. H. W.-M. S. W. Hee-Seok Kim, "A Scalable Tridiagonal Solver for GPUs", in parallel processing(ICPP), Taipei, 2011.
- [١٣] B. B. T. C. Vahid Esfahanian, "An efficient GPU implementation of cyclic reduction solver for high-order compressible viscous flow simulations", Computers & Fluids, vol. 92, pp. 160-171, 2013.
- [١٤] F. O. E. Bulent Tutkun, "A GPU application for high-order compact finite difference scheme", Computers & Fluids, vol. 55, no. 1, pp. 29-35, 2012.
- [١٥] H. M. D. S. I. G. Vahid Esfahanian, "Assessment of WENO schemes for numerical simulation of some hyperbolic equations using GPU", Computers & Fluids, vol. 80, pp. 260-268, 2012.
- [١٦] V. E. Hossein Mahmoodi Darian, "Assessment of WENO schemes for multi-dimensional Euler equations using GPU", NUMERICAL METHODS IN FLUIDS, vol. 76, pp. 961-981, 2014.
- [١٧] F. Q. A. K. a. S. Y.-S. Zhe Fan, "Gpu cluster for high performance computing", in Proceedings of the 2004 ACM/IEEE conference on Supercomputing, 2004.
- [١٨] "A lattice-boltzmann solver for 3d fluid simulation on gpu", Simulation Modelling Practice and Theory, pp. 163-171, 2012.
- [١٩] L. B. a. R. Miller, Algorithms Sequential & Parallel: A Unified Approach, Charles River Media, 2005.
- [٢٠] H. E.-R. a. M. Abd-El-Barr, Advanced Computer Architecture and Parallel, Wiley-Interscience, 2005.
- [٢١] G. K. V. K. a. A. G. A. Grama, Introduction to Parallel Computing, Addison Wesley, 2003.
- [٢٢] S. Tsutsui, Massively Parallel Evolutionary Computation on GPGPU, Dortmund: Springer, 2013.
- [٢٣] C. T. H. a. S. L. Johnsson, "Optimizing tridiagonal solvers for alternating direction methods on boolean cube multiprocessors", SIAM Journal of Scientific and Statistical Computing, vol. 3, no. 11, pp. 563-592, 1990.
- [٢٤] R. W. Hockney, "A fast direct solution of Poisson's equation using Fourier analysis", Journal of the ACM, vol. 1, no. 12, pp. 95-113, 1965.

- [۲۵] U. Ghia, K. N. S. C. T. Ghia, "High-Re solutions for incompressible flow using the Navier Stokes equations and a multigrid method", Computational Physics, 1982.
- [۲۶] E. Erturk, "Discussions On Driven Cavity Flow", International Journal for Numerical Methods in Fluids, vol. 60, pp. 275-294, 2009.
- [۲۷] K. Poochinapan, "Numerical Implementations for 2D Lid-Driven Cavity Flow in Stream Function Formulation", ISRN Applied Mathematics, 2012.
- [۲۸] P. B. Nuno Cardoso, "Time dependent simulation of the Driven Lid Cavity at High Reynolds Number", physics Fluid Dynamics, 2009.
- [۲۹] A. K. J. Prasad, "Reynolds Number and End-Wall Effects on a Lid Driven Cavity Flow", Physics of Fluids, pp. 208-218, 1989.
- [۳۰] E. C. G. Barragy, "Stream Function-Vorticity Driven Cavity Solutions Using p Finite Elements", Computers and Fluids, vol. 26, pp. 453-468, 1997.
- [۳۱] E. G. C. Erturk, "Fourth Order Compact Formulation of Navier Stokes Equations and Driven Cavity Flow at High Reynolds Numbers", International Journal for Numerical Methods in Fluids, vol. 50, pp. 421-436, 2006.
- [۳۲] "Steady-state numerical solution of the Navier–Stokes and energy equations around a horizontal cylinder at moderate Reynolds numbers from 100 to 500", Heat Transfer Eng, vol. 1, no. 17, pp. 31-81, 1996.
- [۳۳] M. Zdravkovich, Flow Around Circular Cylinders: Fundamentals, New York: Oxford University press, 1997.
- [۳۴] N. C., "Fluctuating lift on a circular cylinder: review and new measurements", J. Fluids structure, vol. 1, no. 17, pp. 57-96, 2003.
- [۳۵] H. A. S. B. N. S. Turki, "Two-dimensional laminar fluid flow and heat transfer in a channel with a built-in heated square cylinder", international journal of thermal science, vol. 12, no. 42, pp. 1105-1113, 2003.
- [۳۶] V. E. A. Sharma, "Heat and fluid flow across a square cylinder in the twodimensional laminar flow regime", Numerical Heat Transfer, vol. 3, no. 45, pp. 247-269, 2004.

- [۳۷] J. B. T. Z. F. D. M. Breuer, "Accurate computations of the laminar flow past a square cylinder based on two different methods: lattice-Boltzmann and finite-volume", *Heat and Fluid Flow*, pp. 186-196, 2000.
- [۳۸] K.-S. Y. ,. C.-B. C. Dong-Hyeog Yoon, "Flow past a square cylinder with an angle of incidence", *PHYSICS OF FLUIDS*, vol. 22, no. 4, 2010.
- [۳۹] P. G. ,. R. Z. Mahdi Yousefi Fard, "Numerical Simulations of Flow around Square Cylinder Using an Iterative High Order Difference Scheme", *Journal of Advanced Computing*, vol. 1, pp. 59-70, 2013.
- [۴۰] F.-S. L. Yusuke Koda, "The Lattice Boltzmann Method Implemented on the GPU to Simulate the Turbulent Flow Over a Square Cylinder Confined in a Channel", in *Proceedings of the International Conference on ICA*, 2014.

Abstract

In recent years, use of Graphic Processor Units (GPU) as a parallel processor and accelerator to scientific calculations has been growing up. This type of parallel processing has found a special place in computational fluid dynamics (CFD). Researchers simulate many engineering and natural phenomena with the aid of graphic processors and gained considerable speed up. In this research, two parallel algorithms for solving tri-diagonal system of equations, Cyclic Reduction (CR) and Parallel Cyclic Reduction (PCR) are employed and a different strategy is proposed to solve such system of equations. This approach is called parallel Thomas approach. Lid-Driven cavity problem is used to assessment these parallel algorithms. This problem was simulated by using Thomas algorithm that run on Central Processing Unit (CPU) as well. Runtimes and physical parameters of parallel algorithms, parallel Thomas approach and Thomas algorithm were compared. The maximum speed up of CR, PCR and parallel Thomas approach against CPU runtime was about 4.4x, 5.2x and 38.5x respectively. In this thesis the effect of coalesced memory access and uncoalesced memory access to global memory was studied too, and a 2x speed up achieved in coalesced memory access on GPU.

In other hand Parallel Thomas Algorithm can be implemented on transient problems. Many approaches exists which with discretizing equations led to Tri-Diagonal system of equations, for example Alternating Direction Implicit(ADI) method. To assess Parallel Thomas Algorithm performance in time dependent problems, flow over a square considered which with implementing PTA, speed up about 9x achieved.

Keywords

Tri-Diagonal system of equations, Graphic Processor, Parallel Thomas Approach, Flow over a square, Alternating Direction Implicit method.



University of Shahrood

Faculty of Mechanical Engineering

Simulation of Flow around a Cylinder by Solving the Tri-Diagonal System of Equations on GPUs

Milad Souri

Supervisor:

Dr. Pooria Akbarzade

Dr. Mohsen Nazari

Advisor:

Dr. Hossien Mahmoodi Darian

September 2015