اجزای محدود تطابقی دوبعدی به کمک GPGPU

امیرحسین خاتمی[1]، سعید اصیل قره‌باغی[2*]

۱- دانشکده مهندسی عمران، دانشگاه صنعتی خواجه نصیرالدین طوسی

۲- * دانشکده مهندسی عمران، دانشگاه صنعتی خواجه نصیرالدین طوسی، asil@kntu.ac.ir

چکیده

خطای گسسته‌سازی یکی از خطاهای رایج در روش اجزای محدود است. برای کاهش خطای گسسته‌سازی ممکن است از روش‌های تطابقی استفاده شود. روش‌های تطابقی عموماً حجم محاسبات زیادی دارند؛ یک روش برای کاهش حجم این محاسبات، استفاده از عملگرهای انتقال داده است. حتی باوجود عملگرهای انتقال داده هنوز هم روش تطابقی زمان زیادی را از کاربران می‌گیرد. با توجه به امکانات و توانایی‌های جدیدی که پردازنده‌های گرافیکی به کاربران خود جهت انجام محاسبات همه‌منظوره تحت پلتفرم کودا می‌دهند و صرفه اقتصادی مناسب پردازنده‌های گرافیکی نسبت به پردازنده‌های معمولی، در این مقاله سعی شده است الگوریتمی ارائه شود که بتوان با استفاده از پردازش همه‌منظوره بر روی پردازنده‌های گرافیکی زمان انجام محاسبات را کاهش داد. الگوریتم ارائه‌شده بر اساس تحلیل اولیه اجزای محدود، با شبکه تقریبا یکنواخت شروع می‌شود و در هر مرحله بر اساس گرادیان جابه‌جایی و به‌صورت هوشمند، شبکه را ریزسازی می‌کند. الگوریتم معمول این شیوه در چند مرحله بهبود یافته است. مرحله تشکیل وصله به کمک روش K همسایه پیاده‌سازی شده تا بتوان آن را به صورت مؤثرتر موازی نمود. در مرحله انتقال اطلاعات نیز از یک روش دینامیک جهت تعیین بهترین منحنی از دسته بهترین منحنی‌ها استفاده‌شده است. در پیاده‌سازی ایده‌ها از زبان پایتون استفاده‌شده است تا مخاطب بیشتر داشته و به‌صورت کد منبع باز منتشر شود. نتایج نشان می‌دهد که میزان تسریع این الگوریتم متناسب با تعداد المان‌ها افزایش می‌یابد. به عنوان مثال برای مسئله‌ای با تعداد ۹۰۸ المان، سرعت پردازش برای مراحل یک الی سه از تظریف به ترتیب ۶٫۶، ۹٫۱ و ۱۲٫۷ برابر شده است. مجموع زمان مورد نیاز برای پردازش هر سه مرحله در حالت سریال ۹۶ ثانیه بوده که با پیاده سازی این الگوریتم به ۸ ثانیه کاهش یافته است. نتایج نشان می‌دهد که این نرم افزار می‌تواند برای تسریع آنالیز به روش اجزای محدود تطابقی جهت کاهش خطای گسسته‌سازی استفاده شود.

کلمات کلیدی

عملگر انتقال داده، روش‌های المان محدود تطابقی، پردازنده‌ی گرافیکی، GPGPU ، KNN

۱- مقدمه

یکی از مهم‌ترین شیوه‌های کاهش خطای گسسته‌سازی، استفاده از روش‌های تطابقی[1] است[۱]. روش‌های تطابقی انواع مختلفی دارند که رایج‌ترین این روش‌ها، روش *h-adaptive* است. در این روش، ریزسازی[2] المان‌ها به‌صورت مرحله‌ای انجام می‌شود تا حداکثر خطای گسسته‌سازی به‌اندازه تعیین‌شده برسد[۲].

با پیشرفت علم، مسائلی که با روش اجزای محدود حل می‌شود بزرگ‌تر و پیچیده‌تر شده‌اند؛ همچنین نیاز جامعه به حل دقیق‌تر مسائل افزایش‌یافته است. بزرگ‌تر و پیچیده‌تر شدن مسائل، باعث افزایش حجم محاسبات در روش اجزای محدودشده است. استفاده از روش‌های حل تطابقی برای کاهش خطای گسسته‌سازی حجم محاسبات را دوچندان می‌کند. برای کاهش حجم محاسبات، می‌توان به‌جای حل مسئله از ابتدا و بر روی کل دامنه، از عملگرهای انتقال اطلاعات[3] استفاده کرد؛ که اطلاعات را از شبکه مرحله قبل به شبکه کنونی انتقال می‌دهد. تا به امروز عملگرهای انتقال مختلفی جهت انتقال اطلاعات بین شبکه‌های مراحل متوالی پیشنهادشده است که رایج ترین این روش ها در ادامه شرح داده شده است. روش میانگین نقطه وزن‌دار، توسط گرندی[4] مطرح شد که محافظه‌کارانه و ساده‌ترین روش بود[۳]. این روش، بر اساس میانگین‌گیری وزن‌دار از المان‌های قدیمی که با آن المان جدید در تماس بوده‌اند، محاسبه می‌شود. وزن با استفاده از مساحت مشترک المان با المان جدید تعیین می‌گردد. با کمک این روش، می‌توان مقداری گرادیان از شبکه‌بندی قدیمی به شبکه‌بندی جدید انتقال داد. روش انتقال داده، برای شبکه‌بندی‌های غیرهمسان کاربرد دارد[۴]. روش المان میرا، توسط آربوگاست[5] ارائه شد[۵]. در این روش، دامنه را به چندین زیر المان غیر منطبق تقسیم می‌کنند. این زیر المان، به دو گروه المان میرا و المان نامیرا دسته‌بندی می‌شود. مقادیر در این روش، بر اساس باقی‌مانده وزن‌دار تعیین می‌گردد. وزن این المان، به‌صورت تابعی از شکل المان تعیین می‌شود [۶]. ارتیز و کویگلی، اپراتور انتقال ثابت را با استفاده از تابع مناسب هو-واشیزو[6] پیشنهاد دادند که برای انتقال داده در محیط پیوسته به کار می‌رود[۷]. این روش، بر اساس میان‌یابی از نقاط گاوس شبکه‌بندی قدیمی بناشده است. این روش، برای انتقال داده مسائل غیرخطی کاربرد دارد و در صورت استفاده از یک شبکه المان جدید روی کل دامنه، دچار مشکل می‌شود[۸]. لی و بت[7]، روشی بر پایهٔ میان‌یابی با کمک توابع شکلی ارائه دادند[۹]. در این روش، ابتدا متغیرهای وابسته به زمان به گره‌های شبکه المان قدیمی انتقال داده می‌شوند. در مرحله بعد، با کمک توابع شکل، متغیرهای وابسته به زمان در نقاط مدنظر میان‌یابی می‌شوند. در این روش، برای درستی معادلات سازنده، نیاز به انتقال کرنش پلاستیک مؤثر و تغییر شکل آزمایشی الاستیک بین شبکهٔ المان‌ها وجود دارد. پریک و همکاران، از تکنیک مشابهی استفاده کردند. آن‌ها در این تکنیک، مقادیر جابه‌جایی منتقل‌شده در انتهای تکرار n در شبکه المان قدیمی را به‌عنوان یک راه‌حل آزمایشی برای شبکه المان جدید برای تکرار n در نظر گرفتند. روش‌های بالا، علی‌رغم مزایایی که دارند، دارای خطای زیادی نیز هستند. زینکیویچ و ژو، روش بازیابی فوق همگرا را بر روی المان‌های وصله ارائه دادند[۹]. بر اساس روش بازیابی اپراتور انتقال داده، روش *SPR* توسعه داده شد. این روش، به دلیل استفاده از روش بازیابی تنش، دارای خطای کمتری نسبت به سایر روش‌هاست. روش بازیابی فوق همگرا، بر اساس برازش بهترین تابع چندجمله‌ای، بر روی نقاط فوق همگرای المان وصله بناشده است. این روش، برای مسائل خطی روشی دقیق و قابل قبول است.

[1] Adaptive methods
[2] Refinement
[3] Data transfer operators
[4] Grandy
[5] Arbogast
[6] Hu-washizu
[7] Bathe

برومند[1] و زینکیویچ، در تحقیقات خود، روش SPR را برای محاسبه کرنش در مسائل کشسانی- خمیری [2] توسعه دادند[۱۰]. کیتا مورا[3]، در تحقیق خود، این روش را برای مسائل با تغییر شکل‌های بزرگ در مواد فوق کشسانی[4] بکار برد [۱۱]. تانگ[5] و ساتو[6]، از روش SPR برای مسئله روانگرایی استفاده کردند[۱۲].

در سال ۲۰۰۴، روش بازیابی فوق همگرا، برای تغییر شکل‌های بزرگ توسط گو[7]، هانگ[8] و زونگ[9] اصلاح گردید[۱۳]. گو و همکاران، برای برقراری معادلات تعادل و بهبود دقت در مسائل غیرخطی، روش بازیابی فوق همگرای اصلاح‌شده را پیشنهاد دادند[۱۳]. آن‌ها در تحقیقات خود نشان دادند که این روش، حتی در مسائل غیرخطی، با تغییر شکل‌های بزرگ نیز می‌تواند کارا باشد. در این روش، برای برازش صفحه، از یک چندجمله‌ای استفاده می‌شود که یک درجه بالاتر از درجهٔ چندجمله‌ای مورد استفاده در روش SPR است.

خویی و اصیل قره‌باغی، با استفاده از روش بازیابی فوق همگرا، اپراتور انتقال داده را برای مسائل کشسانی- خمیری سه‌بعدی توسعه دادند. در این تحقیق، با استفاده از روش حداقل مربعات و نتایج اولیه روش اجزای محدود، چندجمله‌ای با پیوستگی‌های C_0 و C_1 بروی نقاط فوق همگرا بر اساس روش SPR برازش شد. آن‌ها نشان دادند که این اپراتور، دقت قابل قبولی برای مسائل غیرخطی دارد[۱۴-۱۷]. روش SPR یکی از دقیق ترین وموثر ترین روش‌های انتقال داده است که در این در پژوهش از این اپراتور به عنوان اپراتور انتقال متغیر های داخلی استفاده شده است. این اپراتور دارای حجم محاسباتی زیادی است و زمان زیادی از کاربر می‌گیرد.

با توجه به اینکه محاسبات عملگر انتقال داده بخش نسبتاً بزرگی از محاسبات حل مسائل اجزای محدود به روش تطابقی را تشکیل می دهند[۱۸]. می‌توان با بهره‌گیری از چندین پردازشگر به‌صورت هم‌زمان، زمان انتقال داده بین شبکه‌ها را به‌طور چشمگیری کاهش داد. مباحث نظری و ریاضی پردازش موازی و استفاده از چندین پردازنده به‌صورت هم‌زمان از اوایل دهه شصت موردتوجه قرار گرفت. این مباحث متکی بر تقسیم روش حل مسئله به بخش‌های کوچک‌تر و مستقل است [۱۹]. یکی از مهم‌ترین تحقیقات در این زمینه، دسته‌بندی پیشنهادشده از سوی فلین[10] (۱۹۶۶) است. وی پردازنده‌ها را به چهار دسته تقسیم کرد. در این پیشنهاد، بر اساس مفاهیم جریان داده و جریان کنترل، پردازنده‌ها به چهار مدل محاسباتی پردازش موازی *SISD*[11] و *SIMD*[12] و *MISD*[13] و *MIMD*[14] تقسیم می‌شوند. با توجه به اینکه محاسبات عملگر انتقال داده، دارای حجم بسیار زیادی از محاسبات تکراری بر روی‌داده‌های تانسوری[15] است، مدل محاسباتی مناسب برای این نوع محاسبات، مدل *SIMD* است[۲۰]. در مدل محاسباتی *SIMD*، استفاده از پردازنده گرافیکی[16] در کاهش زمان محاسبات عملگرهای انتقال داده، بسیار مؤثر است. در پردازنده‌های گرافیکی، وجود تعداد زیادی هسته با توان پردازشی کم که به‌صورت هم‌زمان امکان انجام محاسبات ساده را بر روی تعداد زیادی داده، فراهم می‌کند، سبب می‌شود پردازش موازی این نوع الگوریتم‌ها بر روی پردازنده‌های گرافیکی بسیار بهینه و باصرفه شود[۱۹]. این ویژگی‌های پردازنده‌های گرافیکی به همراه نوع داده‌ها و الگوریتم پیشنهادی باعث شده است این عملگر انتقال داده برای اجرا روی پردازنده‌های گرافیکی مناسب باشد.

درگذشته پردازنده‌های گرافیکی فقط به‌منظور انجام عملیات گرافیکی استفاده می‌شد ولی با پیشرفت معماری پردازنده‌های گرافیکی، امکان انجام محاسبات عمومی بر روی پردازنده گرافیکی فراهم شد. محاسبات عمومی بر روی پردازنده گرافیکی به استفاده از واحد

---

[1] Boroomand
[2] Elasto-plasticity
[3] Kitamura
[4] Hyperelasticity
[5] Tang
[6] Sato
[7] Gu
[8] Hung
[9] Zong
[10] Flynn
[11] Single Instruction Stream, single Data stream
[12] Single Instruction Stream, Multiple Data
[13] Multiple Instruction Stream, Single Data
[14] Multiple Instruction Stream, Multiple Data
[15] Tensor
[16] GPU

پردازش گرافیکی برای کارهایی فراتر از پردازش گرافیکی سنتی اشاره دارد. محاسبات عمومی بر روی پردازنده گرافیکی ابزاری قدرتمند برای سرعت بخشیدن به طیف وسیعی از محاسبات در حوزه‌های تحقیقاتی و علمی-مهندسی است. ماهیت موازی معماری *GPU* اجازه می‌دهد تا سرعت به‌صورت قابل‌توجهی در انواع خاصی از محاسبات در مقایسه با *CPU* های سنتی افزایش یابد.

تا قبل از سال ۲۰۰۷، برای انجام محاسبات عمومی بر روی پردازنده گرافیکی باید از *API* های گرافیکی مانند *OpenGL* و *DirectX* برای برنامه‌نویسی غیرمستقیم پردازنده‌های گرافیکی استفاده می‌شد[۲۰]. شرکت نویدا[1] در سال ۲۰۰۷، پلتفرم کودا[2] را برای پردازش‌های همه‌منظوره در واحدهای پردازش گرافیکی کارت‌های گرافیک مخصوص شرکت خود ارائه داد. این پلتفرم به دلیل سهولت و افزایش سرعت توسعه برنامه‌های با عملکرد بالا، انقلابی در پردازش‌های همه‌منظوره بر روی واحدهای پردازش گرافیکی ایجاد کرد و محبوبیت فراوانی کسب نمود. تحقیقات زیادی درباره پردازش‌های همه‌منظوره روی واحدهای پردازش گرافیکی در حوزه اجزای محدود انجام‌شده است . اکثر این تحقیقات بر روی عملیات *SpMV* [3] در فاز حل یا مرحله تشکیل ماتریس[4] متمرکز بودند[۲۱]. در این زمینه می‌توان به تحقیقات جیانفی ژانگ[5] و دفی شن[6] (۲۰۱۳) اشاره کرد که محاسبات اجزای محدود خطی را در ناحیه الاستیک با استفاده از کودا بر روی پردازندهٔ گرافیکی انجام دادند؛ همچنین به تحقیقات وای ژانگ[7] و همکاران (۲۰۲۰) اشاره نمود که با استفاده از پلتفرم کودا، مسائل پیچیده اجزای محدود را تحلیل و بهینه‌سازی کردند[۲۲, ۲۳].

در این پژوهش، ارائه و اجرای الگوریتمی موازی بر روی پردازنده گرافیکی بر اساس روش تطابقی *h-adaptive* با استفاده از عملگرهای انتقال داده و باهدف کاهش خطای گسسته‌سازی و افزایش سرعت انجام فرایند صورت گرفت. درروش استفاده‌شده این مقاله، به‌منظور کاهش خطای گسسته‌سازی ابتدا ریزسازی بر روی شبکه المان‌ها انجام می‌شود و سپس با کمک عملگرهای انتقال داده، اطلاعات از شبکه قبلی به شبکه کنونی انتقال می‌یابد. این فرآیند به‌طور مکرر انجام می‌شود تا بیشینه خطای گسسته‌سازی به میزان موردنظر کاهش یابد. این فرآیند از دو بخش اصلی ریزسازی شبکه المان‌ها و انتقال اطلاعات شبکه قدیم به شبکه جدید تشکیل‌شده است. در این تحقیق، پیاده‌سازی موازی اپراتورهای انتقال داده بر روی پردازنده‌های گرافیکی (*GPU*) انجام گرفته است. برخلاف تحقیقات پیشین که انتقال داده‌ها را به‌صورت سریال بر روی پردازنده‌های مرکزی (*CPU*) انجام می‌دادند، این پژوهش الگوریتمی را توسعه داده است که با استفاده از پردازش موازی بر روی *GPU*، مدت زمان اجرای الگوریتم به طور قابل ملاحظه ای کاهش می دهد. به عنوان مثال برای مسئله‌ای با تعداد ۹۰۸ المان، سرعت پردازش برای مراحل یک الی سه تظریف به ترتیب ۶,۶، ۹,۱ و ۱۲,۷ برابر شده است. مجموع زمان مورد نیاز برای پردازش هر سه مرحله در حالت سریال ۹۶ ثانیه بوده که با پیاده سازی این الگوریتم به ۸ ثانیه کاهش یافته است. علاوه بر این، مشابه تحقیقات پیشین، این پژوهش نیز کاهش خطای گسسته‌سازی و پخش یکنواخت‌تر آن در دامنه را به همراه داشته است.

۲- انتقال اطلاعات از شبکه قدیم به شبکه جدید

در این فرایند، انتقال اطلاعات توسط عملگرهای انتقال داده صورت می‌گیرد. عملگرهای انتقال داده را می‌توان در حالت کلی به‌صورت زیر تعریف نمود:

$$\mathbf{q}^{new} = T\left(\mathbf{q}^{old}\right) \qquad (۱)$$

در عبارت بالا، Tعملگر انتقال داده، $\mathbf{q}$متغیر میدان، $\mathbf{q}^{old}$ مقدار $\mathbf{q}$در شبکه قدیم و $\mathbf{q}^{new}$در شبکه جدید است.

[1] NVIDIA
[2] CUDA
[3] Sparse matrix–vector multiplication (SpMV)
[4] Matrix assembly
[5] Jianfei Zhang
[6] Defei Shen
[7] Y Zhang

متغیرهایی که رفتار ماده الاستوپلاستیک[1] معمولی را توصیف می‌کنند به دو دسته متغیرهای حالت[2] و متغیرهای داخلی[3] تقسیم می شوند. متغیرهای حالت شامل جابه‌جایی‌های گرهی ($\mathbf{u}_n$) و متغیرهای داخلی شامل تانسور کرنش پلاستیک ($\boldsymbol{\varepsilon}_n^p$)، تانسور کرنش (ε_n)، تانسور تنش کوشی (σ_n) و بردار متغیر داخلی ($\mathbf{q}_n$) می‌شوند. در مسائل اجزای محدود، از دو عملگر انتقال اطلاعات استفاده می شود؛ یکی برای متغیرهای داخلی و دیگری برای متغیرهای حالت [۲۴]. جهت تفکیک بهتر مبنای نظری این دو عملگر، هر یک از آن‌ها در بخش جداگانه ارائه می‌شود.

۲-۱- عملگرانتقال متغیر حالت

عملگر انتقال متغیر حالت (T_1) را می‌توان به‌صورت تعریف کرد:

$$\boldsymbol{u}_B^{new} = T_1\left(\boldsymbol{u}_A^{old}\right) \qquad (۲)$$

که در آن، $\boldsymbol{u}_A^{old}$ اطلاعات میدان جابجایی در مثلث A واقع در شبکه قدیم و $\boldsymbol{u}_B^{new}$ اطلاعات میدان منتقل شده به گره B در شبکه جدید است. مثلث A همان مثلثی از شبکه قدیم است که نقطه B از شبکه جدید را در خود جای می دهد. عملگر (T_1) با کمک توابع شکل المان در شبکه قدیم ($\mathbf{N}^{old}$) تعریف می‌شود. به‌این‌ترتیب، جابه‌جایی‌های نقاط گره‌ای شبکه قدیم با کمک رابطه زیر، به جابه‌جایی‌های نقاط گره‌ای شبکه جدید منتقل می‌شود[۲۵]:

$$\left(\boldsymbol{u}_B^{new}\right) = \left(\boldsymbol{N}^{old}, \boldsymbol{u}_A^{old}\right)\left(\boldsymbol{r}_B\right) \qquad (۳)$$

که در معادله فوق $\boldsymbol{r}_B$ بردار مختصات گره B از شبکه جدید است. در این راستا لازم است ابتدا به کمک مختصات هر گره از شبکه جدید، مشخص شود که آن گره داخل کدام المان از شبکه قدیم قرار می‌گیرد. سپس به کمک توابع شکل آن المان در شبکه قدیم، مقادیر جابجایی در این نقطه محاسبه می‌شود. این فرایند برای همهٔ المان‌ها تکرار می‌شود و جابه‌جایی گرهی را از نقاط گره شبکه قدیم به شبکه جدید انتقال می‌دهد.

۲-۲- عملگر انتقال متغیر داخلی

این عملگر جهت انتقال اطلاعات تنش و کرنش و در صورت لزوم مقادیر افزایشی آن‌ها مورداستفاده قرار می‌گیرد. با فرض آنکه متغیرهای حالت و داخلی در شبکه قدیم در دسترس باشند، می‌توان متغیرهای داخلی در شبکه قدیم را به‌صورت زیر محاسبه نمود که در آن T_2 همان عملگر دوم انتقال اطلاعات است.

$$\left({}^p\boldsymbol{\varepsilon}_n^{new}, \mathbf{q}_n^{new}\right) = T_2\left({}^p\boldsymbol{\varepsilon}_n^{old}, \mathbf{q}_n^{old}\right) \qquad (۴)$$

هدف، تعریف عملگر T_2 است؛ به‌گونه‌ای که بتوان کار انتقال داده را با کمترین خطا و کمترین هزینه محاسباتی انجام داد.

یک روش ساده جهت تعریف این عملگر آن است که ابتدا مطابق آنچه در بخش قبل آمد، اطلاعات جابجایی از شبکه قدیم به شبکه جدید منتقل شود. سپس به کمک توابع شکل در شبکه جدید، مقادیر تنش و کرنش در شبکه جدید محاسبه شوند. در صورت استفاده از این روش، محاسبات در دو مرحله دارای خطا خواهند بود. یک مرحله مربوط به انتقال جابجایی از شبکه قدیم به شبکه جدید است. مرحله دوم با فرضیات مرتبط با روش اجزای محدود قابل‌رؤیت است. ارتباط دادن مقادیر جابجایی به تنش و کرنش اگرچه متعارف و صحیح است اما در اینجا ممکن است باعث بروز دومین خطا شود[۱۸, ۲۴, ۲۶-۲۹].

روش دیگر انتقال تنش و کرنش از شبکه قدیم به شبکه جدید، انتقال اطلاعات به‌صورت مستقیم از نقاط گاوس شبکه قدیم به نقاط گاوس شبکه جدید است. به‌این‌ترتیب خطای محاسباتی صرفاً در یک مرحله به وجود خواهد آمد. همان‌طور که در بخش بعدی مورداشاره قرار خواهد گرفت، مزیت این روش آن است که نقاط گاوس اصطلاحاً فوق همگرا هستند. لذا برآورد تنش و کرنش به‌طور مستقیم در آن

[1] Elastoplastic
[2] State variables
[3] Internal variables

نقاط از دقت بالاتری برخوردار خواهد بود. در راستای عملیاتی نمودن این روش، ابتدا لازم است که تنش و کرنش در شبکه قدیم هموارسازی شود. پس از بازیابی[1] و بهبود نتایج تنش و کرنش در شبکه قدیم، می‌توان از نتایج این متغیرها جهت محاسبه مقادیر متناظر در نقاط گاوس شبکه جدید استفاده نمود. همان‌طور که دیده می‌شود در این روش به یک شیوه بازیابی اطلاعات نیاز است[۳۰, ۳۱].

عملگرهای مختلفی برای انتقال متغیرهای داخلی وجود دارد که می‌توان به عملگرهای SPR[2] و عملگر المان میرا اشاره کرد. در این مقاله از عملگر SPR به‌منظور انتقال متغیرهای داخلی استفاده شد. یکی از اهداف مقاله، بهینه‌سازی این الگوریتم است. پس از محاسبه مقادیر تنش و کرنش بازیابی شده، نه‌تنها می‌توان مقدار خطای موجود در تحلیل را به‌صورت تقریبی بازیابی نمود بلکه می‌توان از این میدان بهبودیافته جهت محاسبه تنش و کرنش در نقاط گاوس شبکه جدید استفاده نمود.

۳- بهبود نتایج تحلیل به کمک وصله شامل نقاط فوق همگرا

همان‌طور که قبلاً بیان شد، در این تحقیق از روش SPR جهت بهبود نتایج تحلیل اجزای محدود استفاده‌شده است. این روش بر پایه استفاده از «نقاط فوق همگرا[3]» بناشده است. نقاط فوق همگرا، نقاطی هستند که متغیر میدان در آن نقاط، دقت بیشتری نسبت به سایر نقاط دارد. در این نقاط، مرتبه همگرایی، حدوداً یک‌مرتبه از مقدار تقریب تابع شکل بالاتر است. این ویژگی اولین بار توسط بارلو مطرح شد[۳۲]. نقاط فوق همگرا برای گرادیان‌های جابه‌جایی مانند تنش و کرنش، نقاط گاوس هستند که به‌عنوان نقاط نمونه‌برداری مورداستفاده قرار می‌گیرند.

SPR از روش بازیابی برای بالا بردن دقت و هموارسازی گرادیان‌های جابه‌جایی استفاده می‌کند. این عملگر نیازمند تعریف یک وصله برای هر نقطه دلخواه است. روش تشکیل این وصله[4] در نسخه اصلی این روش بر مبنای یافتن یک المان از شبکه قدیم است که شامل آن نقطه گاوس از شبکه جدید باشد. پس از یافتن این المان در شبکه قدیم تمام المان‌هایی که دارای مرز مشترک یا گره مشترک با این المان هستند یافت می‌شوند. به مجموعه این المان‌ها وصله می‌گویند. به کمک مختصات نقاط گاوس داخل این وصله، می‌توان یک‌چند جمله‌ای به مقادیر تنش و کرنش برازش داد. سپس به کمک این چندجمله‌ای مقادیر تنش و کرنش را در نقاط گاوس شبکه جدید محاسبه نمود. اگرچه این روش ساده به نظر می‌رسد اما در پیاده‌سازی به مشکلاتی بر خواهد خورد. برای مثال در مرزهای دامنه تحلیل ممکن است تعداد نقاط گاوس به‌قدر کافی نباشد تا بتوان یک‌چند جمله‌ای مناسب برازش داد.

جهت بهبود این روش، در تحلیل حاضر به‌جای تشکیل وصله، به شکل نسخه اصلی روش، نزدیک‌ترین K نقطه گاوس از شبکه قدیم که در میان تمام نقاط گاوس شبکه قدیم کمترین فاصله را با نقطه گاوس شبکه جدید دارند استفاده خواهد شد. این اصلاح در تشکیل وصله علاوه بر حل مشکل پیش‌گفته قابلیت موازی‌سازی مؤثر روی کارت گرافیک را دارا است.

درروش SPR می‌توان تنش بازیابی شده را به‌صورت زیر نوشت[۳۳-۳۵]:

$$\boldsymbol{\sigma}_{\mathbf{p}}^{*} = \mathbf{p}\mathbf{a} \tag{۵}$$

$$\mathbf{a} = [a_1, a_2, a_3, \ldots, a]^T \tag{۶}$$

$\mathbf{p}$ شامل پارامترهای چندجمله‌ای مناسب برای المان است که برای المان دوبعدی به فرم کلی زیر نوشته می‌شود:

$$\mathbf{p} = [1, x, y, x^2 + 2xy + y^2 + \cdots + y^p] \tag{۷}$$

برای به دست آوردن پارامترهای میدان تنش بازیابی شده بر روی نقطه فوق همگرای در حال بررسی، به کمک معادلات فوق، پارامترهای میدان تنش بازیابی شده در آن نقطه به دست می آید. برای برازش منحنی با دقت بیشتر، باید از تعداد عبارات بیشتری از چند جمله $\mathbf{p}$ استفاده کرد. برای به دست آوردن میدان تنش بهبودیافته با پیوستگی از نوع C_0، باید از ترم‌های چندجمله‌ای $[1, x, y]$ استفاده کرد. در این صورت، ماتریس مجهولات به‌صورت $[a_0, a_1, a_2]$ خواهد بود. در این نوع پیوستگی، حداقل به سه نقطه گاوس در

[1] Recovery
[2] Superconvergent Patch Recovery
[3] Superconvergence points
[4] Patch

المان وصله برای برازش این چندجمله‌ای نیاز است. دقت چندجمله‌ای از مرتبه اول می‌باشد. طبیعتاً برای بالا بردن دقت برازش، به تعداد بیشتر از نقاط فوق همگرا نیاز خواهد بود. برای به دست آوردن مقادیر ضرایب مجهول **a**، باید تابع خطای زیر را کمینه کرد:

$$F(\mathbf{a})=\sum_{i=1}^{n}\left(\boldsymbol{\sigma}_h\left(x_i,y_i\right)-\boldsymbol{\sigma}_p^*\left(x_i,y_i\right)\right)^2=\sum\left(\boldsymbol{\sigma}_h\left(x_i,y_i\right)-\mathbf{p}\left(x_i,y_i\right)\mathbf{a}\right)^2 \qquad (۸)$$

در معادله بالا، $\left(x_i,y_i\right)$ مختصات نقاط گاوس است.

برای کمینه شدن $F(\mathbf{a})$ باید ضرایب مجهول **a** در معادله زیر صدق کند:

$$\sum_{i=1}^{n}\mathbf{p}^T\left(x_i,y_i\right)\mathbf{p}\left(x_i,y_i\right)\mathbf{a}=\sum_{i=1}^{n}\mathbf{p}^T\left(x_i,y_i\right)\boldsymbol{\sigma}_h\left(x_i,y_i\right) \qquad (۹)$$

اگر معادله فوق، به شیوه ماتریسیِ $\mathbf{Aa}=\mathbf{b}$ بازنویسی شود، پارامترهای این معادله، به‌صورت زیر به دست خواهد آمد:

$$\mathbf{A}=\sum_{i=1}^{n}\mathbf{p}^T\left(x_i,y_i\right)\mathbf{p}\left(x_i,y_i\right) \qquad (۱۰)$$

$$\mathbf{b}=\sum_{i=1}^{n}\mathbf{p}^T\left(x_i,y_i\right)\boldsymbol{\sigma}_h\left(x_i,y_i\right) \qquad (۱۱)$$

برای به دست آوردن ضرایب مجهول، باید دستگاه $\mathbf{Aa}=\mathbf{b}$ حل شود. با حل دستگاه فوق، مقادیر ضرایب مجهول به دست می‌آید و با جایگذاری آن در معادله (۵)، میدان بازیابی شده حاصل می‌شود. با جایگذاری نقاط گاوس شبکه قدیم در میدان بازیابی شده، تنش و کرنش بازیابی شده در نقاط گاوس شبکه جدید به دست می‌آید. علاوه بر این، می‌توان به کمک مقادیر بهبودیافته تنش و کرنش در شبکه قدیم با مقادیر متناظر آن‌ها که از تحلیل اجزای محدود حاصل‌شده بود مقدار خطای تقریبی در شبکه قدیم را هم برآورد نمود.

۴- تولید و ریزسازی شبکه جدید

برای کاهش خطای گسسته‌سازی به‌صورت بهینه باید در هر مرحله از حل تطابقی، از شبکهٔ جدیدی استفاده کرد که دارای توزیع یکنواخت‌تر خطا نسبت به مرحله قبل باشد. برای تولید این شبکه جدید در هر مرحله، باید نخست مقدار نرم طولی جابه‌جایی را در هر گره از شبکه قدیم با کمک معادله زیر محاسبه کرد[۳۶]:

$$\varphi=\sqrt{\mathbf{u}^T\mathbf{u}} \qquad (۱۲)$$

در معادله فوق، **u** بردار جابه‌جایی هر نقطه از شبکه قدیم است. سپس مقدار متغیر r که به‌صورت زیر تعریف می‌شود در کل دامنه مسئله محاسبه می‌شود.

$$r=h_{old}\left|\frac{\partial\varphi^h}{\partial\tilde{x}}\right|_{\max} \qquad (۱۳)$$

در معادله فوق h_{old} اندازه المان شبکه قدیم و $\frac{\partial\varphi^h}{\partial\tilde{x}}$ گرادیان جابه‌جایی مطلق در هر گره است. درنهایت اندازه المان شبکه جدید از رابطه زیر به دست خواهد آمد.

$$h_{new}=\left(\beta r_{\max}\right)\Big/\left(\left|\frac{\partial\varphi^h}{\partial\tilde{x}}\right|_{\max}\right) \qquad (۱۴)$$

که در آن، β مقداری اختیاری در بازهٔ ۰٫۱ تا ۰٫۴ است[۳۶].

۵- تخمین خطا به روش بازیابی

یکی از مهم‌ترین بخش‌های حل مسائل اجزای محدود، تخمین خطا است. تخمین خطا وظیفهٔ ارزیابی دقت حل مسائل اجزای محدود را بر عهده دارد. روش‌های تخمین خطا در محاسبات اجزای محدود به دودستهٔ تکنیک باقی‌مانده و تکنیک بازیابی تقسیم می‌شود.

در تکنیک باقی‌مانده، تخمین خطا را با جایگذاری پاسخ‌های حاصله از محاسبات اجزای محدود درون معادله دیفرانسیلی حاکم بر مسئله به دست می‌آورند. ولی در تکنیک بازیابی، تخمین خطا از مقایسه مقدار بهبودیافته میدان گرادیان‌های جابه‌جایی با مقادیر حاصله از محاسبات اجزای محدود محاسبه می‌شود. می‌توان با ساده‌سازی معادلات حاکم بر مسئله، رابطه میان مقادیر حاصله از محاسبات اجزای محدود با خطا را به‌صورت زیر نشان داد:

$$\int_{\Omega} \mathbf{B}^T \mathbf{e}_\sigma d\Omega = 0 \qquad (۱۵)$$

در معادله فوق، e_σ خطای تنش، $\mathbf{B}$ ماتریس متعارف در اجزای محدود و Ω دامنه حل مسئله است. به‌این‌ترتیب، سعی بر آن است که میزان خطای انرژی حداقل شود. معادله خطای انرژی در مسائل خطی را می‌توان به‌صورت زیر تعریف کرد[۳۷]:

$$e_\sigma = \left(\int_{\Omega} \left(\boldsymbol{\sigma}^* - \boldsymbol{\sigma}_h\right)^T \mathbf{D}_e^{-1} \left(\boldsymbol{\sigma}^* - \boldsymbol{\sigma}_h\right) d\Omega \right)^{\frac{1}{2}} \qquad (۱۶)$$

در معادله فوق $\boldsymbol{\sigma}_h$ پاسخ به‌دست‌آمده از محاسبات اجزای محدود، $\boldsymbol{\sigma}^*$ میدان تنش بهبودیافته و $\mathbf{D}_e$ مدول الاستیته ماده است. در مسائل غیرخطی، معادله خطای انرژی را می‌توان به‌صورت زیر نوشت[۲۴]:

$$e_\sigma = \left(\int_{\Omega} \left(\boldsymbol{\sigma}^* - \boldsymbol{\sigma}_h\right)^T \left(\Delta\boldsymbol{\varepsilon}^* - \Delta\boldsymbol{\varepsilon}_h\right) d\Omega \right)^{\frac{1}{2}} \qquad (۱۷)$$

در معادله فوق، $\boldsymbol{\sigma}_h$ پاسخ حاصله از محاسبات اجزای محدود در آخرین مرحله زمانی، $\boldsymbol{\sigma}^*$ میدان تنش بهبودیافته در آخرین مرحله زمانی و $\Delta\boldsymbol{\varepsilon}^*$ تفاوت میدان کرنش بهبودیافته در مراحل زمانی متوالی و $\Delta\boldsymbol{\varepsilon}_h$ تفاوت کرنش حاصله از محاسبات اجزای محدود در مراحل زمانی متوالی است. مقدار $\Delta\boldsymbol{\varepsilon}_h$ را می‌توان از رابطه زیر محاسبه کرد:

$$\Delta\boldsymbol{\varepsilon}_h = \mathbf{B}\left(\bar{\mathbf{u}}_n - \bar{\mathbf{u}}_{n-1}\right) \qquad (۱۸)$$

درنهایت می‌توان خطای تنش را به‌صورت زیر تعریف نمود[۳۸]:

$$e_\sigma = \sigma^* - \hat{\sigma} \qquad (۱۹)$$

در معادله فوق، مقدار $\hat{\sigma}$ برابر با مقدار تقریبی تنش حاصله از محاسبات اجزای محدود و σ^* برابر با مقدار تنش بهبودیافته و هموارشده میدان تنش تقریبی است.

۶- ساختار کلی الگوریتم

با این مقدمات، مقادیر متغیرهای داخلی از شبکه قدیم به شبکه جدید انتقال می‌یابند. جمع‌بندی گام‌های صورت گرفته جهت پیاده‌سازی روش به‌صورت زیر قابل ارائه است.

۱) یک شبکه اولیه تولیدشده و مسئله تحلیل می‌شود.
۲) به کمک مقادیر جابجایی در هر گره از شبکه قدیم، مقدار نرم طولی جابجایی یا φ در آن نقطه محاسبه می‌شود.
۳) به کمک مقادیر φ در هر نقطه از شبکه قدیم، جهت‌های هندسی مربوط به بیشترین و کمترین تغییرات تابع φ محاسبه می‌شود.
۴) میزان کشیدگی یا به عبارت دقیق‌تر اندازه h جدید هر المان به کمک این مقادیر کمینه و بیشینه محاسبه می‌شود.

۵) به کمک این اطلاعات شبکه جدید طوری تولید می‌شود که المان‌ها در راستای جهت کمینه‌ی تغییرات φ کشیده شوند و در جهت بیشینه‌ی این تغییرات فشرده گردند. به‌این‌ترتیب اندازه المان‌ها به‌صورت تطابقی محاسبه خواهد شد.

۶) هر گره از شبکه جدید در شبکه قدیم جانمایی می‌شود تا یک المان از شبکه قدیم، این گره را در خود جای دهد. با یافت شدن این المان، مقادیر جابجایی در آن گره‌ی شبکه جدید به کمک توابع شکل المان قدیم محاسبه می‌شود. به‌این‌ترتیب عملگر اول انتقال داده به اتمام می‌رسد.

۷) به ازای هر نقطه گاوس از شبکه جدید، تعداد K نقطه گاوس از شبکه قدیم طوری یافت می‌شوند که این K نقطه کمترین فاصله را با نقطه گاوس شبکه جدید داشته باشند. این روش را K همسایه نقطه گاوس گوییم.

۸) یک وصله شامل این K همسایه نقطه گاوس تشکیل می‌شود. در این وصله مقادیر تنش و کرنش برای نقطه گاوس از شبکه جدید محاسبه و ذخیره می‌شود.

۹) مقادیر بهبودیافته تنش و کرنش برای نقاط گاوس شبکه قدیم به کمک وصله پیش‌گفته و برازش به‌دست‌آمده در گام قبلی محاسبه می‌شود. در اینجا مراحل عملگر دوم به اتمام می‌رسد.

۱۰) به کمک مقادیر بازیابی شده تنش و کرنش در شبکه قدیم و مقادیر محاسباتی آن‌ها که قبلاً به کمک روش اجزای محدود به‌دست‌آمده بود میزان خطای تنش و کرنش محاسبه می‌شود. از این خطا نیز می‌توان جهت یافتن نقاط یا مسیرهای حساس داخل دامنه حل بهره برد. همچنین می‌توان از این مقادیر و خصوصاً خطای کرنش جهت محاسبه طول جدید المان که باید در شبکه جدید تولید شود استفاده نمود. تفاوت این خطا با خطایی که از نرم طولی φ به دست می‌آید این است که خطای کرنش در نقاط گاوس محاسبه می‌شود ولی نرم طولی φ در نقاط گرهی محاسبه خواهد شد. بسته به توانایی های نرم‌افزار مولد شبکه، می‌توان از هر یک از این خطاها استفاده کرد. در اینجا مرحله برآورد خطا و استفاده از آن به اتمام رسیده است.

۱۱) در این گام اطلاعات تحلیل به روش اجزای محدود به شبکه جدید منتقل‌شده است. در اینجا می‌توان بدون تکرار تحلیلی که تاکنون صورت پذیرفته بود، تحلیل را از این مرحله ادامه داد.

۱۲) در این مرحله، شبکه جدید مورد تحلیل قرارگرفته است. اگر مقادیر حداکثر و نحوه توزیع خطا قابل‌قبول نباشد، این شبکه، شبکه قدیم نامیده و الگوریتم تکرار می‌شود.

در پیاده‌سازی الگوریتم‌های پیشنهادی در این پژوهش از زبان برنامه‌نویسی پایتون استفاده‌شده است. به‌طور مشخص، با توجه به ماهیت موازی محاسبات در روش‌های تطابقی و نیاز به تسریع فرایند انتقال داده، از پلتفرم *CUDA* برای بهره‌گیری از قابلیت‌های پردازش همه‌منظوره بر روی پردازنده‌های گرافیکی (*GPGPU*) استفاده‌شده است. برای پیاده‌سازی اپراتورهای انتقال داده بر روی پردازنده گرافیکی، از دو کتابخانه *CuPy* و *Numba* بهره گرفته‌شده است. کتابخانه *CuPy* برای پیاده‌سازی محاسبات ماتریسی و *Numba* برای تسریع اجرای کدهای پایتون بر روی *GPU* از طریق ترجمه‌ی مستقیم توابع به زبان *CUDA* طراحی‌شده‌اند. هر دو این کتابخانه‌ها توسط شرکت *NVIDIA* توسعه‌یافته و به‌صورت گسترده برای پردازش‌های سنگین موازی پیشنهاد می‌شوند.

در این مطالعه، جهت تحلیل و بررسی تأثیر موازی‌سازی، الگوریتم‌های انتقال داده به سه صورت موازی بر روی پردازنده گرافیکی، موازی بر روی پردازنده مرکزی و سریالی بر روی پردازنده مرکزی پیاده‌سازی شده‌اند. در این راستا، از کتابخانه‌ی *NumPy* نیز برای انجام محاسبات برداری و ماتریسی در بخش پردازنده مرکزی استفاده‌شده است. به‌منظور اجرای تحلیل اجزای محدود اولیه و استخراج داده‌های شبکه، از نرم‌افزار آباکوس استفاده‌شده است. برنامه توسعه داده‌شده، داده‌های تحلیلی و شبکه اولیه را به‌صورت ورودی‌های استاندارد از خروجی‌های این نرم‌افزار دریافت می‌کند و پس از اعمال الگوریتم تطابقی و انتقال داده، نتایج نهایی را در قالبهای قابل بارگذاری در آباکوس ایجاد می کند.

۷- الگوریتم محاسباتی پیشنهادی بر روی *GPU*

همان‌طور که پیش‌تر گفته شد این فرایند از دو بخش اصلی تشکیل‌شده است: ۱- ریزسازی شبکه المان؛ ۲- انتقال اطلاعات از شبکه قدیم به شبکه جدید. بخش نخست به کمک مقادیر بازیابی شده خطای موجود در تحلیل اجزای محدود شبکه قدیم سعی دارد شبکه جدید را طوری ایجاد نماید که خطای تحلیل اجزای محدود در شبکه جدید یکنواخت‌تر و هموارتر باشد. علاوه بر این مطلوب است که میزان حداکثر خطا در شبکه جدید از مقداری معین که هدف محقق است تجاوز ننماید. بخش دوم نیز برای انتقال متغیر داخلی است. این قسمت از الگوریتم به دو مرحله «تشکیل وصله و برازش منحنی» تقسیم می‌شود. در این پژوهش، در بخش تشکیل وصله، برای افزایش سرعت و انعطاف‌پذیر نمودن اندازه وصله از الگوریتم K-nearest neighbors استفاده‌شده است. یکی از دلایل استفاده از این روش، طبیعت الگوریتم آن و قابلیت موازی‌سازی بهتر است. جهت افزایش دوچندان سرعت، الگوریتم برای پردازنده گرافیکی بهینه‌سازی شد.

بخش تشکیل وصله را می‌توان مطابق جدول ۱ و به‌این‌ترتیب خلاصه کرد: ابتدا آرایه‌ای ساخته می‌شود که تعداد ستون‌های آن، به‌اندازه تعداد نقاط گاوس شبکه جدید و تعداد سطرهای آن، به‌اندازهٔ تعداد نقاط گاوس شبکه قدیم است. هر خانه از این آرایه نماینده فاصله نقطه گاوس «j» ام شبکه جدید تا نقطه گاوس «i» ام شبکه قدیم است. سپس با کمک برنامه نوشته‌شده در، فاصله نقاط گاوس از هم محاسبه و درون آرایه ذخیره می‌شود. توابع برنامه به‌گونه‌ای محاسبات را مدیریت می‌کنند که محاسبات به دست آوردن هر فاصله توسط یک هسته از پردازنده گرافیکی انجام شود و حتی‌المقدور همهٔ این محاسبات یک‌باره صورت گیرد.

جدول ۱: تابع تشکیل وصله بر اساس الگوریتم K-Nearest Neighbors

Patch creator function, based on K-Nearest Neighbors algorithm

```
from numba import cuda
import numpy as np

@cuda.jit('float32(float32,float32,float32,float32)',device=True)
def distance(x1, y1, x2, y2):
    return ((x1 - x2)**2 + (y1 - y2)**2)**0.5

@cuda.jit('void(float32[:],float32[:],float32[:],float32[:],float32[:,:])')
def distance_matrix(x1, y1, x2, y2, result):
    i, j = cuda.grid(2)
    if i < result.shape[0] and j < result.shape[1]:
        result[i, j] = distance(x1[i], y1[i], x2[j], y2[j])
```

در پایان این مرحله، ماتریسی از فاصله‌ها به دست می‌آید. هر سطر از این ماتریس را با کمک الگوریتم Hybrid Sorting که ترکیبی از Quick Sort Algorithm و Merge Sort Algorithm است مرتب کرده و سپس با انتخاب K تا از هر سطر به‌عنوان اندیس‌های نقاط گوسی وصله مربوط به المان آن سطر، وصله را تشکیل می‌دهیم. این فرآیند با کمک کتابخانه *CuPy* بروی پردازنده‌ی گرافیکی به‌صورت بهینه انجام می‌پذیرد.

بعد از تشکیل المان وصله، نوبت به برازش منحنی می‌رسد. در این مرحله برای افزایش دقت برازش منحنی به‌جای استفاده برازش چندجمله‌ای با درجه ثابت از برازش چندجمله‌ای بر روی هر وصله با درجه مناسب مخصوص آن وصله استفاده می‌شود. درجه مناسب هر وصله بر اساس مقایسه مقدار مجذور مربعات خطای نقاط برازش شده با مقدار حقیقی‌شان در هر درجه با درجه قبلی خود مشخص می‌گردد. در صورت افزایش مقدار مجذور مربعات خطا، برای جلوگیری از بیش برازش داده، درجه قبلی به‌عنوان درجه مناسب انتخاب می‌شود. سپس از چندجمله‌ای نهایی بر اساس آن درجه استفاده می‌شود. مقادیر بهبودیافته نتایج تحلیل، از جایگذاری مکان نقطه گرهی شبکه‌ی قدیم درون چندجمله‌ای نهایی به دست می‌آید. این کار برای همهٔ وصله‌ها انجام می‌شود تا مقدار میدان بهبودیافته در همهٔ نقاط گرهی شبکه قدیم به دست آید. در گام بعدی مقادیر جابه‌جایی در هر گره از شبکه جدید به کمک توابع شکل یک المان از شبکه قدیم

محاسبه خواهد شد. این المان همان المانی است که گره مذکور از شبکه جدید در آن المان از شبکه قدیم قرارگرفته است. این محاسبات به‌صورت توزیع‌شده روی پردازنده‌های کارت گرافیک انجام می‌گیرد.

برای انتقال متغیر حالت از شبکه قدیم به شبکه جدید، محاسبات این عملگر به بخش‌های زیر تقسیم می‌شود:

۱) مشخص نمودن مکان هر گره معین از شبکهٔ جدید در شبکه قدیم و تشخیص المانی از شبکهٔ قدیم که آن نقطه در آن المان قرارگرفته است؛
۲) تشکیل ماتریس تابع شکل؛
۳) محاسبات مقدار جابه‌جایی در گره موردنظر.

بخش اول از انتقال متغیر حالت مطابق الگوریتم جدول ۲ صورت می‌گیرد که بر اساس ضرب خارجی بناشده است. الگوریتم بررسی می‌کند که نقطه‌ای درون مثلثی با رئوس معلوم قرار دارد یا خیر. برای اجرای بهینه بر روی پردازنده گرافیکی، پیش‌بینی‌شده است که همهٔ نقاط گرهی شبکه جدید حتی‌المقدور به‌صورت یک‌باره بر روی همهٔ المان‌های شبکه قدیم بررسی شود. جزییات این تابع در جدول ۲ ارائه شده است.

جدول ۲ : تابع تشخیص نقطه درون مثلث

Test if a point is inside a triangle

```
@cuda.jit('int32(float64,float64,float64,float64,float64,\
    float64,float64,float64)',device=True)
def checker(x_a,y_a,x_b,y_b,x_c,y_c,O_x,O_y):
    telerance=1e-12
    if ((x_b-x_a)*(O_y-y_a)-(y_b-y_a)*(O_x-x_a) <= telerance and
    (x_c-x_b)*(O_y-y_b)-(y_c-y_b)*(O_x-x_b) <= telerance and
    (x_a-x_c)*(O_y-y_c)-(y_a-y_c)*(O_x-x_c) <= telerance) or\
    ((x_b-x_a)*(O_y-y_a)-(y_b-y_a)*(O_x-x_a) >= -telerance and
    (x_c-x_b)*(O_y-y_b)-(y_c-y_b)*(O_x-x_b) >= -telerance and
    (x_a-x_c)*(O_y-y_c)-(y_a-y_c)*(O_x-x_c) >= -telerance):
        return 1
    else:
        return -1
```

اجرای این تابع بر روی هسته‌های پردازنده گرافیکی به گونه‌ای مدیریت می‌شود که هر هسته از پردازنده گرافیکی مسئول بررسی یک المان و یک گره باشد.

در بخش بعدی الگوریتم با توجه به خطی بودن المان مثلثی می‌توان توابع ماتریس تابع شکل را به‌گونه‌ای برای المان‌ها محاسبه کرد که هر هسته از پردازنده‌های گرافیکی مسئول اجرای محاسبه یک المان باشد و همهٔ محاسبات این بخش حتی‌المقدور به‌صورت یک‌باره انجام شود. ذیلاً بخشی از پیاده‌سازی ارائه می‌شود.

جدول ۳ تولید توابع شکل المان‌های C_0 مثلثی

Creating shape functions for C_0 triangle element

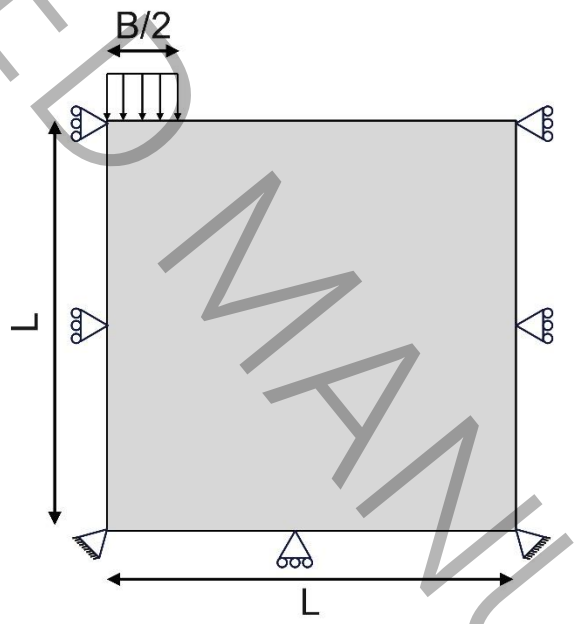
```
@cuda.jit('float64(float64,float64,float64,float64,\
    float64,float64,float64,float64)',device=True)
def calculator_N1(x1,y1,x2,y2,x3,y3,x_A,y_A):
    return (x2*y3-y2*x3-x_A*y3+x_A*y2+y_A*x3-y_A*x2)/\
        (x2*y3-y2*x3-x1*y3+x1*y2+y1*x3-y1*x2)

@cuda.jit('float64(float64,float64,float64,float64,\
    float64,float64,float64,float64)',device=True)
def calculator_N2(x1,y1,x2,y2,x3,y3,x_A,y_A):
    return (x_A*y3-y_A*x3-x1*y3+x1*y_A+y1*x3-y1*x_A)/\
        (x2*y3-y2*x3-x1*y3+x1*y2+y1*x3-y1*x2)
```

```
@cuda.jit('float64(float64,float64,float64,float64,\
    float64,float64,float64,float64)',device=True)
def calculator_N3(x1,y1,x2,y2,x3,y3,x_A,y_A):
    return (x2*y_A-y2*x_A-x1*y_A+x1*y2+y1*x_A-y1*x2)/\
        (x2*y3-y2*x3-x1*y3+x1*y2+y1*x3-y1*x2)
```

اجرای این توابع بر روی هسته‌های پردازنده گرافیکی به‌گونه‌ای مدیریت می‌شود که حتی‌المقدور هر هسته از پردازنده گرافیکی مسئول اجرای محاسبات یک گره باشد. درنهایت با ضرب تابع شکل هر المان در جابه‌جایی نقاط، مقدار جابه‌جایی نقاط گرهی شبکه جدید به دست می‌آید. در بخش ریزسازی شبکه بر اساس رابطه‌ای ذکرشده، مقدار اندازه جدید هر المان را محاسبه کرده و سپس شبکه جدید به کمک نرم‌افزار *GMSH*، بر اساس اندازه‌های جدید تولید می‌شود[۳۹].

۸- شبیه‌سازی عددی

به‌منظور نشان دادن عملکرد مؤثر الگوریتم محاسباتی پیشنهادی و مقایسه صحت و سرعت آن، مسئله‌ای دوبعدی برای بررسی انتخاب شد که یک‌بار توسط پردازنده مرکزی و بار دیگر توسط پردازنده گرافیکی تحلیل و اجرا گردید. مسئله انتخاب‌شده توسط سرورا[1] و همکارانش ارائه‌شده است که شرایط مرزی و هندسی آن، در شکل ۱ دیده می‌شود.



شکل ۱ : هندسه و شرایط مرزی آزمون پانچ

The geometry and boundary conditions for punch test

ویژگی‌های مکانیکی خاک استفاده‌شده در مسئله، عبارت است از:

جدول ۴ ویژگی‌های مکانیکی خاک در آزمون پانچ

Mechanical properties of soil in punch test

ویژگی	واحد	مقدار
ρ	$\frac{\mathrm{kg}}{\mathrm{m}^3}$	۱۰۴

[1] M. Cervera

E	kPa	۱۰^{۷}
ν	-	۰/۴۸
C_0	kPa	۴۹۰
φ	Degree	۲۰

در جدول ۴، ρ بیانگر چگالی، E نشان‌دهنده‌ی مدول یانگ، ν نسبت پوآسن، C_0 ضریب پیوستگی اولیه خاک و φ زاویه اصطکاک خاک می‌باشد. در این مسئله، پی به‌عنوان مدلی الاستیک در نظر گرفته‌شده است. مدول الاستیک پی در مقایسه با خاک به‌گونه‌ای در نظر گرفته‌شده است که بتوان پی را صلب فرض کرد.

در ابتدا دامنه حل توسط شبکهٔ اولیه‌ای با تعداد ۱۹۰ گره و تعداد ۳۳۳ المان مثلثی خطی گسسته‌سازی شد. شبکه اولیه در شکل ۲ دیده می‌شود. سپس سه مرحله تظریف شبکه صورت گرفت که به ترتیب درشکل ۳، شکل ۴ و شکل ۵ نمایش داده‌شده است. در این مسئله کرنش به‌عنوان یکی از متغیرهای داخلی و جابه‌جایی گره‌ای به‌عنوان متغیر حالت انتخاب شد. کانتورهای هر متغیر در مراحل الگوریتم موردبررسی قرار گرفت. تخمین خطا به‌منظور کنترل دقت و عملکرد در هر مرحله بر اساس روابط بخش ۵ محاسبه گردید. اطلاعات تخمین خطای نسبی و مطلق در هر مرحله، در جدول ۵ و جدول ۶ آمده است.

قابل‌ذکر است که مقدار بیشینه خطا در جدول ۵ صرفاً برای نشان دادن کاهش آن در مراحل مختلف ارائه‌شده است. این نوع از خطا در تحلیل اجزای محدود ممکن است ملاک مناسبی برای بررسی نتایج نباشد. درواقع بخش‌هایی از دامنه ممکن است به دلیل نحوه مدل‌سازی به‌صورت غیرواقعی خطایی بالاتر را گزارش کنند. برای مثال، تعداد نقاط به‌کاررفته هنگام مدل‌سازی اتصال پی به خاک باعث ایجاد این نوع خطا می‌شود. لذا تمرکز بر مقدار میانگین خطا و انحراف از معیار آن است. همان‌طور که در جدول ۵ مشاهده می‌شود، با تظریف بهتر شبکه، نه‌تنها میانگین خطا کاهش‌یافته است بلکه خطا به‌صورت یکنواخت‌تر در دامنه توزیع‌شده است چراکه انحراف از معیار خطا نیز کاهش‌یافته است.

جدول ۵ : خطای نسبی هنگام تظریف شبکه

Relative errors during mesh refinement

مراحل ریزسازی	بیشترین مقدار خطای نسبی	میانگین مقدار خطای نسبی	انحراف معیار خطای نسبی
شبکهٔ اولیه	۱۰۰/۳۴٪	۲۷/۱۰٪	۲/۷۹×۱۰$^{-۱}$
مرحله اول	۸۴/۲۲٪	۱۱/۱۲٪	۱/۵۳×۱۰$^{-۱}$
مرحله دوم	۷۹/۰۰٪	۶/۶۴٪	۱/۰۸×۱۰$^{-۱}$
مرحبه سوم	۶۸/۲۰٪	۳/۶٪	۷/۱۸×۱۰$^{-۲}$

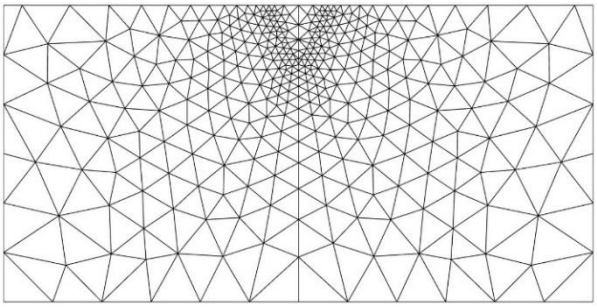
جدول ۶ نشان می‌دهد که چگونه مقادیر مطلق خطا هنگام تظریف شبکه به‌صورت مناسب کاهش‌یافته‌اند. ارائه هم‌زمان جدول ۵ و جدول ۶ تصویر بهتری از خطای مطلق و نسبی و چگونگی توزیع آن‌ها در دامنه به دست می‌دهد. قابل ذکر است که بروز خطای نسبی، در بیشتر مواقع در نقاط گوشه دامنه حل ایجاد می شد که دلیل آن کاهش تعداد نقاط گاوس مورد نیاز بوده است. بیشتزین خطای نسبی در این حالات، اکثرا ناشی از تقسیم اعداد نزدیک به صفر به یکدیگر بوده است.

جدول ۶ : خطاهای مطلق هنگام تظریف شبکه

Absolute errors during mesh refinement

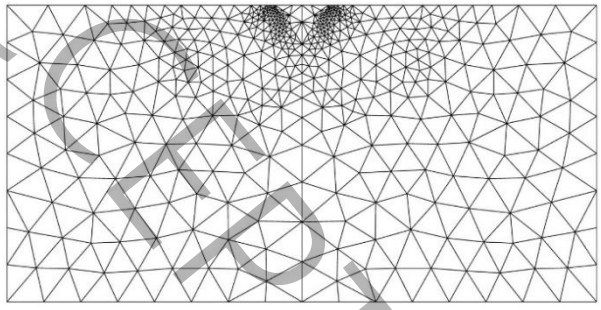
مراحل ریزسازی	بیشترین مقدار خطای مطلق	خطای مجذور میانگین مربعات *RMSE*	میانگین مربع خطای پیش‌بینی *MSE*	میانگین مقدار خطای مطلق *MAE*	انحراف معیار خطای مطلق
شبکهٔ اولیه	۶/۶۷×۱۰$^{-۱}$	۱/۲۸×۱۰$^{-۱}$	۱/۶۵×۱۰$^{-۲}$	۵/۷۰×۱۰$^{-۲}$	۱/۱۲×۱۰$^{-۱}$

مرحله اول	$۲/۷۳\times۱۰^{-۱}$	۳.۹۵×۱۰^{-۲}	$۱/۵۶\times۱۰^{-۳}$	$۲/۰۶\times۱۰^{-۲}$	$۳/۲۲\times۱۰^{-۲}$
مرحله دوم	$۲/۱۳\times۱۰^{-۱}$	$۳/۶۹\times۱۰^{-۲}$	$۱/۳۶\times۱۰^{-۳}$	$۱/۵۷\times۱۰^{-۲}$	$۲/۴۵\times۱۰^{-۲}$
مرحبه سوم	$۱/۸۲\times۱۰^{-۱}$	$۳/۵۴\times۱۰^{-۲}$	$۱/۲۵\times۱۰^{-۳}$	$۱/۰۲\times۱۰^{-۲}$	$۱/۹۰\times۱۰^{-۲}$



شکل ۲ : شبکهٔ اولیه در آزمون پانچ

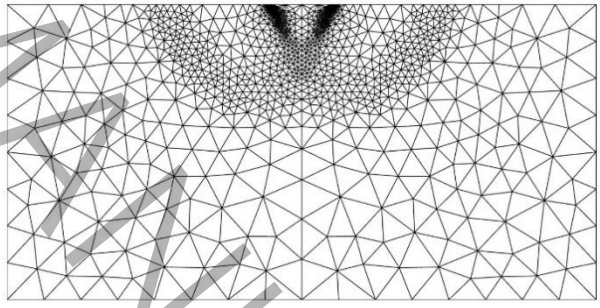
Initial mesh in punch test



شکل ۳ : مرحلهٔ اول تظریف شبکه

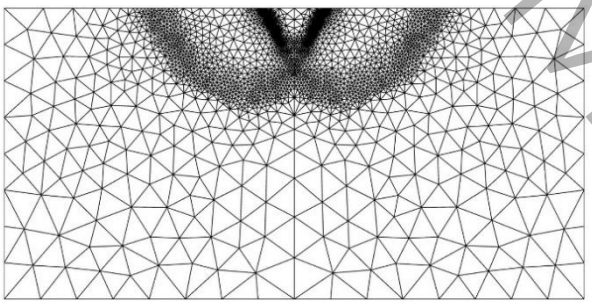
First refined mesh

همان‌گونه که در شکل ۳، شکل ۴ و نهایتاً شکل ۵ قابل‌مشاهده است، مسیر بحرانی محلی شدن کرنش در حال شکل گرفتن است و الگوریتم به‌درستی توانسته است مسیر واقعی بحرانی را نشان دهد. علاوه بر این، الگوریتم به‌صورت تطابقی نسبت به افزایش ابعاد المان‌های دور از این مسیر اقدام کرده است. البته الگوریتم قادر است ابعاد المان‌ها را بیش ازآنچه در اشکال آمده است افزایش دهد بااین‌همه افزایش بی‌رویه ابعاد باعث ایجاد نوع دیگری از خطاهای گسستگی می‌شود که در آن، المان‌های خیلی بزرگ به المان‌های خیلی کوچک متصل شده‌اند.



شکل ۴ : مرحلهٔ دوم تظریف شبکه

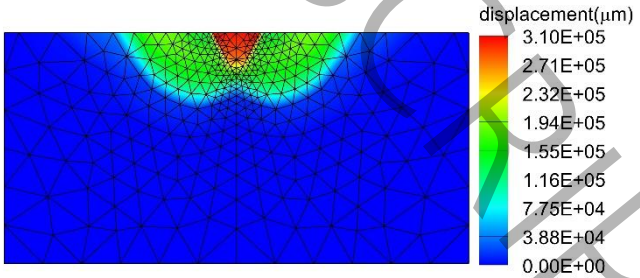
Second refined mesh



شکل ۵ : مرحلهٔ سوم تظریف شبکه

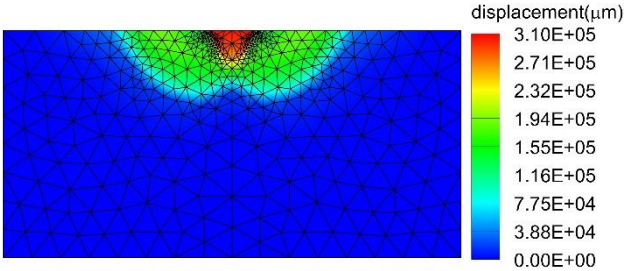
Third refined mesh

کانتور جابه‌جایی شبکه ابتدایی در شکل ۶ و کانتور جابه‌جایی شبکه‌های ریزسازی‌شدهٔ مرحله اول تا سوم به ترتیب درشکل ۷، شکل ۸ و شکل ۹ دیده می‌شود.



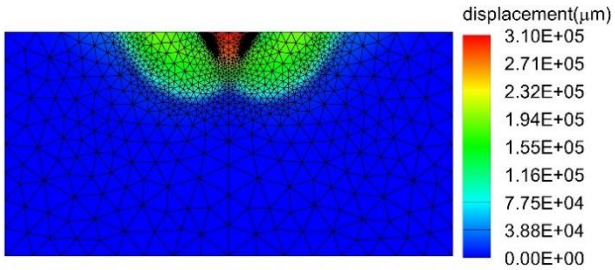
شکل ۶ : جابه‌جایی در شبکهٔ اولیه

Initial mesh displacement



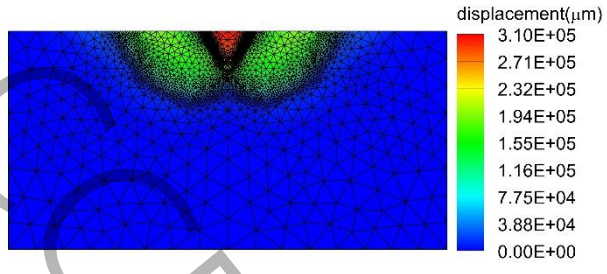
شکل ۷ : جابه‌جایی درتظریف اول

Displacement in 1st refinement



شکل ۸ : جابه‌جایی در تظریف دوم

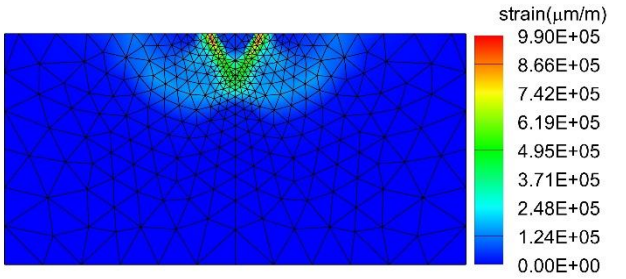
Displacement in 2nd refinement



شکل ۹ : جابه‌جایی در تظریف سوم

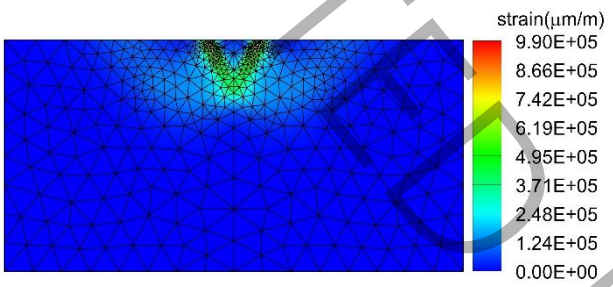
Displacement in 3rd refinement

کانتورهای کرنش در شبکه ابتدایی در شکل ۱۰ و کانتورهای کرنش شبکه‌های ریز شده مرحله اول تا سوم به ترتیب در شکل ۱۱، شکل ۱۲ و شکل ۱۳ نشان داده‌شده است.



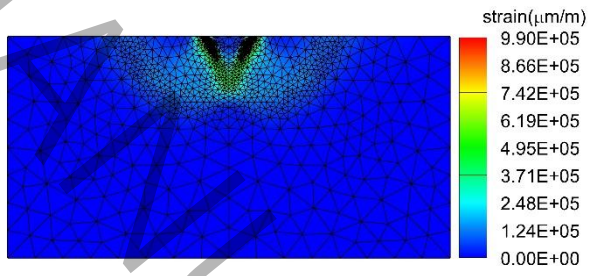
شکل ۱۰ : بیش‌ترین کرنش در شبکهٔ اولیه

Maximum strain in initial mesh



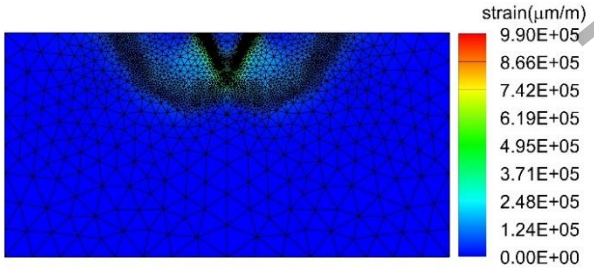
شکل ۱۱ : بیش‌ترین کرنش در تظریف اول

Maximum strain in 1st refinement



شکل ۱۲ : بیش‌ترین کرنش در تظریف دوم

Maximum strain in 2nd refinement



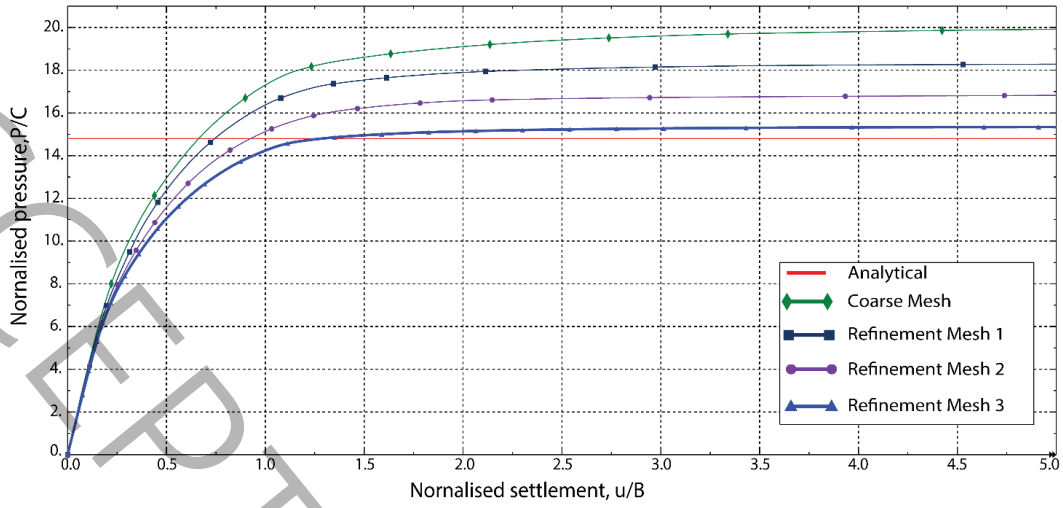
شکل ۱۳ : بیش‌ترین کرنش در تظریف سوم

Maximum strain in 3rd refinement

همان‌طور که از اشکال مذکور می‌توان دریافت، الگوریتم به‌طور صحیح مسیر محلی شدن کرنش یعنی متمرکز شدن مقادیر قابل‌توجه کرنش در یک ناحیه به‌خصوص را تشخیص داده است. این ناحیه با آنچه‌که از واقعیت دریافت شده است مطابقت دارد و نشان‌دهنده مسیری است که خاک نشست خواهد کرد.

به‌منظور صحت سنجی الگوریتم ارائه‌شده در این مقاله، نمودار نیرو-جابه‌جایی شبکه‌های تظریف در شکل ۱۴ نشان داده‌شده است. مقدار بار محدود پلاستیک[1] محاسبه‌شده با روش آنالیز حد[2] (مرجع[۴۰]) مقدار نظری فشار نرمالیزه شده p/c برابر۱۴,۸ می‌باشد. همان‌طور که شکل ۱۴ مشاهده می‌شود شبکه ریزسازی شده در مرحله سوم نسبت به شبکه اولیه نتایج دقیق‌تری را پیش‌بینی کرده است.

[1] Plastic limited load
[2] Method of limited analysis



شکل ۱۴ : نمودار نیرو-جابه‌جایی مدلها در آزمون پانچ

Force-Displacement diagram for all models in punch test

۹- کارایی و سرعت‌افزایی الگوریتم

برای بررسی و مقایسه کارایی و سرعت اجرای الگوریتم ارائه‌شده در مقاله، الگوریتم یک‌بار بر روی پردازنده مرکزی به‌صورت سری و بار دیگر به‌صورت موازی بر روی پردازندهٔ گرافیکی اجرا گردید. سپس نتایج سرعت‌افزایی[1] آن مقایسه شد. سخت‌افزاری که این الگوریتم بر روی آن اجرا شد دارای پردازنده مرکزی intel Xeon با توان پردازشی ۲٫۵ گیگاهرتز و پردازنده گرافیکی Nvidia Quadro P4000 با ۱۷۹۲ هسته کودا و با حافظه دسترسی تصادفی ۳۰ گیگابایت بود که نتایج اجرا در هر مرحله در جدول ۷ ارائه‌شده است.

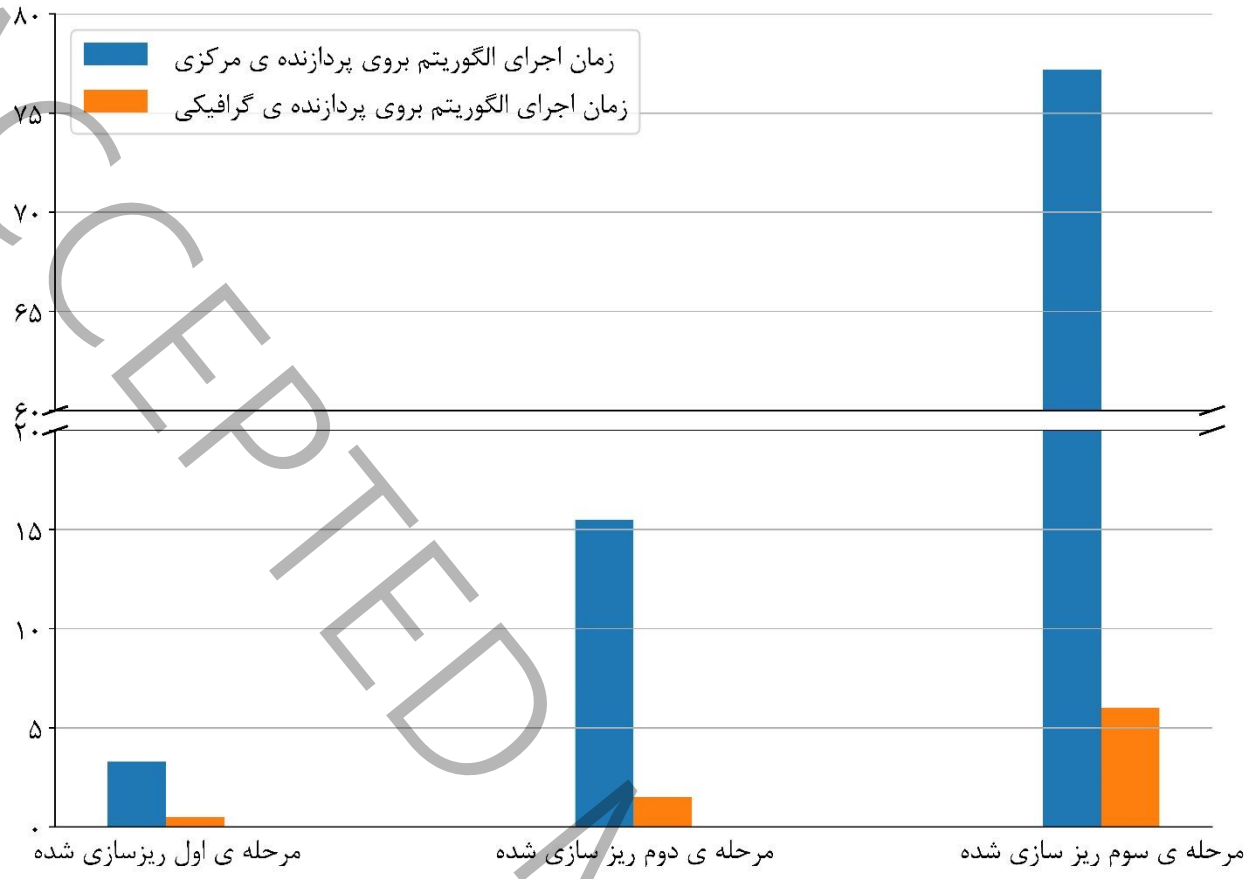
جدول ۷ : زمان اتمام فرآیند انتقال روی پردازنده مرکزی و پردازندهٔ گرافیکی

Elapsed time using CPU and GPU

مراحل ریزسازی	پردازنده گرافیکی (ثانیه)	پردازنده مرکزی (ثانیه)
اول	۰/۵	۳/۳
دوم	۱/۷	۱۵/۵
سوم	۶/۱	۷۷/۲

همان‌طور که از جدول ۷ مشاهده می‌شود، در پیچیده‌ترین حالت زمان کلی که برای اجرای الگوریتم در پردازش به‌صورت موازی بر روی پردازنده گرافیکی صرف شده است تقریباً ۶ ثانیه و متناظر همین مقدار در حالت اجرای سری الگوریتم بر روی پردازنده مرکزی ۷۷ ثانیه است که بیش از ۱۲ برابر زمان صرف شده در حالت موازی می باشد. نمایش گرافیکی مقایسهٔ زمان اجرای الگوریتم به‌صورت موازی بر روی پردازنده گرافیکی و حالت سریال در شکل ۱۵ آمده است.

[1] Speed up



شکل ۱۵ : مقایسه زمان اجرای الگوریتم روی کارت گرافیک و حالت سری روی پردازنده مرکزی

Comparing the elapsed time on CPU and GPU

۱۰- جمع‌بندی

اگرچه روش تطابقی در آنالیز اجزای محدود شیوه‌ای کارآمد است، اما خود بر پیچیدگی محاسبات می‌افزاید. برای استفاده بهینه از قابلیت‌های تحلیل تطابقی باید به نحوی سرعت اجرای این الگوریتم را بالاتر برد. پیاده‌سازی مستقیم این الگوریتم روی کارت گرافیک ممکن است در تلاش اول به نتیجه نرسد چراکه الگوریتم نسخه اصلی از نظر فنی قابلیت موازی‌سازی موثر ندارد. برای این منظور تغییراتی در الگوریتم ایجاد شد تا بتواند به‌صورت موازی روی کارت گرافیک اجرا شود. یکی از این تغییرات استفاده از روش K همسایگی است که در این تحقیق به‌صورت K نقطه گاوس استفاده شد. این روش جایگزین روش معمول تشکیل وصله گردید. الگوریتم پیشنهادی که در این تحقیق استفاده شد راهی مؤثر برای کاهش خطای گسسته‌سازی مبتنی بر روش المان محدود تطابقی (AFEM) است. این الگوریتم با گسسته‌سازی هوشمند دامنه مسئله باعث افزایش دقت و بهبود در حل نهایی می‌شود. این الگوریتم به‌صورت پیوسته در هر مرحله، خطای گسسته‌سازی را کاهش می‌دهد و این عمل به‌صورت هوشمند و خودکار صورت می‌گیرد. نتایج تخمین خطا و منحنی نیروی کنترل مسئله آزمون پانچ، نشان‌دهنده صحت عملکرد و کاربرد آن است. میزان توانایی الگوریتم در سرعت‌افزایی در هر مرحله، به تعداد المان‌ها بستگی دارد. با ریزترشدن افزایش تعداد المان‌ها، این الگوریتم کارایی و سرعت بیشتری از خود نشان می‌دهد. به‌طور نمونه در مثال عددی ذکرشده سرعت پردازش در مراحل اول تا سوم ریزسازی به ترتیب ۶,۶، ۹,۱ و ۱۲,۷ برابر حالت سریال شده است. مجموع میزان زمان لازم برای انجام تمام مراحل در حالت سریال ۹۶ ثانیه و در حالت موازی روی کارت گرافیک برابر ۸ ثانیه می باشد. این میزان سرعت افزایی نشان از عملکرد مناسب نرم افزار در انتقال داده ها به صورت موازی است. به‌طور عمومی این الگوریتم با پردازش موازی بر روی پردازنده گرافیکی کارایی و عملکرد بهتری را از خود نشان می‌دهد. اگرچه این تحقیق روی کارت گرافیک نه‌چندان جدید انجام شد

بااین‌همه می‌توان انتظار داشت که با افزایش قابلیت‌های سخت‌افزاری نظیر کارت گرافیک جدیدتر بتوان به مقادیر بالاتر سرعت‌افزایی رسید. این الگوریتم می‌تواند ابزار قدرتمندی برای پیش و پس پردازش مسائل مهندسی و علوم محاسباتی باشد.

۱۱- مراجع

[1] O.C. Zienkiewicz, J.Z. Zhu, A simple error estimator and adaptive procedure for practical engineerng analysis, in, 1987, pp. 337-357.
[2] L.Y. Li, P. Bettess, J.W. Bull, T. Bond, I. Applegarth, Theoretical formulations for adaptive finite element computations, Communications in Numerical Methods in Engineering, 11(10) (1995) 857-868.
[3] J. Grandy, Conservative remapping and region overlays by intersecting arbitrary polyhedra, Journal of Computational Physics, 148(2) (1999) 433-466.
[4] X. Jiao, M.T. Heath, Common-refinement-based data transfer between non-matching meshes in multiphysics simulations, International Journal for Numerical Methods in Engineering, 61(14) (2004) 2402-2427.
[5] T. Arbogast, L.C. Cowsar, M.F. Wheeler, I. Yotov, Mixed finite element methods on nonmatching multiblock grids, SIAM Journal on Numerical Analysis, 37(4) (2000) 1295-1315.
[6] M.M. Rashid, Material state remapping in computational solid mechanics, International Journal for Numerical Methods in Engineering, 55 (2002) 431-450.
[7] M. Ortiz, J.J. Quigley IV, Adaptive mesh refinement in strain localization problems, Computer Methods in Applied Mechanics and Engineering, 90 (1991) 781-804.
[8] G.T. Camacho, M. Ortiz, Computational modelling of impact damage in brittle materials, International Journal of solids and structures, 33(20-22) (1996) 2899-2938.
[9] N.-S. Lee, K.-J. Bathe, Error indicators and adaptive remeshing in large deformation finite element analysis, Finite Elements in Analysis and Design, 16(2) (1994) 99-139.
[10] B. Boroomand, O.C. Zienkiewicz, Recovery procedures in error estimation and adaptivity. Part II: Adaptivity in nonlinear problems of elasto-plasticity behaviour, Computer Methods in Applied Mechanics and Engineering, 176 (1999) 127-146.
[11] M. Kitamura, H. Gu, H. Nobukawa, A study of applying the superconvergent patch recovery (SPR) method to large deformation problem, Journal of the Society of Naval Architects of Japan, 2000(187) (2000) 201-208.
[12] X. Tang, T. Sato, Adaptive mesh refinement and error estimate for 3-D seismic analysis of liquefiable soil considering large deformation, Journal of natural disaster science, 26(1) (2004) 37-48.
[13] H. Gu, Z. Zong, K.C. Hung, A modified superconvergent patch recovery method and its application to large deformation problems, Finite Elements in Analysis and Design, 40 (2004) 665-687.
[14] A. Khoei, S. Gharehbaghi, Modelling of localized plastic deformation via the adaptive mesh refinement, International Journal of Nonlinear Sciences and Numerical Simulation, 4(1) (2003) 31-46.
[15] S.A. Gharehbaghi, A.R. Khoei, Three-dimensional superconvergent patch recovery method and its application to data transferring in small-strain plasticity, Computational Mechanics, 41 (2008) 293-312.
[16] A.R. Khoei, S.A. Gharehbaghi, Three-dimensional data transfer operators in large plasticity deformations using modified-SPR technique, Applied Mathematical Modelling, 33 (2009) 3269-3285.
[17] A.R. Khoei, S.A. Gharehbaghi, A.R. Tabarraie, A. Riahi, Error estimation, adaptivity and data transfer in enriched plasticity continua to analysis of shear band localization, Applied Mathematical Modelling, 31 (2007) 983-1000.

[18] A.R. Khoei, S.A. Gharehbaghi, A.R. Azami, A.R. Tabarraie, SUT-DAM: An integrated software environment for multi-disciplinary geotechnical engineering, in, Elsevier, 2006, pp. 728-753.
[19] J. Peddie, The History of the GPU - New Developments, in, Springer International Publishing, 2023, pp. 1-410.
[20] K. Proudfoot, W.R. Mark, S. Tzvetkov, P. Hanrahan, A real-time procedural shading system for programmable graphics hardware, in, Association for Computing Machinery, 2001, pp. 159-170.
[21] M. Kronbichler, K. Ljungkvist, Multigrid for Matrix-Free High-Order Finite Element Computations on Graphics Processors, in, ACM PUB27 New York, NY, USA 2019.
[22] Y. Zhang, X. Yan, X. Ren, S. Wang, D. Wu, B. Bai, Parallel implementation and branch optimization of EBE-FEM based on CUDA platform, in, 2020, pp. 595-600.
[23] J. Zhang, D. Shen, GPU-based implementation of finite element method for elasticity using CUDA, in, IEEE Computer Society, 2014, pp. 1003-1008.
[24] S.A. Gharehbaghi, A.R. Khoei, Three-dimensional superconvergent patch recovery method and its application to data transferring in small-strain plasticity, in, Springer Verlag, 2008, pp. 293-312.
[25] O.C. Zienkiewicz, R.L. Taylor, J.Z. Zhu, , The finite element method [electronic resource] : its basis and fundamentals / O.C. Zienkiewicz, R.L. Taylor, J.Z. Zhu., in, Amsterdam ; Boston : Elsevier Butterworth-Heinemann, 2005., 2005.
[26] A.R. Khoei, A.R. Tabarraie, S.A. Gharehbaghi, H-adaptive mesh refinement for shear band localization in elasto-plasticity Cosserat continuum, in, Elsevier, 2005, pp. 253-286.
[27] A.R. Khoei, S.A. Gharehbaghi, The superconvergence patch recovery technique and data transfer operators in 3D plasticity problems, in, Elsevier, 2007, pp. 630-648.
[28] A.R. Khoei, S.A. Gharehbaghi, A.R. Tabarraie, A. Riahi, Error estimation, adaptivity and data transfer in enriched plasticity continua to analysis of shear band localization, in, Elsevier, 2007, pp. 983-1000.
[29] A.R. Khoei, S.A. Gharehbaghi, Three-dimensional data transfer operators in large plasticity deformations using modified-SPR technique, in, Elsevier, 2009, pp. 3269-3285.
[30] B. Boroomand, O.C. Zienkiewicz, Recovery procedures in error estimation and adaptivity. Part II: Adaptivity in nonlinear problems of elasto-plasticity behaviour, in, North-Holland, 1999, pp. 127-146.
[31] O.C. Zienkiewicz, B. Boroomand, J.Z. Zhu, Recovery procedures in error estimation and adaptivity Part I: Adaptivity in linear problems, Computer Methods in Applied Mechanics and Engineering, 176(1-4) (1999) 111-125.
[32] J. Barlow, Optimal stress locations in finite element models, in, John Wiley & Sons, Ltd, 1976, pp. 243-251.
[33] O.C. Zienkiewicz, J.Z. Zhu, The superconvergent patch recovery (SPR) and adaptive finite element refinement, in, 1992, pp. 207-224.
[34] O.C. Zienkiewicz, J.Z. Zhu, The superconvergent patch recovery and a posteriori error estimates. Part 2: Error estimates and adaptivity, in, John Wiley & Sons, Ltd, 1992, pp. 1365-1382.
[35] O.C. Zienkiewicz, J.Z. Zhu, The superconvergent patch recovery and a posteriori error estimates. Part 1: The recovery technique, in, John Wiley & Sons, Ltd, 1992, pp. 1331-1364.
[36] O.C. Zienkiewicz, M. Huang, M. Pastor, Localization problems in plasticity using finite elements with adaptive remeshing, in, John Wiley & Sons, Ltd, 1995, pp. 127-148.
[37] O.C. Zienkiewicz, R.L. Taylor, J.Z. Zhu, The finite element method [electronic resource] : its basis and fundamentals / O.C. Zienkiewicz, R.L. Taylor, J.Z. Zhu., in, Amsterdam ; Boston : Elsevier Butterworth-Heinemann, 2005., 2005.
[38] A.R. Khoei, Computational plasticity in powder forming processes, in, Elsevier, 2005, pp. 449.
[39] C. Geuzaine, J.F. Remacle, Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities, in, John Wiley & Sons, Ltd, 2009, pp. 1309-1331.

[40] M. Cervera, N. Lafontaine, R. Rossi, M. Chiumenti, Explicit mixed strain–displacement finite elements for compressible and quasi-incompressible elasticity and plasticity, in, Springer Verlag, 2016, pp. 511-532.

Two-Dimensional Adaptive Finite Element Using GPGPU

A. H. Khatami[1], S. Asil Gharebaghi [*,1]

[1]Civil Engineering Faculty, K. N. Toosi University of Technology

ABSTRACT

The discretization error is one of the most common errors in the finite element method. One way to reduce this error is by using adaptive methods. Adaptive methods generally involve a large computational load; a technique for reducing this load is the use of data transfer operators. Even with data transfer operators, adaptive methods still require significant time from users. Given the new capabilities provided by graphics processing units (GPUs) for general-purpose computing under the CUDA platform, and the economic efficiency of GPUs compared to standard processors, this paper aims to present an algorithm that can reduce computation time through general-purpose GPU processing. The proposed algorithm begins with an initial finite element analysis using a nearly uniform mesh and refines the mesh intelligently at each step based on the displacement gradient. The conventional algorithm has been improved at several points. The patch formation stage is implemented using the K-nearest neighbor method to facilitate more efficient parallelization. In the data transfer stage, a dynamic method is employed to select the optimal curve from a set of the best curves. The results show that the acceleration of this algorithm increases proportionally with the number of elements. For instance, for a problem with 908 elements, the processing speed for stages one through three increased by factors of 6.6, 9.1, and 12.7, respectively. The total time required for all three stages in serial processing was 96 seconds, which was reduced to 8 seconds using this algorithm.

KEYWORDS

Data Transfer Operator, Adaptive Finite Element Method, GPU, GPGPU, KNN

[1] Corresponding Author: Email: username@EmailServer.com