سنجش و مقایسه‌ی توانایی‌های روش‌های حل دستگاه معادله‌های خطی در زبان‌ها و محیط‌های مختلف برنامه‌نویسی

محمدرضا توکلی‌زاده[1]، مرتضی قالیشویان[2]، مصطفی صالحی احمدآباد[2] و حسین اکرمی‌زاده[2]

1- استادیار گروه مهندسی عمران، دانشگاه فردوسی مشهد

2- دانشجوی کارشناسی ارشد سازه، دانشگاه فردوسی مشهد

drt@um.ac.ir

خلاصه

یکی از چالش‌های پیش‌رو در علوم مهندسی، چگونگی ذخیره‌سازی و حل دستگاه معادلات خطی با تعداد مجهولات بالا می‌باشد. این مساله در مهندسی عمران برای حل دستگاه معادلات حاکم بر رفتار سازه‌هایی که تعداد درجات آزادی بالایی دارند مورد توجه است. در این مقاله، نخست از روش‌های مختلف حل دستگاه معادلات کلی سخن گفته می‌شود. سپس، سرعت حل این روش‌ها و فضای لازم برای ذخیره‌سازی داده‌ها با حل چندین نمونه‌ی عددی مورد مقایسه قرار می‌گیرند. پس از معرفی روش‌های برتر، توانایی عملکرد چندین کامپایلر و محیط برنامه نویسی شناخته شده در حل دستگاه معادلات با یکدیگر مقایسه می‌گردند. محیط‌های مورد بررسی در این مقاله عبارتند از: Fortran Compaq PowerStation, C++, C#, MATLAB, Mathematica. در گام بعد، سرعت عملکرد توابع کتابخانه‌ای محیط‌های برنامه‌نویسی مذکور در حل دستگاه معادلات خطی بررسی می‌شوند. از آنجایی که بیشتر دستگاه معادلات حاکم بر رفتار سازه‌ها از نوع تنک می‌باشند، بررسی شیوه‌های مختلف ذخیره‌سازی این گونه دستگاه معادلات و سرعت حل آن‌ها با یکدیگر نیز مورد مقایسه قرار می‌گیرند.

کلمات کلیدی: دستگاه معادلات خطی، محیط‌های برنامه‌نویسی، سرعت حل، روش ذخیره‌سازی، توابع کتابخانه‌ای

1. مقدمه

مسائل موجود در بسیاری از شاخه‌های علوم مهندسی به حل دستگاه معادلات خطی منتهی می‌گردد. در مهندسی عمران، تحلیل یک سازه نیازمند حل دستگاه معادلات حاکم بر رفتار آن است. به طوریکه، تحلیل سازه‌ی با n درجه‌ی آزادی، حل دستگاه معادله‌ی n مجهولی را به دنبال دارد. تحلیل‌گران روش‌های گوناگونی را برای حل دستگاه معادلات خطی پیشنهاد کرده‌اند. انتخاب روش حل مناسب به‌ویژه در سازه‌های با تعداد درجات آزادی بالا از اهمیت فراوانی برخوردار است. معیارهای انتخاب روش حل، شامل توانایی همگرایی، سرعت انجام محاسبات و میزان حافظه‌ی مورد نیاز برای تحلیل می‌باشند. از سوی دیگر با توجه به پیشرفت علم و ظهور انواع نرم‌افزارها و محیط‌های برنامه نویسی، انجام یک مقایسه‌ی جامع برای بررسی سرعت عملکرد و توانایی این محیط‌ها در حل دستگاه معادلات امری لازم و ضروری است. از سوی دیگر، با توجه به آن‌که در بیشتر مسائل تحلیل سازه با ماتریس ضرایب تنک روبرو هستیم، بررسی انواع راه‌کارهای ذخیره‌سازی ماتریس‌های تنک مورد توجه است.

2. روش‌های حل دستگاه معادلات خطی

از جمله روش‌های شناخته شده در حل دستگاه معادلات خطی شامل: راه‌کارهای حذفی گوس، تجزیه‌ی گوس، تجزیه‌ی چولسکی و روش‌های تکراری ژاکوبی و گوس- سایدل می‌باشند. برای یافتن کارایی روش‌های بالا، افزون بر مقایسه‌ی سرعت حل آنان، عملکرد کامپایلرهای مختلف برنامه‌نویسی نیز مورد سنجش قرار می‌گیرند. این مقایسه، با کدنویسی روش تجزیه‌ی گوس در محیط‌های مزبور صورت می‌پذیرد. همچنین، مقایسه‌ی سرعت حل این کامپایلرها با استفاده از توابع کتابخانه‌ای خود آن‌ها انجام و موارد برتر معرفی می‌شوند.

روش حذفی گوس یکی از قدیمی‌ترین روش‌های حل دستگاه معادلات خطی می‌باشد[1]. روش حذفی گاوس برای محاسبه‌ی رتبه، دترمینان و معکوس ماتریس‌ها نیز مورد استفاده قرار می‌گیرد. این راه‌کار، برای ماتریس‌های معین مثبت و غالب قطری دارای پایداری عددی است. در حالت

کلی نیز با استفاده از فن محور جزئی می‌توان به پایداری عددی رسید[2]. برای حل دستگاه $[A]\{x\}=\{b\}$ با این شیوه، ماتریس افزوده دستگاه را نوشته و با اعمال سطری مقدماتی، آن را به صورت مثلثی در می‌آورند. سپس، محاسبه‌ی مجهولات از آخرین معادله آغاز می‌گردد. عامل‌های $[A]$، $\{x\}$ و $\{b\}$ به ترتیب، ماتریس ضرایب، بردار مجهولات و بردار ثابت‌ها می‌باشند.

شیوه‌ی تجزیه‌ی ماتریس‌ها به وسیله‌ی آلان ماتیسون پیشنهاد گردید[3]. در این روش، ماتریس ضرایب به دو ماتریس پایین مثلثی (L) و بالا مثلثی (U) تجزیه می‌گردد. باید دانست، ماتریس پایین مثلثیدارای درایه‌های قطری واحد و ماتریس بالا مثلثی به شکل کلی است. سپس، با استفاده از عمل جایگزینی پیشرو و پسرو به‌سادگی دستگاه معادلات حل می‌گردد. شایان توجه است کهتجزیه‌ی ماتریس با روش‌های گوناگونی انجام می‌شود که رایج‌ترین آن‌ها شیوه‌ی حذفی گوس است [4].

گونه‌ی دیگر تجزیه‌ی ماتریس ضرایب که در حل دستگاه معادلات خطی به‌کار می‌رود به‌نام روش چولسکی معروف است. در این روش درایه های قطری ماتریس بالا مثلثی واحد می‌باشند. افزون بر آن، درایه های ماتریس‌های افزوده نیز از روابط کلی به دست می‌آیند. مجهولات دستگاه معادلات با رابطه‌های زیر تعیین می‌گردند:

$$\begin{aligned}&[A]\{x\}=\{b\}\\&[L][U]\{x\}=[L]\{c\}=\{b\}\\&[U]\{x\}=\{c\}\end{aligned}\qquad(1)$$

روش ژاکوبی راه‌کارهای غیر قطعی در حل دستگاه معادلات می‌باشد. این شیوه از الگوریتمی تکراری برای همگرایی به پاسخ استفاده می‌کند. در این روش، ابتدا معادلات به گونه‌ای جابه‌جامی‌شود تا درایه های قطری از سایر درایه های هم‌سطر خود بزرگ‌تر باشند. سپس، نخستین مجهول از معادله‌ی اول، دومین مجهول از معادله‌ی دوم و به همین ترتیب سایر مجهول‌ها محاسبه می‌گردند. محاسبات با جایگزینی یک دسته جواب اولیه در سمت راست معادلات ادامه می‌یابد. با این جایگزینی، یک دسته پاسخ دیگر به دست می‌آید. این روند تا یکسان شدن مقادیر جایگزین‌شده و مقادیر محاسبه شده ادامه پیدا می‌کند. باید دانست، دقت خطای تحلیل در این مرحله مشخص می‌شود. روش گوس–سایدل نیز مشابه روش ژاکوبی است، با این تفاوت که در این شیوه، پس از مشخص شدن یک مجهول، در تکرارهای بعدی از آن در محاسبه‌ی سایرمجهولات استفادهمی‌شود.

2–1 بررسی عددی روش‌ها

از آنجا که موضوع حل دستگاه معادلات خطی، بخش بزرگی از محاسبات مهندسی و دیگررشته‌ها را در بر می‌گیرد، در این بخش به بررسیروش‌های مقدماتی ارائه شده در بالا می‌پردازیم. بدین منظور، کد مربوط هر روش، در محیط‌های مختلف برنامه نویسی نوشته شد. در تمامی کدهای نوشته شده سعی بر آن بوده است که عملیات جبری الگوریتم‌ها یکسان باشند. کامپایلرهایی که در این پژوهش مورد بررسی قرار گرفته‌اند، به طور گسترده‌ای در بین پژوهش‌گران مورد استفاده قرار می‌گیرند. جدول (1) کامپایلرها و محیط‌های برنامه نویسی مورد استفاده در این مطالعه را نمایش می‌دهد. شرح این کامپایلرها و محیط‌های برنامه نویسی در ادامه خواهد آمد. همچنین رایانه مورد استفاده در این پژوهش دارای پردازشگرمدلCore i7 2600k با 8 هسته برای پردازش و 4 گیگابایت حافظه جانبی می‌باشد.

متلب (MATLAB) نام یکی از نرم‌افزارهای رایانه‌ای برای انجام محاسبات ریاضی است.تمام داده‌ها در متلب به شکل یک ماتریس ذخیره می‌شوند. هسته‌ی متلب برای سرعت و کارایی بالا به زبان برنامه‌نویسی سی نوشته شده است ولی رابط گرافیکی آن به زبان برنامه‌نویسی جاوا پیاده سازی گشته است.برنامه‌های متلب اکثراً متن‌باز هستند و در واقع متلب (مانند بیسیک) مفسر است نه کامپایلر. انعطاف‌پذیری این نرم‌افزار راحت بودن کار با آن، از مزیت‌های این محیط برنامه‌نویسی می‌باشد. همچنین، هر ساله، شرکت سازنده و گروه‌های مختلف، از جمله دانشگاه‌های سرتاسر جهان و برخی شرکت‌های مهندسی، جعبه ابزارهای خاص–کاربردی به آن می‌افزایند که باعث افزایش کارایی و محبوبیت آن شده‌است. در مقاله‌ی کنونی، از نسخه‌ی متلب 7/2 استفاده شده است.

جدول 1- کامپایلرهای و محیط برنامه نوسی مورد استفاده

کامپایلر	محیط برنامه نویسی
C++	Microsoft Visual Studio 2008
C#	Microsoft Visual Studio 2008
Fortran 90	Microsoft Fortran PowerStation 4.0,Professional Edition
Fortran 95	Compaq Visual Fortran 6.5 ,Professional Edition
-	MATLAB 2008
-	WolfarmMathematica 7.0

متمتیکا (Mathematica)، یک نرم‌افزار جبری بسیار رایج است که اکثر توابع نرم‌افزاری موردنیاز در ریاضی و علوم طبیعی را در اختیار استفاده‌کنندگان آن قرار می‌دهد. از مهم‌ترین قابلیت‌های این نرم‌افزار،می‌توان به دارا بودن سامانه‌ی رایانه‌ای جبری برای بررسی نمادین معادله‌ها، توابع ترسیمی و تجسمی برای رسم نمودارها، توانایی حل عددی معادله‌ها و محیط برنامه نویسی مجزا نام برد.

زبان برنامه‌نویسی C++، یک زبان برنامه‌نویسی رایانه‌ای همه‌منظوره، شیءگرا و نزدیک به زبان سیستم می‌باشد. این زبان از برنامه‌نویسی ساخت‌یافته وشیءگرا پشتیبانی می‌کند.C++ از پرطرفدارترین زبان‌های برنامه‌نویسی تجاری است.

زبان برنامه‌نویسی#C، زبانی شیءگرا و سطح بالا از خانواده‌ی زبان‌های شرکت مایکروسافت است. زبان #C، یک زبان برنامه نویسی چند الگویی است که مدل‌های تابعی، امری، عمومی، شی گرا و جزگرا را پشتیبانی می‌کند. خاطر نشان می‌سازد، محیط این زبان برنامه نویسی ساده و مدرن است.

فرترن (FORTRAN)، زبان برنامه‌نویسی مفسری است. فرترن زبانی ساده و محاسباتی می‌باشد و پروژه‌های بسیاری از رشته‌های فنی مهندسی به کمک این زبان نوشته و اجرا شده‌است. فرترن اولین زبان برنامه نویسی سطح بالا است. نسخه‌های اخیر فرترن بر خلاف نسخه‌های قدیمی دارای قابلیت‌های شی‌گرایی هستند. فرترن دومین زبانی است که از سوی موسسه ملی استاندارد آمریکا، به عنوان یک زبان برنامه نویسی استاندارد شناخته شد.

در ابتدا، برای مقایسه سرعت روش‌های حل دستگاه معادلات خطی،برای تمامی روش‌های مزبور کد در نرم افزار متمتیکا نوشته شد و برای چند دستگاه معادله‌ی با تعداد مجهولات مختلف حل گردید. جدول (2) نتایج این محاسبات را نشان می‌دهد.

جدول 2-زمان و مقدار حافظه اشغال شده برای حل دستگاه معادلات در نرم افزار متمتیکا

توضیحات		روش					
تعداد مجهولات	حافظه مورد استفاده(MB)	تجزیه‌ی گوس	گوس-جردن	چولسکی	گوس-سایدل	ژاکوبی	زمان(ثانیه)
100	<1	0.516	0.447	0.875	1.109	1.2	
500	<1	13.937	14.953	87.531	22.408	24.3	
1000	15	74.156	110.67	*	70.141	90.34	
2500	98	818.453	*	*	*	*	
5000	392	*	*	*	*	*	

* مقدار غیر قابل قبول

نتایج نشان می‌دهد روش تجزیه گوس از سایر روش‌هاسریع‌تر است. روش‌های تکراری ژاکوبی و گوس- سایدل در حالت خاصی که ماتریس ضرایب قطری غالب باشد، به جواب همگرا خواهد شد. همچنین، به دلیل فرآیند تکراری، مدت زمان بیشتری را برای حل دستگاه معادلات نیاز دارد. در حالتی که تعداد معادلات از 2500 بیشتر شود هیچ کدام از روش‌ها در زمان معقول به جواب نمی‌رسند.

در گام بعدی، برای سنجش توانایی کامپایلرها و محیط‌های برنامه نویسی، کد روش تجزیه گوس در محیط‌های مختلف نوشته شد. این محاسبات برای چندین دستگاه معادله‌ی خطی انجام گرفت. جدول (3) نتایج این عملیات را نشان می‌دهد.

این جدول برتری کامپایلر C++ را نسبت به سایر روش‌ها نشان می‌دهد. زبان برنامه نویسی فرترن پاورستیشن در جایگاه دوم قرار می‌گیرد. در این میان، نرم افزار متمتیکا ضعیف‌ترین نتیجه را دارا است.

امروزه نرم افزار های برنامه سازی دارای کتابخانه‌های محاسباتی می‌باشند. یک نمونه از این کتابخانه‌ها، کتابخانه‌یImsl است که به همراه نسخه های مختلف نرم افزار فرترن و سی ارائه شده است. این کتابخانه دارای روندهای مختلف برای حل دستگاه معادله می‌باشد. همچنین نرم افزارهای متلب و متمتیکا نیز دارای کتابخانه‌هایی برای حل دستگاه معادلات خطی می‌باشند. جدول (4) نتایج حل دستگاه معادلات را با استفاده از کتابخانه‌ی این

نرم افزارها را نشان می‌دهد. شایان توجه است، نرم افزار متمتیکا از قابلیت چند هسته‌ای بودن پردازش‌گر نیز استفاده می‌کند که باعث کاهش زمان تحلیل می‌شود.

جدول3-زمان (ثانیه) و مقدار حافظه اشغال شده برای حل دستگاه معادلات در محیط‌های مختلف برنامه نویسی

کامپایلرها و محیط‌های برنامه نویسی								
زمان(ثانیه)	C#	C++	MATLAB	Fortran Compaq	Fortran Powerstation	Mathematica	حافظه مورد استفاده (MB)	تعداد مجهولات
	0.0011	0.000	0.001	0.015	0.0001	0.516	<1	100
	0.568	0.341	0.672	0.765	0.511	13.94	<1	500
	5.387	3.345	6.543	6.812	4.263	74.16	15	1000
	103.42	65.65	123.43	150.43	85.26	818.45	98	2500
	*	*	*	*	*	*	392	5000

* مقدارغیر قابل قبول

با توجه به نتایج بدست آمده و نمایش داده شده در جدول (4)، کتابخانه‌ی نرم افزار های متمتیکا و متلب بهترین عملکرد را از خود نشان داده‌اند. کتابخانه‌ی نسخه‌ی پاورستیشن نیز عملکرد بهتری را نسبت به نسخه‌ی کامپک دارا است. از سوی دیگر، کامپایلر C# در پایین‌ترین رتبه قرار می‌گیرد.

جدول 4- زمان (ثانیه) لازم برای حل دستگاه معادلات با استفاده از کتابخانه‌های مختلف

توابع کتابخانه											تعداد مجهولات
زمان(ثانیه)	Mathematica	MATLAB	C#		Fortran Compaq(Imsl)			Fortran PowerStation (Imsl)			
	-	-	IMSL QR	IMSL LU	DLFIRG LU	LSARG LU	LSLRG LU	DLFIRG LU	LSARG LU	LSLRG LU	
	0	0.0022	0.05	0.031	0.015	0.015	0.0034	0.027	0.025	0.0034	100
	0.035	0.092	0.9	0.45	0.113	0.111	0.052	0.141	0.141	0.078	500
	0.203	0.473	6.67	3.392	0.641	0.641	0.453	0.875	0.891	0.688	1000
	1.652	3.543	NA	51.62	10.254	10.313	8.732	13.42	13.294	12.093	2500
	15.323	26.421	NA	NA	94.532	90.354	80.961	116.453	109.785	104.578	5000

3. حل دستگاه معادلات تنک

روش اجزاء محدود با تبدیل معادلات دیفرانسیل حاکم بر رفتار پدیده های فیزیکی،آن‌ها را به صورت دستگاه معادلات خطی تبدیل می‌کند. این دستگاه معادلات دارای تعداد زیادی درایه صفر می‌باشد. از این رو ذخیره و حل این معادلات با استفاده از روش‌های مقدماتی مقرون به صرفه نمی‌باشد. ماتریسی که دارای درایه های صفر فراوان باشد، ماتریس تنک نام دارد[5]. بنابراین شیوه‌ی ذخیره سازی ماتریس‌های تنک از اهمیت فراوانی برخوردار است. در بخش کنونی، روش‌های مختلف ذخیره سازی ماتریس ضرایب دستگاه معادلات خطی مورد بررسی قرار می‌گردد.

3-1 ذخیره سازی به روش نواری کلی

همان‌گونه که پیداست، این روش برای دستگاه ماتریس‌هایی مورد استفاده قرار می‌گیرد که حالت نواری داشته باشند. ماتریس $A=(a_{ij})$ با اندازه‌ی $n\times n$ را یک ماتریس نواری می‌نامند اگر تمامی درایه‌های خارج از نوار قطری صفر باشند. پهنا و اندازه‌ی نوار قطری، با دو ضریب ثابت k_1 و k_2 مشخص می‌شود. این تعریف به صورت زیر رابطه‌سازی می‌شود:

$$a_{i,j}=0 \quad if \quad j<i-k_1 \quad or \quad j>i+k_2; \quad k_1,k_2\geq 0 \qquad (2)$$

عامل‌های k_1 و k_2 به ترتیب، بیانگر پهنای نیم نوار سمت چپ و راست ماتریس می‌باشد. پهنای نوار ماتریس نیز با معادله‌ی زیریافت می‌شود:

$$bandwidth = k_1 + k_2 + 1 \tag{3}$$

ماتریس نواری کلی، با ذخیره کردن درایه‌های روی نوار قطری و صفر قرار دادن سایر درایه‌ها ذخیره می‌شود. برای مثال، ماتریس A با ابعاد 6×6 و پهنای نوار 3، به صورت ماتریس A' با ابعاد 6×3 ذخیره می‌شود. عامل‌های k_1 و k_2 در ماتریس مثال کنونی برابر 1 می‌باشند.

$$A = \begin{bmatrix} a_{11} & a_{12} & 0 & 0 & 0 & 0 \\ a_{21} & a_{22} & a_{23} & 0 & 0 & 0 \\ 0 & a_{32} & a_{33} & a_{34} & 0 & 0 \\ 0 & 0 & a_{43} & a_{44} & a_{45} & 0 \\ 0 & 0 & 0 & a_{54} & a_{55} & a_{56} \\ 0 & 0 & 0 & 0 & a_{65} & a_{66} \end{bmatrix} \tag{4}$$

$$A' = \begin{bmatrix} 0 & a_{11} & a_{12} \\ a_{21} & a_{22} & a_{23} \\ a_{32} & a_{33} & a_{34} \\ a_{43} & a_{44} & a_{45} \\ a_{54} & a_{55} & a_{56} \\ a_{65} & a_{66} & 0 \end{bmatrix} \tag{5}$$

3-1-1 ذخیره سازی با روش نواری متقارن (BandSPD)

راه‌کار پیش‌رو برای دستگاه‌های ماتریس معین مثبت مورد استفاده قرار می‌گیرد که دارای ویژگی نواری متقارن باشند. ماتریس $A = (a_{ij})$ با اندازه‌ی $n \times n$ را یک ماتریس متقارن نواری می‌نامند اگر رابطه‌های زیر برقرار باشند. اندازه‌ی نیم نوار چپ یا راست ماتریس با عامل k مشخص می‌شود.

$$\begin{aligned} &a_{i,j} = 0 \quad if \quad j < i - k \quad or \quad j > i + k; \quad k \geq 0 \\ &a_{i,j} = a_{ji} \quad , \quad y^T A\, y = 0; \quad y \neq 0, \quad y \in R^n \end{aligned} \tag{6}$$

نحوه‌ی ذخیره‌سازی در این روش با مثال زیر آشکار می‌شود. ماتریس A به صورت ماتریس A' با ابعاد 6×3 ذخیره می‌گردد.

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 & 0 & 0 \\ & a_{22} & a_{23} & a_{24} & 0 & 0 \\ & & a_{33} & a_{34} & a_{35} & 0 \\ & & & a_{44} & a_{45} & a_{46} \\ & sym. & & & a_{55} & a_{56} \\ & & & & & a_{66} \end{bmatrix} \tag{7}$$

$$A' = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{22} & a_{23} & a_{24} \\ a_{33} & a_{34} & a_{35} \\ a_{44} & a_{45} & a_{46} \\ a_{55} & a_{56} & 0 \\ a_{66} & 0 & 0 \end{bmatrix} \tag{8}$$

3-1-2 ذخیره سازی با روش دورنما (ProfileSPD)

این فن برای دستگاه ماتریس‌های متقارن معین مثبت کاربرد دارد. بر این پایه، رابطه‌های زیر برای ماتریس $A = (a_{ij})$ برقرار است.

$$a_{i,j} = a_{ji} \quad , \quad y^T A\, y = 0; \quad y \neq 0, \quad y \in R^n \tag{9}$$

بر پایه‌ی مثال زیر، در هر ستون ماتریس اصلی، درایه‌های نخستین سطر ناصفر آن تا روی قطر اصلی، در یک ماتریس یک بعدی ذخیره می‌شود. بنابراین، ماتریس A با ابعاد 6×6، به صورت دو ماتریس تک بعدی A' و A'' ذخیره می‌گردد. باید دانست، این فن با نام راه‌کار دورنما (Skyline) نیز شناخته می‌گردد.

$$A = \begin{bmatrix} a_{11} & a_{12} & 0 & 0 & 0 & 0 \\ & a_{22} & a_{23} & 0 & a_{25} & 0 \\ & & a_{33} & 0 & 0 & 0 \\ & & & a_{44} & a_{45} & a_{46} \\ & sym. & & & a_{55} & 0 \\ & & & & & a_{66} \end{bmatrix} \tag{10}$$

$$A' = [a_{11} \;\; a_{12} \;\; a_{22} \;\; a_{23} \;\; a_{33} \;\; a_{44} \;\; a_{25} \;\; 0 \;\; a_{45} \;\; a_{55} \;\; a_{46} \;\; 0 \;\; a_{66}] \tag{11}$$

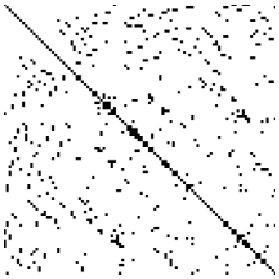
$$A'' = [1 \;\; 3 \;\; 5 \;\; 6 \;\; 10 \;\; 13] \tag{12}$$

ماتریس A''، شمار تجمعی درایه‌های ذخیره شده تا روی قطر اصلی در هر ستون را نشان می‌دهد.

3-1-3 ذخیره سازی با روش تنک متقارن (SparseSPD)

حل بسیاری از مسائل اجزاء محدود و تفاوت‌های محدود به معادلات خطی با ماتریس‌های ضرایب تنک منتهی می‌شود. یادآوری می‌کند، با ذخیره سازی درایه‌های غیر صفر ماتریس ضرایب، می‌توان به کاهش قابل توجهی در میزان حافظه مورد نیاز برای تحلیل رسید. به این منظور می‌توان از فن‌هایی که برای دستگاه معادله‌های تنک [6] و تنک متقارن [7] مورد استفاده قرار می‌گیرند، نام برد. از این میان می‌توان به راه‌کارهای DOK، LLI، COO و Yale اشاره کرد. به‌منظور پرهیز از طولانی شدن مطلب، از شرح الگوریتم‌های مربوطه خودداری می‌گردد. برای آگاهی از فرآیند

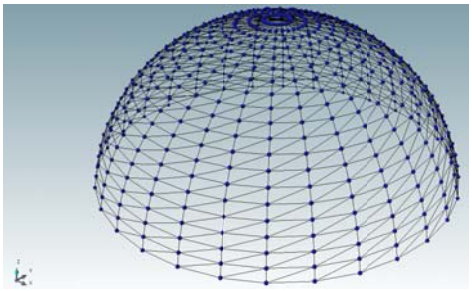
روش‌ها می‌توان به منابع مربوطه رجوع نمود. شمای یک ماتریس تنک متقارن در شکل(1) نشان داده شده است.در این شکل، نقاط سیاه نشانگر درایه‌های ناصفر ماتریس می‌باشند. قطر اصلی ماتریس نیز با خط مشخص شده است.



شکل 1- شمای ماتریس تنک مربوط به حل یک مسئله‌ی جزء محدود

3-2 نمونه های عددی

ذخیره سازی به روش‌ها فوق و استفاده از روش‌های مقدماتی در حل دستگاه معادلات تأثیر چشمگیری در سرعت حل و فضای مورد نیاز برای حل می‌گذارد. برای روشن شدن این مطلب، چند سازه‌ی فضایی گنبدی شکل، همانند آنچه در شکل (2) آمده است، با تعداد درجات آزادی متفاوت در بازه‌ی 18000 درجه آزادی تا 450000 درجه آزادی، تحت اثر بارهای استاتیکی، تحلیل گردید. باید دانست، این نمونه‌های عددی با تغییر تعداد اجزاء گنبد در راستای قائم و شعاعی، به‌دست آمده‌اند. خاطر نشان می‌سازد، مسئله‌ی گنبد کروی مزبور، توسط نویسندگان پیشنهاد و داده دهی شده است. زمان‌های تحلیل هر کدام و میزان فضای لازم برای تحلیل در جدول‌های (5) و(6) آمده است. شایان توجه است، نحوه‌ی شماره‌گذاری درجات آزادی یک سازه از اهمیت خاصی برخوردار است و در تحلیل سازه نقشی اساسی دارد. همچنین، تفاوت جدول‌های (5) و (6) در شیوه شماره گذاری گره‌های سازه می‌باشد. شیوه کاتهیل- مککی (Cuthill-McKee) بازگشتی یکی از شیوه های شناخته شده در شماره‌گذاری گره‌های سازه می‌باشد [8]. نتایج جدول (6) برپایه شیوه مذکور ارائه شده است. در حالت دیگر شماره گذاری درجه‌های آزادی به صورت ساده انجام شده است.



شکل 2- گنبد خرپایی کروی شکل

همان طور که از نتایج بر می‌آید، روش تنک نسبت به روش‌های دیگر چه به لحاظ سرعت و چه به لحاظ حجم فضای مورد نیاز، عملکرد بسیار مناسب‌تری داشته است. از طرفی با مقایسه دو جدول معلوم خواهد شد که در روش تنک نیازی به شماره گذاری درجه آزادی سازه نبوده و این کار باعث کند شدن روند تحلیل و افزایش فضای مورد نیاز جهت ذخیره سازی می‌شود. مقایسه‌ی این شیوه های ذخیره با روش ساده ذخیره سازی نیز بسیار جالب است. برای مثال برای ذخیره ماتریس با 432006 سطر و ستون نیاز به نزدیک به 1 ترابایت فضا می‌باشد در صورتی که در حالت ذخیره سازی به روش تنک تنها کمی بیشتر از 1 گیگابایت فضا مورد نیاز است.

جدول 5- زمان (میلی‌ثانیه) و فضای لازم (مگابایت) برای حل دستگاه معادلات تنک با شماره گذاری ساده

	تعداد درجه‌های آزادی												
	432006		216009		97203		64806		38076		19443		روش
میلی ثانیه	Fail	>2500	Fail	>2500	Fail	>2500	19047	1206	4516	462	2313	232	BandGeneral
	Fail	>2500	Fail	>2500	Fail	1010	17826	491	4000	220	1985	131	BandSPD
	Fail	>2500	Fail	>2500	Fail	594	4312	301	1156	160	547	74	ProfileSPD
	29390	1388	14531	699	3875	280	2016	190	1093	136	469	71	SparseSPD

جدول6- زمان (میلی‌ثانیه) و فضای لازم (مگابایت) برای حل دستگاه معادلات تنک باشماره گذاری به روش بازگشتی کاتهیل- مککی

	تعداد درجه‌های آزادی												
	432006		216009		97203		64806		38076		19443		روش
میلی ثانیه	Fail	>2500	Fail	>2500	Fail	>2500	10891	681	2766	276	1391	153	BandGeneral
	Fail	>2500	Fail	>2500	Fail	600	5265	320	1421	145	734	85	BandSPD
	Fail	>2500	Fail	>2500	14797	605	4250	303	1157	149	593	83	ProfileSPD
	37172	1465	15406	734	4938	326	2281	190	1016	131	468	67	SparseSPD

4. نتیجه گیری

در بخش نخست مقاله‌ی حاضر، سرعت عملکرد روش‌های حل حذفی گوس، تجزیه‌ی گوس، تجزیه‌ی چولسکی و روش‌های تکراری ژاکوبی و گوس- سایدل مورد سنجش قرار گرفت. به این منظور، کد این روش‌ها در نرم افزار متمتیکا نوشته و برای پنج دستگاه معادله خطی بررسی شد. نتایج به دست آمده برتری روش تجزیه‌ی گوس را نشان می‌دهد. در گام بعد، سرعت تحلیل محیط‌های برنامه نویسی و کامپایلرهای فرترن-کامپک، فرترن-پاورستیشن، متمتیکا، متلب ،C++ و C# مورد مقایسه قرار گرفت. بر این اساس، الگوریتم روش تجزیه‌ی گوس در تمامی محیط‌ها برنامه نویسی و برای پنج دستگاه معادله مورد تحلیل قرار گرفت. در این میان، C++ عملکرد بهتری داشته و فرترن-پاورستیشن نیز در جایگاه بعدی قرار می‌گیرد. محیط متمتیکا نیز دارای ضعیف‌ترین نتیجه می‌باشد. سرعت حل کتابخانه‌ی نرم‌افزارهایمتمتیکا و متلب نسبت به سایر موارد سرعت بیشتری دارد. نتایج، نشان‌دهنده‌ی برتری نسخه‌ی پاورستیشن به نسخه‌ی فرترن-کامپک است.همچنین، روش تنک متقارن نسبت به روش‌های دیگر، عملکرد بسیار مناسب‌تری را از نظر سرعت و حجم فضای مورد نیاز برای تحلیل دار است. در راه‌کار تنک، نیازی به بهینه‌سازی شماره‌گذاری درجه آزادی سازه نبوده و این کار باعث کند شدن روند تحلیل و افزایش فضای مورد نیاز جهت ذخیره‌سازی می‌شود.

5. مراجع

1. Calinger, R. (1999), "*A Contextual History of Mathematics*", Prentice Hall

2. Golub, G. H. and Van Loan, C. F. (1996), "*Matrix Computations*", Johns Hopkins

3. Poole, D. (2006), "*Linear Algebra: A Modern Introduction*", Thomson Brooks/Cole

4. Okunev, P. and Johnson, C. (1997), "*Necessary and Sufficient Conditions for Existence of the LU Factorization of an Arbitrary Matrix*", arXiv:Math, Cornell University Library

5. Stoer, J. and Bulirsch, R. (2002), "*Introduction to Numerical Analysis*", Springer

6. Demmel, J., Eisenstat, S. C., Gilbert, J. R., Li, X. S. and Liu, J. W. H. (1999), "*A Super-nodal Approach to Sparse Partial Pivoting*", SIAM Journal of Matrix Analysis and Applications, 20(3)

7. Law, K. H. and McKay, D. R. (1993), "*A Parallel Row-Oriented Sparse Solution Method for Finite Element Structural Analysis*", International Journal for Numerical Methods in Engineering, 36(17)

8. Cuthill, E. and McKee, J. (1969), "*Reducing the Bandwidth of Sparse Symmetric Matrices*", Proceedings of the 24th National Conference, ACM