



دانشگاه صنعتی امیرکبیر  
(پلی تکنیک تهران)  
دانشکده مهندسی مکانیک

## پروژه اول درس بررسی المان محدود غشاء الاستیک

نگارش  
رضا نوپور هولاری (۴۰۲۱۲۶۹۲۴)

استاد درس  
جناب آقای دکتر محمد رضا اسلامی



## چکیده

در پروژه اول درس المان محدود، به بررسی خیز غشاء الاستیک به روش المان محدود پرداخته می‌شود. در ابتدا، کلیات غشاء و معادلات حاکم بر مسئله و همچنین شرح توضیحاتی از روش حل و الگوریتم داده شده است. سپس، در انتهای نتایج حاصل در غالب نمودار و جدول شرح داده شده است. از دستاوردهای این پروژه می‌توان به این اشاره کرد که استفاده از المان با تعداد گره‌های بیشتر، می‌توان به کاهش تعداد نقاط المان برای همگرایی در جواب نهایی (خیز در مسئله حاضر) اشاره کرد.

## صفحه

## فهرست مطالعه

۶	فصل اول مقدمه و مرور ادبیات.....
۷	۱- اهمیت مطالعه خیز غشاء.....
۸	۲- مروری بر پژوهش‌های خیز غشاء.....
۹	۳- المان محدود.....
۱۱	۴- جمع‌بندی.....
۱۲	فصل دوم مسئله المان محدود غشاء.....
۱۳	۱- روش المان محدود.....
۱۴	۲- غشاء الاستیک.....
۱۴	۱-۲-۲- مسئله مقدار مرزی.....
۱۴	۲-۲-۲- مسئله اکستررم.....
۱۸	۳-۲- روش حل.....
۱۸	۱-۳-۲- اسمبلی ماتریس گلوبال.....
۲۰	۲-۳-۲- محاسبه پهنهای باند.....
۲۱	۳-۳-۲- روش حذفی گاوس.....
۲۴	۴-۳-۲- مسائل استاتیکی، روش Skyline.....
۲۶	فصل سوم برنامه کامپیووتری.....
۲۹	۱-۳- پیش‌پردازش.....
۳۱	۲-۳- پردازش.....
۳۲	۳-۳- پس‌پردازش.....
۳۳	فصل چهارم نتایج.....
۳۴	۱- ابزار صحّت‌سنگی.....
۳۷	۲-۴- مطالعه همگرایی کد کامپیووتری.....
۴۷	۳-۴- مطالعه پارامتری.....
۵۳	فصل پنجم جمع‌بندی و نتیجه‌گیری.....
۵۵	منابع و مراجع.....
۵۶	منابع.....
۵۷	پیوست‌ها.....

## صفحه

## فهرست اشکال

شکل ۲ - ۱ علامت‌گذاری برای پیوستار تقریبی *	$R$	۱۳
شکل ۲ - ۲ علامت‌گذاری برای یک المان پایه (e)	Dev C++	۱۵
شکل ۲ - ۳ روش skyline banded یک ماتریس		۲۵
شکل ۳ - ۱ نمای کلی کد ابتدایی		۲۷
شکل ۳ - ۲ نمای کلی کد اصلاح شده در Dev C++		۲۸
شکل ۳ - ۳ فلوچارت اجرای کد المان محدود C++ برای غشاء الاستیک		۲۹
شکل ۳ - ۴ شکل تغییر شکل غشاء برای مش مثلثی سه گره یک دامنه مستطیلی [۱۰]		۳۲
شکل ۴ - ۱ خیز غشاء مستطیلی، محاسبه شده از حل معادله پواسون توسط ابزار حل معادلات جزئی با مش		۰۱
شکل ۴ - ۲ نمایش مش ۰۰۱ در محاسبه خیز غشاء مستطیلی، حل معادله پواسون توسط ابزار حل معادلات		۳۵
شکل ۴ - ۳ خیز غشاء مستطیلی، محاسبه شده از حل معادله پواسون توسط ابزار حل معادلات جزئی با مش		۳۵
شکل ۴ - ۴ خیز غشاء مستطیلی دو بعدی در محاسبه خیز غشاء مستطیلی، حل معادله پواسون توسط ابزار حل معادلات جزئی		۰۱۸
شکل ۴ - ۵ نمایش سه بعدی همگرایی المان مثلثی با ۳ گره تعداد المان‌های طولی و عرضی برای یک غشاء		۳۶
شکل ۴ - ۶ نمایش دو بعدی همگرایی المان مثلثی با ۳ گره تعداد المان‌های عرضی برای یک غشاء $8 \times 5$ مستطیلی		۳۷
شکل ۴ - ۷ نمایش دو بعدی همگرایی المان مثلثی با ۳ گره تعداد المان‌های طولی برای یک غشاء $8 \times 5$ مستطیلی		۳۸
شکل ۴ - ۸ نمای جزئی نمایش سه بعدی همگرایی المان مثلثی با ۳ گره تعداد المان‌های طولی برای یک غشاء $8 \times 5$ مستطیلی		۳۹
شکل ۴ - ۹ نمای خیز غشاء مستطیلی با المان مش مثلثی سه گره برای $nx = 5, ny = 8$		۴۰
شکل ۴ - ۱۰ نمای خیز غشاء مستطیلی با المان مش مثلثی سه گره برای $nx = 10, ny = 16$		۴۰
شکل ۴ - ۱۱ نمای خیز غشاء مستطیلی با المان مش مثلثی سه گره برای $nx = 20, ny = 32$		۴۱
شکل ۴ - ۱۲ نمای خیز غشاء مستطیلی با المان مش مثلثی سه گره برای $nx = 30, ny = 48$		۴۱
شکل ۴ - ۱۳ نمای خیز غشاء مستطیلی با المان مش مثلثی سه گره برای $nx = 40, ny = 68$		۴۲

..... شکل ۴ - ۱۴ نمایش سه بعدی همگرایی المان مثلثی با ۶ گره تعداد المان‌های طولی و عرضی برای یک غشاء $8 \times 5$ مستطیلی.	۴۳
..... شکل ۴ - ۱۵ نمایش دو بعدی همگرایی المان مثلثی با ۶ گره تعداد المان‌های عرضی برای یک غشاء $8 \times 5$ مستطیلی.	۴۳
..... شکل ۴ - ۱۶ نمایش دو بعدی همگرایی المان مثلثی با ۶ گره تعداد المان‌های طولی برای یک غشاء $8 \times 5$ مستطیلی.	۴۴
..... شکل ۴ - ۱۷ نمای خیز غشاء مستطیلی با المان مش مثلثی شش گره برای $nx = 5, ny = 8$	۴۵
..... شکل ۴ - ۱۸ نمای خیز غشاء مستطیلی با المان مش مثلثی شش گره برای $nx = 10, ny = 16$	۴۵
..... شکل ۴ - ۱۹ نمای خیز غشاء مستطیلی با المان مش مثلثی شش گره برای $nx = 15, ny = 24$	۴۶
..... شکل ۴ - ۲۰ نمای خیز غشاء مستطیلی با المان مش مثلثی شش گره برای $nx = 20, ny = 32$	۴۶
..... شکل ۴ - ۲۱ تغییرات خیز غشاء مستطیلی در حالت اولیه مقطع $8 \times 5$ مستطیلی.	۴۷
..... شکل ۴ - ۲۲ تغییرات خیز غشاء مستطیلی با تغییرات فشار وارد بر غشاء.	۴۸
..... شکل ۴ - ۲۳ شماتیک تغییرات خیز غشاء مستطیلی با تغییرات طولی و عرضی مقطع مستطیلی.	۴۹
..... شکل ۴ - ۲۴ شماتیک کانتور تغییرات خیز غشاء مستطیلی با تغییرات طولی و عرضی مقطع مستطیلی.	۵۰
..... شکل ۴ - ۲۵ شماتیک غشاء دایره‌ای با شعاع ۱.	۵۱
..... شکل ۴ - ۲۶ نمای خیز غشاء دایره‌ای با المان مش مثلثی سه گره برای لایه اول خارجی با تعداد المان ۶۴.	۵۱
..... شکل ۴ - ۲۷ نمای خیز غشاء دایره‌ای با المان مش مثلثی سه گره برای لایه اول خارجی با تعداد المان ۱۲۸.	۵۲
..... شکل ۴ - ۲۸ نمای خیز غشاء دایره‌ای با المان مش مثلثی سه گره برای لایه اول خارجی با تعداد المان ۲۵۶.	۵۲

صفحه

فهرست جداول

جدول ۲ - ۱ الگوریتم روش حذف گاوی	۲۳
جدول ۴ - ۱ تغییرات خیز با تغییرات طولی و عرضی مقطع مستطیلی	۴۹
جدول پ - ۱ کد C++ مسئله المان محدود شعاع دایره‌ای	۵۸

## فصل اول

### مقدمه و مرور ادبیات

## مقدمه

در این بخش ابتدا خلاصه‌ای از اهمیت خیز غشاء و روش المان محدود در پژوهش‌ها بیان خواهد شد.

### ۱-۱- اهمیت مطالعه خیز غشاء

غشاء<sup>۱</sup>، در زمینه مهندسی سازه، به ماده نازک و انعطاف پذیری اطلاق می‌شود که می‌تواند بارها را عمدتاً از طریق عملکرد غشا تحمل کند، جایی که تنش‌ها عمدتاً کششی هستند. غشاء و فرآیندهای غشایی معمولاً در سازه‌هایی مانند چادرها، سقف‌های پارچه‌ای و سازه‌های بادی استفاده می‌شوند.

تجزیه و تحلیل خیز غشا می‌تواند به دلایل زیر حائز اهمیت باشد.

۱. **یکپارچگی سازه<sup>۲</sup>:** درک خیز غشاء به مهندسان کمک می‌کند تا اطمینان حاصل کنند که سازه می‌تواند بارهای اعمال شده را بدون تغییر شکل یا شکست بیش از حد تحمل کند.

۲. **پیش بینی عملکرد<sup>۳</sup>:** با تجزیه و تحلیل خیز، مهندسان می‌توانند رفتار ساختار غشایی در شرایط بارگذاری مختلف را پیش بینی کنند و آن را بهینه نمایند.

۳. **ایمنی<sup>۴</sup>:** بررسی خیز به ارزیابی ایمنی سازه کمک می‌کند، اطمینان حاصل می‌کند که در محدوده قابل قبول باقی می‌ماند و خطری برای کاربران یا محیط اطراف ایجاد نماید.

۴. **زیبایی<sup>۵</sup>:** تجزیه و تحلیل خیز برای حفظ ظاهر مطلوب ساختارهای غشایی ضروری است و اطمینان حاصل می‌کند که شکل و زیبایی خود را با گذشت زمان حفظ می‌کنند.

به طور کلی، می‌توان گفت تجزیه و تحلیل خیز یک غشاء از اهمیت ویژه‌ای برخوردار است.

<sup>1</sup> Membrane

<sup>2</sup> Structural Integrity

<sup>3</sup> Performance Prediction

<sup>4</sup> Safety

<sup>5</sup> Aesthetics

## ۱-۲- مروری بر پژوهش‌های خیز غشاء

از نمونه پژوهش‌های انجام شده بر روی خیز غشاء می‌توان به پژوهش شاو و پرون در سال ۱۹۵۴ اشاره کرد [۱]. در این مطالعه با فرمول‌بندی مسئله خیز غیرخطی غشاء تخت بر حسب مولفه‌های جابجایی و استفاده از تقریب‌های تفاضل محدود با روش ریلکسیشن-ایتریشن<sup>۱</sup>، یک رویکرد کارآمد برای حل معادلات دیفرانسیل جزئی مرتبه دوم غیرخطی معرفی شد که امکان تجزیه و تحلیل مسائل غشایی با پیچیدگی‌ها و بار سیستم‌های مختلف را فراهم می‌کند. کاو و پرون در ۱۹۷۱ مسئله خیزهای بزرگ غشاء‌های دلخواه مسطح را مورد مطالعه قرار دادند، موارد شامل غشاهای دایره‌ای، مستطیلی و چند ضلعی بودند [۲]، به صورت ویژه غشاء‌های دایروی متقارن محوری در [۳] بررسی شد. در پژوهش دیگری در ۱۹۷۷، انحرافات بزرگ غشاهای مستطیلی تحت فشار یکنواخت مورد پژوهش قرار گرفت [۴]. مقاله [۵] یک راه حل تحلیلی جدید برای خیز بار غشاهای معرفی شد، که روابط قابل مقایسه بین بار و انحراف را با راه حل‌های تحلیلی موجود نشان داد، اما با ثابت‌های بالاتر که نزدیک به نتایج آنالیز FEM بودند. راه حل جدید مطابقت قابل قبولی با اندازه گیری‌های تجربی با استفاده از غشاهای نیترید سیلیکون نشان می‌دهد. در پژوهشی در ۲۰۰۵، یک رویکرد جدید برای تجزیه و تحلیل غشاهای فضایی الاستیک معرفی شد، که معادلات دیفرانسیل حاکم را در مختصات دکارتی با حداقل سطح غشاء به عنوان مرجع فرمول‌بندی می‌کند. با استفاده از روش معادله آنالوگ (Analogue Equation Method) برای ادغام مستقیم، این روش معادلات دیفرانسیل جزئی غیرخطی را با معادلات غشای تخت خطی جایگزین می‌کند و عناصر ناشناخته را از طریق روش المان مرزی (BEM) تعیین می‌کند تا جابه‌جایی‌ها و ثربخشی نتایج تنش را در غشاهای در مقایسه با روش‌های تحلیلی و عددی به طور کارآمد و دقیق نشان دهد [۶]. فیتچر در پژوهشی تحت حمایت ناسا در بررسی مجدد راه حل کلاسیک هنکی را برای انحرافات بزرگ یک غشای همسانگرد دایره‌ای تحت فشار یکنواخت مورد بحث قرار می‌دهد و یک خطای جبری و اهمیت گنجاندن جزء شعاعی فشار اعمال شده را بر جسته می‌کند. نتایج نشان می‌دهد که راه حل تصحیح شده و شامل مؤلفه شعاعی برای غشاهای با بار سبک بسیار دقیق هستند، اما با تشدید بار، با تفاوت‌های قابل توجهی در حداقل تنش‌ها و انحراف‌ها در شرایط تنش بالا، واگرا می‌شوند. این مطالعه نشان می‌دهد که نمایش دقیق‌تر شکل انحراف غشا تحت

<sup>۱</sup> Relaxation-iteration procedure

بارگذاری فشار واقعی، اثربخشی طراحی بازتابندها را صرفاً بر اساس نظریه هنکی به چالش می‌کشد، و توزیع‌های ضخامت غشایی غیریکنواخت را آشکار می‌کند که بر انبساط شعاعی مرزهای دایره‌ای بر اساس پارامترهای مواد و بارگذاری تأثیر می‌گذارد [۷]. در سال‌های اخیر استفاده از مواد جدید و با خواص گوناگون افزایش یافته است، لیم [۸]، خیز غشاهای دایره‌ای ساخته شده از مواد آگزتیک با نسبت پواسون منفی<sup>۱</sup> در مقایسه با مواد معمولی را مورد بررسی قرار داد. این مدل‌های نیمه تجربی را برای ساده‌سازی محاسبات و بهبود دقت در تخمین انحراف معرفی می‌کند، و نشان می‌دهد خیز غشاء دایره‌ای با منفی‌تر شدن نسبت پواسون افزایش می‌یابد. مطالعه [۹] تغییر شکل غشاهای ارتوتروپیک<sup>۲</sup> دایره‌ای تحت یک نیروی متمرکز مرکزی را با استفاده از تئوری غشاء Föppl-von Kármán و حل‌های عددی در MATLAB بررسی می‌کند. این تحقیق یک پارامتر کلیدی،  $k$  را شناسایی می‌کند که بر رفتار سازه تأثیر می‌گذارد، نشان می‌دهد که تغییرات در مدول الاستیک و نسبت پواسون عمدتاً بر انحراف در غشاهای ارتوتروپ اثر گذارند در حالی که دقت معادله غشای ارتوتروپ پیشنهادی را از طریق مقایسه با شبیه‌سازی‌های ABAQUS نشان می‌دهد.

### ۳-۱- المان محدود

در سالنامه‌های مکانیک، روش اجزای محدود (FEM) تاریخچه‌ای دارد که به اوایل قرن بیستم بازمی‌گردد. بذر FEM توسط رویاپردازانی مانند والتر ریتز و بوریس گالرکین کاشته شد، که مشارکت‌های پیشگامانه آنها زمینه را برای توسعه این تکنیک عددی قدرتمند فراهم کرد. والتر ریتز، اعجوبه ریاضی زمان خود، روش variational را در دهه ۱۹۲۰ معرفی کرد که بعداً به عنوان پیشرو برای FEM عمل کرد. رویکرد ریتز شامل تقریب حل معادلات دیفرانسیل با استفاده از یک سری توابع آزمایشی<sup>۳</sup> بود. این روش که به روش ریتز معروف است، گام مهمی در جهت تحلیل عددی مسائل ساختاری پیچیده است.

<sup>1</sup> Negative Poisson's ratio

<sup>2</sup> orthotropic

<sup>3</sup> trial functions

با تکیه بر کار ریتز، بوریس گالرکین در اواسط قرن بیستم الگوی حل استفاده شده را اصلاح کرد و چیزی را به وجود آورد که امروزه به عنوان روش گالرکین شناخته می‌شود. نوآوری گالرکین شامل ضرب معادله دیفرانسیل در یکتابع وزن و ادغام در دامنه است که منجر به سیستمی از معادلات جبری می‌شود که می‌تواند به صورت محاسباتی حل شود. روش گالرکین چارچوب سیستماتیک و دقیق‌تری برای حل معادلات دیفرانسیل از طریق تقریب عددی ارائه کرد. همگرایی ایده‌های ریتز و گالرکین زمینه را برای FEM فورمولاسیون و حل روش اجزای محدود در دهه ۱۹۵۰ فراهم کرد. مهندسان و ریاضیدانان پتانسیل FEM را به عنوان یک ابزار همه کاره و کارآمد برای حل طیف وسیعی از مسائل مهندسی، به ویژه در تجزیه و تحلیل سازه، تشخیص دادند.

کاربردهای اولیه FEM بر حل مسائل مربوط به مکانیک سازه، مانند تحلیل تیرها، خرپاها و قاب‌ها متمرکز بود. توانایی این روش در مدیریت هندسه‌های پیچیده و شرایط مرزی، آن را به ابزاری ارزشمند برای پیش‌بینی رفتار سازه‌ها تحت شرایط بارگذاری مختلف تبدیل کرده است. با ظهور رایانه‌های دیجیتال در نیمه دوم قرن بیستم، قابلیت‌های FEM به طور تصاعدی گسترش یافت. به لطف قدرت محاسباتی ارائه شده توسط این ماشین‌ها، مهندسان اکنون می‌توانند مسائل بزرگتر و پیچیده‌تر را با دقت و کارایی بیشتر حل کنند.

پذیرش گسترده FEM در صنایعی مانند هواپضا، خودروسازی و مهندسی عمران موقعیت آن را به عنوان یک تغییر دهنده بازی در زمینه مکانیک تثبیت کرد. بسته‌های نرم‌افزار تجاری FEM در دهه‌های ۱۹۷۰ و ۱۹۸۰ پدیدار شدند و این روش را برای مهندسان و طراحان در سراسر جهان در دسترس‌تر کردند. با ادامه پیشرفت منابع محاسباتی، محققان شروع به استفاده از FEM در طیف وسیع تری از رشته‌ها، از جمله دینامیک سیالات، انتقال حرارت، و الکترومغناطیسی کردند. تطبیق پذیری و تطبیق پذیری این روش، آن را به ابزاری برای شبیه‌سازی طیف متنوعی از پدیده‌های فیزیکی در حوزه‌های مختلف علمی و مهندسی تبدیل کرده است.

در عصر مدرن، FEM به ابزاری ضروری برای مهندسان و دانشمندان تبدیل شده است و آنها را قادر می‌سازد تا با مسائل پیچیده در محیط‌های مجازی با دقت و قابلیت اطمینان رفتار کنند. از دستگاه‌های ریزمقیاس در فناوری نانو گرفته تا پروژه‌های زیرساختی در مقیاس بزرگ، FEM نقشی محوری در طراحی، تحلیل و بهینه‌سازی سیستم‌های مکانیکی ایفا می‌کند.

تکامل FEM از ایده های رویایی ریتز و گالرکین به وضعیت فعلی آن به عنوان سنگ بنای مکانیک محاسباتی گواهی بر نوآوری و پشتکار محققان است.

در زمینه مهندسی سازه، زمانی که پیچیدگی معادلات دیفرانسیل حاکم بر مسائل سازه از قلمرو حل های تحلیلی قبل حصول فراتر می رود، روش های عددی و نیمه تحلیلی مختلفی برای مقابله با این چالش پیشنهاد می شود. FEM بسیار تطبیق پذیر است و می تواند برای طیف گسترده ای از مسائل در مهندسی سازه، از جمله مسائل با هندسه های پیچیده و خواص مواد اعمال شود [۱۰-۱۳].

## ۴-۱- جمع‌بندی

با توجه به مطالب بیان شده، می توان نتیجه گرفت مطالعه غشاءها در مهندسی دارای کاربردهای فراوانی می باشد. بنابراین در این پژوهه، بررسی المان محدود یک غشاء مورد مطالعه قرار خواهد گرفت. در فصل دوم، به مدلسازی المان محدود پرداخته می شود، سپس در فصل چهارم، به بررسی کد المان محدود موجود و در فصل چهارم نتایج شرح داده خواهند شد. در انتها نیز جمع‌بندی و نتیجه‌گیری بیان شده است.

## فصل دوم

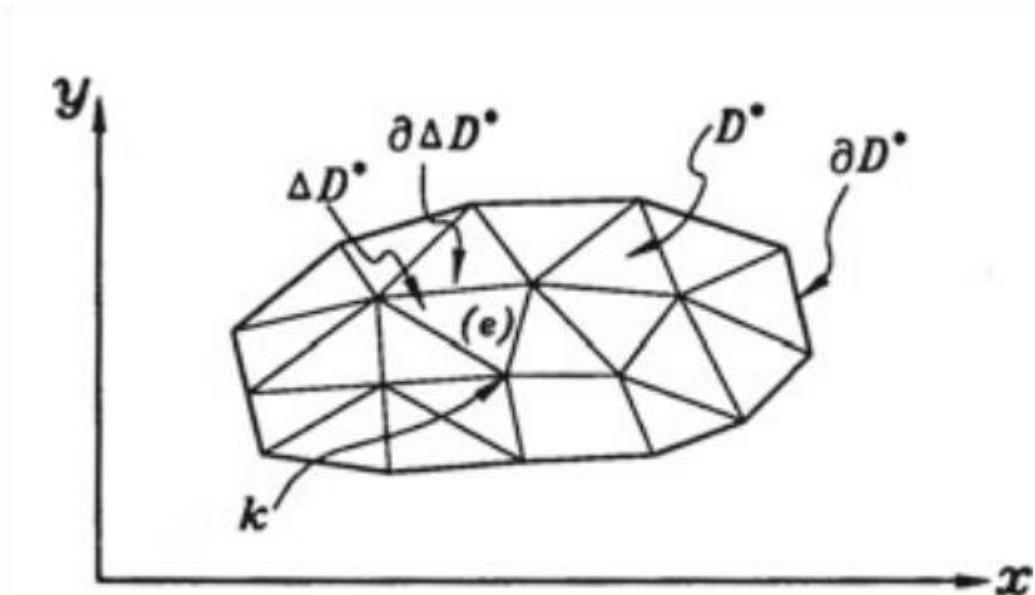
### مسئله المان محدود غشاء

## مقدمه

در این قسمت مسئله المان محدود پروژه مورد مطالعه قرار می‌گیرد [۱۰].

### ۱-۲- روش المان محدود

با تقسیم دامنه یک مشکل معمولی به تعدادی از عناصر دلخواه شروع می‌شود. شکل ۱-۲ مفهوم تقسیم دامنه راه حل را به تعدادی از عناصر مثلثی کوچکتر نشان می‌دهد. هر یک از این تقسیمات فرعی یک المان نامیده می‌شود. در شکل ۱-۲، اطراف المان به طور دلخواه به صورت صاف انتخاب شده‌اند. بعدهاً متذکر می‌شویم که هم هندسه المان و همتابع تقریبی می‌توانند غیرخطی انتخاب شوند. یعنی ممکن است اضلاع المان منحنی باشد، المان ممکن است چهار ضلع با شکل نامنظم انتخاب شود و تابع تقریبی نیز فرم مرتبه بالاتری داشته باشد.



شکل ۲ - ۱ علامت‌گذاری برای پیوستار تقریبی  $R^*$

روش ریتز می‌تواند برای به حداقل رساندن عملکرد انرژی مورد استفاده قرار گیرد. یعنی حل آزمایشی یا تابع درونیابی با ضرایب نامشخص مرتبط با جابجایی نقاط گره فرض می‌شود. سپس، اصل انرژی پتانسیل برای تعیین ضرایب به گونه‌ای اعمال می‌شود که حل آزمایشی نزدیک به حل دقیق در هر عنصر باشد.

٢-٢ - غشاء الاستيك

در این بخش خیز (کوچک) یک غشای بی وزن الاستیک از صفحه مرجع  $(y, x)$  مورد بررسی قرار می‌گیرد. فرمول مسئله مقدار مرزی مرتبط و مسئله اکسترموم معادل در ادامه آورده شده است.

۲-۱-۲-۳- مسئله مقدار مرزی

خیز غشاء  $\phi$ , به صورت معادله پواسون تعریف می‌شود,

$$\nabla^2 \phi = -\frac{p}{T} \quad \text{روی دامنه} \quad (1-2)$$

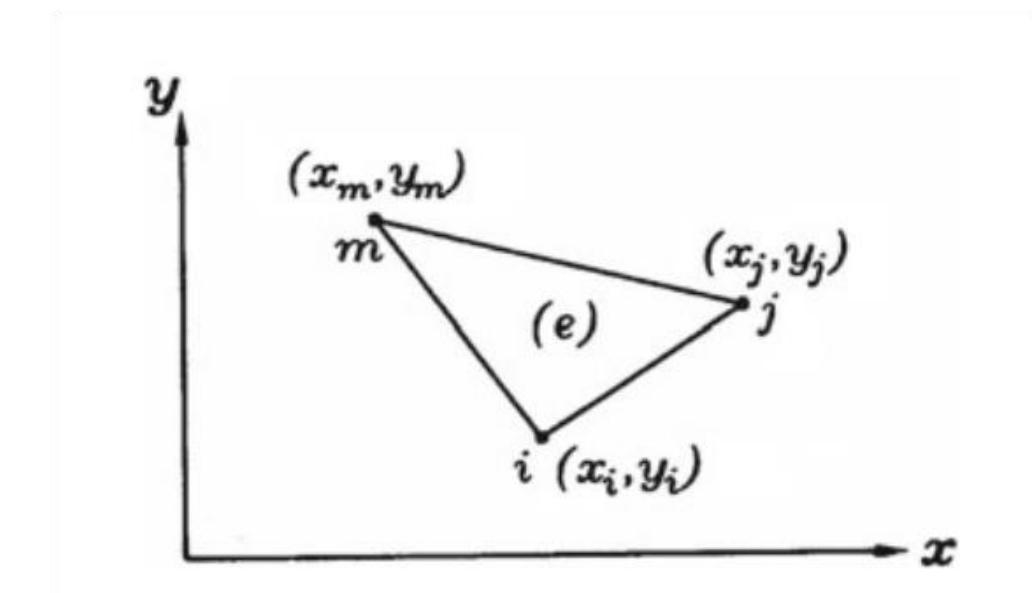
که در آن،  $\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$ ، و  $p$  بیانگر فشار روی سطح غشاء می‌باشد. نیروی کششی نسبت به طول غشاء واحد برابر  $T$  می‌باشد. شرط مرزی به صورت  $\phi = f_1(s)$  روی  $\partial D$  که  $f_1$  بر روی مرز به عنوان تابعی از طول قوس  $s$  تعریف می‌شود.

۲-۲-۲- مسئله اکسترمن

خیز غشاء  $\phi$ , به صورتی تعریف می‌شود که,

$$V = \int_D \left\{ \frac{T}{2} \left[ \left( \frac{\partial^2 \phi}{\partial x^2} \right)^2 + \left( \frac{\partial^2 \phi}{\partial y^2} \right)^2 \right] - \phi p \right\} dx dy \quad (7-7)$$

تابع آزمایشی  $\phi$ , برای  $\phi$ , بر روی هر المان در  $D^*$  نمایش المان محدود فرض می‌شود. شکل ۲-۲ نمایش اجزای محدود دامنه حل مسئله را نشان می‌دهد.



شکل ۲ - ۲ علامت گذاری برای یک المان پایه (e)

احتیاج بر این است که  $\phi^*$ , به اندازه کافی smooth باشد تا مقدار  $V = V[\phi^*]$ , معادله ۲-۲، مینیمم شود. همچنین باید  $\phi^*$  شرایط مرزی نیرویی را ارضاء نماید. همچنین، تابع آزمایشی  $\phi^*$ , بر روی هر المان در  $D^*$  به اندازه دقیق آن  $\phi$  نزدیک باشد. اساساً مسئله اصلی با درجات آزادی نامتناهی به مسئله‌ای با تعداد درجات آزادی محدود کاهش می‌باید. اگر المان‌های موجود در  $D^*$  کوچکتر باشند، یا اگر تابع درون یابی از مرتبه بالاتری برخوردار باشد، پس هدف ما از تطبیق  $\phi^*$  با  $\phi$  باید حداقل در اصل نزدیکتر شود. مشکلات عملی، مانند خطای گرد کردن و زمان محاسبات، کاربر را قادر می‌کند تا در مورد اندازه المان و درجه تقریبی تابع شکل انتخاب شده برای مدل‌سازی مسئله، انتخاب‌های عاقلانه انجام دهد. با در نظر گرفتن شکل ۲ - ۲، انرژی پتانسیل تقریبی مرتبه برابر است با،

$$V^* = \sum_{e=1}^r (V^*)^e \quad (3-2)$$

که  $r$  نشان‌دهنده‌ی تعداد المان‌های  $D^*$  می‌باشد.

$$(V^*)^e = \int_{\Delta D^e} \left\{ \frac{T}{2} \left[ \left( \frac{\partial^2 \phi^*}{\partial x^2} \right)^2 + \left( \frac{\partial^2 \phi^*}{\partial y^2} \right)^2 \right] - p \phi^* \right\} dx dy \quad (4-2)$$

در فرمول اشاره شده،  $\Delta D^e$  بیانگر زیردامنه مرتبه با المان  $e$  روی  $D^*$  می‌باشد. شرط لازم برای حداقل بودن  $V^*$  این است که،

$$\frac{\partial V^*}{\partial \phi_k^*} = 0 \quad k = 1, 2, \dots, n \quad (5-2)$$

در این عبارت  $n$  تعداد نقاط گره بر روی  $D^*$  می‌باشد.تابع درونیابی خطی  $\phi$  فرض شده برابر است با،

$$\phi = a_1 + a_2 x + a_3 y \quad (6-2)$$

با در نظر گرفتن سه گره،

$$\begin{aligned} \phi &= \phi_i \quad \text{at } x = x_i; \quad y = y_i \\ \phi &= \phi_j \quad \text{at } x = x_j; \quad y = y_j \\ \phi &= \phi_m \quad \text{at } x = x_m; \quad y = y_m \end{aligned} \quad (7-2)$$

در نتیجه،

$$\begin{aligned} a_1 &= \frac{1}{2\Delta} (a_i \phi_i + a_j \phi_j + a_m \phi_m) \\ a_2 &= \frac{1}{2\Delta} (b_i \phi_i + b_j \phi_j + b_m \phi_m) \\ a_3 &= \frac{1}{2\Delta} (c_i \phi_i + c_j \phi_j + c_m \phi_m) \end{aligned} \quad (8-2)$$

دو برابر مساحت المان در نظر گرفته شده،  $e$ ، برابر است با

$$2\Delta = \det \begin{vmatrix} 1 & x_i & y_i \\ 1 & x_j & y_j \\ 1 & x_m & y_m \end{vmatrix}$$

۶

$$\begin{cases} a_i = x_j y_m - x_m y_j \\ b_i = y_j - y_m \\ c_i = x_m - x_j \end{cases} \quad \begin{cases} a_j = x_m y_i - x_i y_m \\ b_j = y_m - y_i \\ c_j = x_i - x_m \end{cases} \quad \begin{cases} a_m = x_i y_j - x_j y_i \\ b_m = y_i - y_j \\ c_m = x_j - x_i \end{cases} \quad (9-2)$$

با مرتب سازی معادلات و جایگذاری در ۶-۲ خواهیم داشت،

$$\begin{aligned} \phi &= \frac{1}{2\Delta} \left\{ (a_i + b_i x + c_i y) \phi_i + (a_j + b_j x + c_j y) \phi_j \right. \\ &\quad \left. + (a_m + b_m x + c_m y) \phi_m \right\} \end{aligned} \quad (10-2)$$

با در نظر گرفتن،

$$N_i = \frac{a_i + b_i x + c_i y}{2\Delta}, N_j = \frac{a_j + b_j x + c_j y}{2\Delta}, N_m = \frac{a_m + b_m x + c_m y}{2\Delta} \quad (11-2)$$

معادله ۱۰-۲ برای یک المان به صورت زیر خواهد شد،

$$\phi^{(e)} = \langle N_i \quad N_j \quad N_m \rangle \begin{Bmatrix} \phi_i \\ \phi_j \\ \phi_m \end{Bmatrix}^{(e)} = \langle N \rangle \{\phi\}^{(e)} \quad (12-2)$$

در این معادله بردار درجه آزادی یا مختصات تعمیم یافته المان  $(e)$  با  $\{\phi\}^{(e)}$  نشان داده شده است. با در نظر گرفتن معادله حاکم،

$$V = \sum_{e=1}^r \int_{D(e)} \left\{ \frac{T}{2} \left[ \left( \frac{\partial \phi}{\partial x} \right)^2 + \left( \frac{\partial \phi}{\partial y} \right)^2 \right] - p\phi \right\} dx dy \quad (13-2)$$

با انجام عملیات‌های ریاضی خواهیم داشت،

$$\frac{\partial V^e}{\partial \phi_i} = \int_{D(e)} \left\{ T \left[ \frac{\partial \phi}{\partial x} \frac{\partial}{\partial \phi_i} \left( \frac{\partial \phi}{\partial x} \right) + \frac{\partial \phi}{\partial y} \frac{\partial}{\partial \phi_i} \left( \frac{\partial \phi}{\partial y} \right) \right] - p \frac{\partial \phi}{\partial \phi_i} \right\} dx dy \quad (14-2)$$

با جایگذاری نتیجه معادله ۱۲-۲،

$$\begin{aligned} \frac{\partial V^e}{\partial \phi_i} &= \int_{D(e)} \left\{ \frac{T}{4\Delta^2} \left[ \langle b_i \quad b_j \quad b_m \rangle \begin{Bmatrix} \phi_i \\ \phi_j \\ \phi_m \end{Bmatrix}^{(e)} b_i \right. \right. \\ &\quad \left. \left. + \langle c_i \quad c_j \quad c_m \rangle \begin{Bmatrix} \phi_i \\ \phi_j \\ \phi_m \end{Bmatrix}^{(e)} c_i \right] - p N_i \right\} dx dy \end{aligned} \quad (15-2)$$

که با دانستن اینکه ضرایب همگنی اعداد ثابت هستند، داریم،

$$\frac{\partial V^e}{\partial \phi_i} = \frac{T}{4\Delta^2} [\langle b_i b_i \quad b_j b_i \quad b_m b_i \rangle + \langle c_i c_i \quad c_j c_i \quad c_m c_i \rangle] \begin{Bmatrix} \phi_i \\ \phi_j \\ \phi_m \end{Bmatrix}^{(e)} \quad (16-2)$$

$$- \int_{D(e)} p N_i dx dy$$

با ثابت در نظر گرفتن فشار روی المان، برای ترم آخر داریم،

$$\begin{aligned} - \int_{D(e)} p N_i dx dy &= - \frac{p}{2\Delta} \int_{D(e)} (a_i + b_i x + c_i y) dx dy \\ &= - \frac{p}{2} (a_i + b_i \bar{x} + c_i \bar{y}) = - \frac{p\Delta}{3} \end{aligned} \quad (17-2)$$

علامت  $(\bar{\ })$ ، بیانگر مرکز المان در نظر گرفته شده،  $(e)$ ، می‌باشد. بنابراین،

$$\frac{\partial V^e}{\partial \phi_i} = \frac{T}{4\Delta^2} [\langle b_i b_i \quad b_j b_i \quad b_m b_i \rangle + \langle c_i c_i \quad c_j c_i \quad c_m c_i \rangle]^{(e)} \begin{Bmatrix} \phi_i \\ \phi_j \\ \phi_m \end{Bmatrix}^{(e)} - \frac{p\Delta}{3} \quad (17-2)$$

که در این مثال،  $T$  و  $p$  در  $D(e)$  ثابت در نظر گرفته شده‌اند. با محاسبه برای هر المان و در نهایت نوشتن روش ریتز، خواهیم داشت،

$$\frac{\partial V}{\partial \phi_i} = \sum_{e=1}^{NE} \frac{\partial V^e}{\partial \phi_i} = 0 \quad (18-2)$$

با در نظر گرفتن تمامی المان‌ها و سپس اسمبلی با توجه به جایگاه به معادله حاکم بر مسئله به فرم زیر در خواهد آمد، که در بخش بعدی به روش‌های حل آن پرداخته می‌شود.

$$[K]\{\phi\} = \{F\} \quad (19-2)$$

### ۳-۲- روشن حل

هنگامی که مشتقات برای به دست آوردن ماتریس‌های اجزای محدود برای یک عنصر پایه ( $e$ ) تکمیل شد، ماتریس‌های عنصر جمع (اسمبل) می‌شوند تا معادله تعادل اجزای محدود را تشکیل دهند. روش‌های کامپیوترا برای حل این معادله نهایی شامل مونتاژ ماتریس، محاسبه پهنه‌ای باند<sup>۱</sup>، اعمال شرایط مرزی، الگوریتم حل و تهیه نتایج خروجی است.

#### ۳-۱- اسمبلی ماتریس گلوبال<sup>۲</sup>

ماتریس‌های عنصر دارای ابعاد محدود هستند و به تعداد کل گره‌ها در هر عنصر و درجات آزادی در هر گره مربوط می‌شوند. این ماتریس‌ها در ماتریس‌های گلوبال اسمبل می‌شوند که ابعاد آنها به تعداد کل گره‌های مسئله و درجات آزادی در هر گره بستگی دارد. بنابراین، ماتریس‌های گلوبال ابعاد بزرگی دارند و

<sup>1</sup> Bandwidth Calculation

<sup>2</sup> Assembly of the Global Matrices

به عنوان مجموع ماتریس‌های عناصر ابعاد بسیار کوچکتر به دست می‌آیند. این بدان معناست که فرآیند مونتاژ ماتریس‌های عنصر فقط یک جمع ساده نیست، بلکه یک فرآیند بسط و جمع<sup>۱</sup> است.

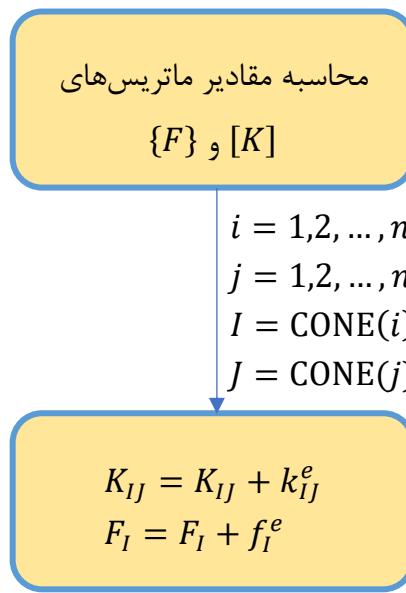
دامنه حلی را در نظر بگیرید که به  $N$  گره گستته شده است. علاوه بر این، تنها یک درجه آزادی در هر گره به نام  $u$  را در نظر بگیرید. یک عنصر ( $e$ ) از حوزه حل ممکن است دارای گره‌ها، مثلاً،  $l, m$  و  $n$  و ماتریس ناشناخته عنصر باشد.

$$\langle u \rangle^{(e)} = \langle u_l \quad u_m \quad u_n \rangle^{(e)} \quad (20-2)$$

ماتریس ناشناخته گلوبالی که شامل مجھولات  $u_l \quad u_m \quad u_n$  عنصر ( $e$ ) است، برابر است با،

$$\langle U \rangle = \langle u_1 \quad u_2 \dots \quad u_l \dots \quad u_m \dots \quad u_n \dots \quad u_{NN} \rangle^{(e)} \quad (21-2)$$

ماتریس ناشناخته گلوبالی شامل  $u_1$  تا  $u_{NN}$  به ترتیب افزایش اعداد است، که در آن  $u_l \quad u_m \quad u_n$  در  $\langle U \rangle$  به ترتیب شماره زیرنویس خود قرار می‌گیرند. الگوریتم در این قسمت برابر است با،



این الگوریتم ساده ماتریس‌های سختی و نیرو را برای همه عناصر در حوزه حل جمع‌آوری می‌کند.

<sup>1</sup> expansion and summation

### ۲-۳-۲- محاسبه پهنای باند<sup>۱</sup>

با توجه به ماهیت تقریب المان محدود، ماتریس‌های جرم و سختی گلوبال حاصل باندبندی شده‌اند. دلیل آن بسیار ساده است، زیرا هر المان فقط به تعدادی المان در اطراف و گرهای خود متصل است. تشخیص پهنای باند ماتریس‌های جرم و سختی و استفاده از یک الگوریتم حل عددی که از پهنای باند ماتریس‌ها استفاده می‌کند بسیار مهم است [۱۴].

این برخلاف روش استاندارد حذف گاووس است که روش حل را بر اساس  $n \times n$  ماتریس می‌سازد. برای تعیین پهنای باند ماتریس سختی، ممکن است عضو  $K_{IJ}$  ماتریس سختی گلوبال در نظر گرفته شود. اعضای قطری ماتریس گلوبال در مکان‌های  $J = I$  قرار دارند. برای مثلث بالایی ماتریس سختی (جایی

که  $J > I$ ، باند یک عنصر  $K_{IJ}$  ماتریس گلوبال  $I$  هستند یا،

$$b_{IJ}^e = J - I = \text{CONE}(j) - \text{CONE}(i) \quad J > I \quad (22-2)$$

حال، وقتی ماتریس‌های سختی المان  $k_{ij}^e$  مونتاژ می‌شوند، اعضای آن‌ها که در فاصله‌ای دور از قطر قرار دارند، نوار ماتریس گلوبال برای آن المان و برای ردیف‌های مورد بررسی هستند. بنابراین، پهنای باند المان برای ردیف  $I$ ، زمانی که ماتریس سختی عنصر  $k_{ij}^e$  مونتاژ می‌شود، حداقل مقدار  $J$  برای تمام جمله‌های غیر صفر آن ردیف است و

$$\begin{aligned} i &= 1, 2, \dots, n_{de} \\ I &= \text{CONE}(i) \\ j &= 1, 2, \dots, n_{de} \\ b_I^e &= \max(\text{CONE}(j) - I) \end{aligned} \quad (23-2)$$

هنگامی که همه المان‌ها مونتاژ می‌شوند، پهنای باند برای ردیف  $I$  ماتریس گلوبالی برای تمامی المان‌ها (e) برابر است با،

$$\begin{aligned} e &= 1, NE \\ b_I &= \max(b_I^e) \end{aligned} \quad (24-2)$$

پهنای باند نهایی ماتریس گلوبال برابر حداقل  $b_I$  تمام سطراها است، و به صورت زیر تعریف می‌شود،

<sup>1</sup> Bandwidth Calculation

$$\begin{aligned} I &= 1, NN \\ b &= \max(b_I) \end{aligned} \quad (25-2)$$

که در آن  $N E$  و  $NN$  به ترتیب تعداد کل المانها و نقاط گرهی در ناحیه حل هستند. با روشی مشابه، ارتفاع باند المان  $h_J^e$ ، ارتفاع باند هر ستون از ماتریس سختی گلوبال  $h_J$ ، و ارتفاع کل باند ماتریس سختی  $h$  به دست می‌آید. ارتفاع باند المان  $h_{IJ}^e$  تمام عبارات غیر صفر ستون  $J$  است که از الگوریتم زیر حاصل می‌شود،

$$\begin{aligned} j &= 1, 2, \dots, n_{de} \\ J &= \text{CONE}(i) \\ i &= 1, 2, \dots, n_{de} \\ h_J^e &= \max(J - \text{CONE}(i)) \end{aligned} \quad (26-2)$$

ارتفاع باند  $h_J$  ستون  $J$  ماتریس سختی گلوبال  $[K]$ ، زمانی که همه المان‌ها مونتاژ می‌شوند، برابر است با

$$h_J = \max(h_J^e) \quad (27-2)$$

و ارتفاع کل باند  $h$  ماتریس سختی گلوبال برابر است با،

$$\begin{aligned} J &= 1, NN \\ h &= \max(h_J) \end{aligned} \quad (28-2)$$

ارتفاع باند هر ستون برای روش حل *skylines* مورد نیاز است، که بعداً در این فصل مورد بررسی قرار می‌گیرد.

### ۲-۳-۳- روش حذفی گاووس<sup>۱</sup>

روش حذف گاووس به طور گسترده برای حل سیستم‌های معادلات به دست آمده از طریق مدل‌سازی اجزای محدود استفاده می‌شود. روش استاندارد حذف گاووس برای حل یک سیستم معادلات با یک ماتریس ضریب ثابت به اندازه  $n \times n$  یا یک ماتریس مربع توسعه داده شده است. این روش تا حدی برای حل به ماتریس‌های banded مستطیلی که به طور طبیعی برای مسائل اجزای محدود به دست می‌آیند نیز اصلاح شده است. حذف گاووس یک روش مستقیم حل سیستم معادلات است. یک جایگزین

<sup>۱</sup> The Gauss elimination method

برای این روش، الگوریتم iteration and relaxation است که گاهی اوقات برای حل یک سیستم معادلات به دست آمده از مدل سازی اجزای محدود استفاده می شود.

برای توصیف این روش، سیستم کلی سه معادله خطی زیر را در نظر بگیرید،

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 &= c_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 &= c_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 &= c_3 \end{aligned} \quad (29-2)$$

در مرحله اول، معادله اول در  $a_{21}/a_{11}$ - ضرب می شود و به معادله دوم اضافه می شود تا جایگزین معادله دوم شود. به همین ترتیب، معادله اول در  $a_{31}/a_{11}$ - ضرب و به معادله سوم اضافه می شود تا جایگزین معادله سوم شود. مجموعه نهایی معادلات تبدیل می شود به،

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 &= c_1 \\ a'_{22}x_2 + a'_{23}x_3 &= c'_2 \\ a'_{32}x_2 + a'_{33}x_3 &= c'_3 \end{aligned} \quad (30-2)$$

که در آن  $a'$  و  $c'$  ضرایب جدید حاصل از ضرب و جمع هستند. نتیجه عملیات مرحله اول حذف متغیر  $x_1$  از معادلات دوم و سوم بود. حال به عنوان مرحله دوم سعی می کنیم  $x_2$  را از معادله سوم حذف کنیم. با ضرب کردن معادله دوم در  $a'_{32}/a_{22}$ - و اضافه کردن آن به معادله سوم خواهیم داشت،

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 &= c_1 \\ a'_{22}x_2 + a'_{23}x_3 &= c'_2 \\ a''_{33}x_3 &= c''_3 \end{aligned} \quad (31-2)$$

که در آن  $a''_3$  و  $c''_3$  از عملیات ریاضی به دست می آیند. سیستم معادلات (31-2) اکنون مثلثی شده و آماده حل است. حل از آخرین معادله شروع می شود که در آن داریم،

$$x_3 = c''_3/a''_{33} \quad (32-2)$$

مقدار  $x_3$  از معادله (32-2) با معادله دوم یعنی (31-2) جایگزین می شود، و برای محاسبه  $x_2$  استفاده می شود. در نهایت، مقادیر  $x_3$  و  $x_2$  به معادله اول (31-2) جایگزین می شوند و  $x_1$  به دست می آید. پرسه حل زمانی آغاز می شود که سیستم معادلات مثلثی شود، همانطور که در معادله (31-2) مشاهده می شود. پرسه حل اشاره شده به صورت ماتریسی بنابر روش زیر خواهد بود،

$$[D] = [A|C] = \begin{bmatrix} a_{11} & a_{12} & a_{13} & c_1 \\ a_{21} & a_{22} & a_{23} & c_2 \\ a_{31} & a_{32} & a_{33} & c_3 \end{bmatrix} \quad (32-2)$$

$$[S_3][S_2][S_1][D] = \begin{bmatrix} a_{11} & a_{12} & a_{13} & c_1 \\ 0 & a'_{22} & a'_{23} & c'_2 \\ 0 & 0 & a''_{33} & c''_3 \end{bmatrix}$$

که در آن،

$$[S_1] = \begin{bmatrix} 1 & 0 & 0 \\ -\frac{a_{21}}{a_{11}} & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, [S_2] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -\frac{a_{31}}{a_{11}} & 0 & 1 \end{bmatrix} \quad (33-2)$$

$$[S_3] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -\frac{a'_{32}}{a'_{22}} & 1 \end{bmatrix}$$

الگوریتم این قسمت در جدول ۲ - ۱ آورده شده است.

جدول ۲ - ۱ الگوریتم روش حذف گاووسی.

$$d_{11} = k_{11}$$

for  $j = 2, \dots, n$

$$g_{m_j, j} = k_{m_j, j}$$

$$g_{ij} = k_{ij} - \sum_{r=m_m}^{i-1} l_{ri} g_{rj} \quad i = m_j + 1, \dots, j-1.$$

where

$$l_{ij} = \frac{g_{ij}}{d_{ii}} \quad i = m_j, \dots, j-1$$

$$d_{jj} = k_{jj} - \sum_{r=m_j}^{j-1} l_{rj} g_{rj}.$$

که در آن ( $m_i, m_m = \text{Max}(m_i, m_j)$ ) شماره ردیف اولین المان غیر صفر در ستون  $i$  است و شماره ردیف اولین المان غیر صفر در ستون  $j$  است.

### ۴-۳-۲- مسائل استاتیکی، روش Skyline

ماتریس سختی به دست آمده در مسئله اجزای محدود همیشه یک ماتریس banded است. بنابراین، روش حل حذف گاوس باید تا حدی اصلاح شود تا ماتریس banded را مدیریت کند، نه یک ماتریس مرتب کامل. الگوریتم حاصل، ظرفیت ذخیره سازی مورد نیاز روش حذف استاندارد گاوس را به میزان قابل توجهی کاهش می‌دهد، علاوه بر این، برش‌های عمده در عملیات ریاضی غیر ضروری برای اعضای صفر که در خارج از باند ماتریس قرار دارند، می‌شود. با این حال، اعضای صفر داخل باند ماتریس در نظر گرفته می‌شوند و عملیات ریاضی لازم برای آنها در طول فرآیند حذف باید انجام شود. روش Skyline اساساً برای حذف برخی از اعضای صفر داخل باند ماتریس سختی ارائه شده است. نتایج، بیشتر در عملیات‌های غیر ضروری ریاضی و کاهش زمان محاسبات می‌باشد. علاوه بر این، ماتریس مستطیلی  $[K]$  به یک آرایه یک بعدی  $\{A\}$  تبدیل می‌شود که منجر به بهبود بیشتر ظرفیت ذخیره‌سازی می‌شود.

برای توصیف روش Skyline، یک ماتریس سختی نرمال  $[K]$  را در نظر بگیرید، همانطور که در شکل ۲ - ۳ نشان داده شده است. فرض بر این است که ماتریس  $[K]$  متقارن است و بنابراین، تنها نیمه بالایی ماتریس نشان داده شده است. شکل ۲ - ۳ نشان می‌دهد که تنها نیمه بالایی ماتریس، از جمله اعضای مورب در نظر گرفته شده است.

Skyline در بالای هر عنصر غیر صفر ستون ترسیم شده است. بنابراین، عناصر بالای Skyline در هر ستون، صفر هستند و عناصر صفر زیر Skyline در محاسبات در نظر گرفته می‌شوند. با بررسی شکل ۲ - ۳، مشخص می‌شود که برخی از اعضای صفر که در ماتریس نواری در نظر گرفته می‌شوند، از عملیات محاسباتی در الگوریتم Skyline حذف می‌شوند. در مورد استراتژی ذخیره سازی، یک سیستم شماره گذاری باید برای ذخیره عناصر زیر Skyline در آرایه یک بعدی  $(I) A$  اتخاذ شود. همانطور که در شکل ۲ - ۳ نشان داده شده است، با دانستن ارتفاع Skyline هر ستون، می‌توان مکان اعضای مورب  $[K]$  را در آرایه یک بعدی  $\langle DK \rangle$  ذخیره کرد. اولین مکان  $\langle DK \rangle$  همیشه ۱ است و آخرین شماره مکان یک

عدد بزرگتر از آخرين مکان ستون در  $[K]$  است. (شماره ۲۳ در شکل ۲ - ۳ نشان می‌دهد که آخرين عضو غیر صفر ماتریس  $[K]$  در ستون ۸ در ۲۲ قرار دارد).

Skyline

(a) 
$$[K] = \begin{bmatrix} k_{11} & k_{12} & 0 & 0 & 0 & 0 & 0 \\ k_{22} & k_{23} & 0 & k_{25} & 0 & 0 & 0 \\ k_{33} & k_{34} & k_{35} & 0 & k_{37} & 0 & 0 \\ k_{44} & k_{45} & k_{46} & 0 & 0 & 0 & 0 \\ k_{55} & k_{56} & k_{57} & 0 & 0 & 0 & 0 \\ k_{66} & k_{67} & k_{68} & 0 & 0 & 0 & 0 \\ k_{77} & k_{78} & 0 & 0 & 0 & 0 & 0 \\ k_{88} & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

(b) 
$$[A] = \begin{bmatrix} A_1 & A_3 & 0 & 0 & 0 & 0 & 0 \\ A_2 & A_5 & 0 & A_{11} & 0 & 0 & 0 \\ A_4 & A_7 & A_{10} & 0 & A_{19} & 0 & 0 \\ A_6 & A_9 & A_{14} & A_{18} & 0 & 0 & 0 \\ A_8 & A_{13} & A_{17} & 0 & 0 & 0 & 0 \\ A_{12} & A_{16} & A_{22} & 0 & 0 & 0 & 0 \\ A_{15} & A_{21} & 0 & 0 & 0 & 0 & 0 \\ A_{20} & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

(c)  $DK(I) = \langle 1 \ 2 \ 4 \ 6 \ 8 \ 12 \ 15 \ 20 \ 23 \rangle$

شکل ۲ - ۳ روش skyline یک ماتریس banded

## فصل سوم

### برنامه کامپیووتری

## مقدمه

در این بخش ابتدا خلاصه‌ای از برنامه کامپیوترا حاضر ارائه می‌شود. قسمت‌های مختلف برنامه برای غشاء الاستیک به سه دسته اصلی (پیش پردازش، پردازش و پس پردازش) تقسیم شده و به طور جداگانه مورد بحث قرار می‌گیرد. این برنامه در محیط C++ نوشته شده است. کد اشاره شده در شکل ۳ - ۱ نمایش داده شده است، که این سه بخش در آن قابل روئیت می‌باشد.

```

void main(void)
{
    char type='b'; //specifies whether to choose the banded or skyline form.
                  //the value 'b' refers to banded form.
                  //the value 's' refers to skyline form.

    mesh_rd_3nte(5.0,8.0); پیش پردازش

    switch(type) {
        case 'b' :
            core_3nte(type);
            boundary_sym_banded(nn,nb,gk,p,const1,const2);
            solve_gauss_sym_banded(nn,nb,gk,p);
            break;
        case 's' :
            bandwidth();
            core_3nte(type);
            boundary_sym_skyline(nn,nb,gk,p,maxh,const1,const2);
            solve_LUdecom_sym_skyline(nn,gk,p,maxh);
            break;
    }

    file_out(); پس پردازش
}

```

شکل ۳ - ۱ نمای کلی کد ابتدایی.

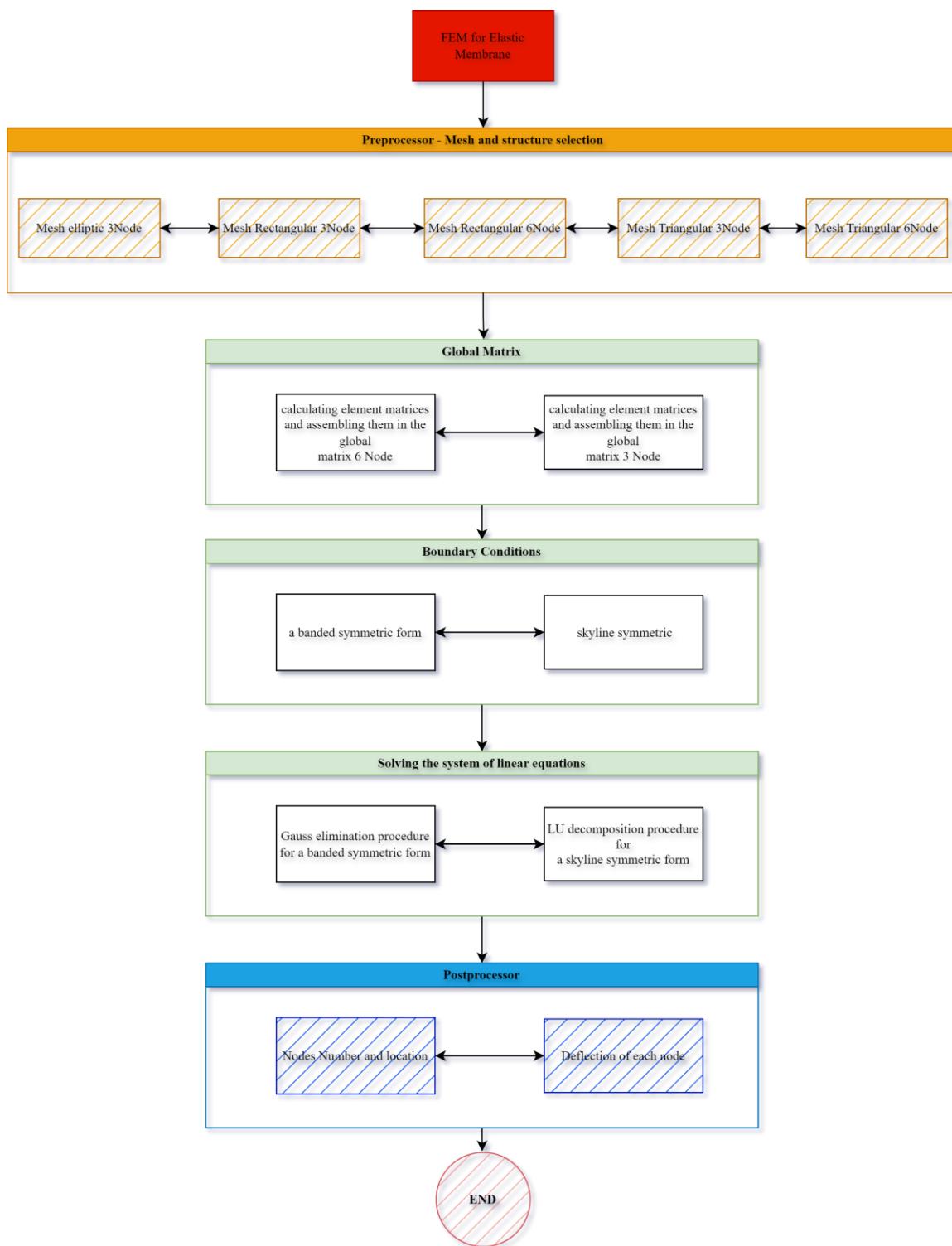
اما در خط اول این کد، عبارت void main(void) باستی تغییر پیدا کند (مرحله صفر - دیباگ)، و به فرم int نوشته شود. خواهیم داشت، شکل ۳ - ۲.

```

101 int main()
102 {
103     char type='b'; //specifies whether to choose the banded or skyline form.
104         //the value 'b' refers to banded form.
105         //the value 's' refers to skyline form.
106
107     mesh_rd_3nte(Xd,Yd);
108
109     switch(type){
110         case 'b' :
111             core_3nte(type);
112             boundary_sym_banded(nn,nb,gk,p,const1,const2);
113             solve_gauss_sym_banded(nn,nb,gk,p);
114             break;
115         case 's' :
116             bandwidth();
117             core_3nte(type);
118             boundary_sym_skyline(nn,nb,gk,p,maxh,const1,const2);
119             solve_LUdecom_sym_skyline(nn,gk,p,maxh);
120             break;
121     }
122
123
124     file_out();
125 }
```

شکل ۳ - ۲ نمای کلی کد اصلاح شده در Dev C++

در خط ۱۰۸، اینبار به جای استفاده از اعداد ۵ و ۸ برای طول و عرض غشاء الاستیک، از حالت پارامتری  $Xd$  و  $Yd$  استفاده شده است و در خطهای ماقبل بدنه اصلی کد تعریف می‌شوند یا می‌توانند از کاربر دریافت شوند. فلوچارت این الگوریتم کد C++ در شکل ۳-۳ رسم شده است.



شکل ۳ - ۳ فلوچارت اجرای کد المان محدود C++ برای غشاء الاستیک.

### ۱-۳- پیش‌پردازش

در واحد پیش‌پردازش، پنج گزینه مختلف برای استفاده در نظر گرفته شده است که در حالت اولیه بر روی mesh\_rd\_3nte قرار گرفته است، ولی می‌توان این حالت را با توجه به مسئله و احتیاج کاربر تغییر داد.

۱- زیربرنامه mesh\_rd\_3nte.cpp، که مش را برای یک دامنه مستطیلی توسط المان مثلثی با سه گره ایجاد می‌کند. ورودی‌های این برنامه طول و عرض دامنه مستطیلی و تعداد تقسیمات در هر جهت می‌باشد.

۲- زیربرنامه mesh\_rd\_6nte.cpp، که مش را برای یک دامنه مستطیلی توسط المان مثلثی با شش گره ایجاد می‌کند. ورودی‌های این برنامه طول و عرض دامنه مستطیلی و تعداد تقسیمات در هر جهت می‌باشد.

۳- زیربرنامه mesh\_td\_3nte.cpp، که مش را برای یک دامنه مثلثی قائم الزاویه توسط المان مثلثی با سه گره ایجاد می‌کند. ورودی‌های این برنامه ابعاد مثلث و تعداد تقسیمات است که در هر دو جهت برابر فرض می‌شود.

۴- زیربرنامه mesh\_td\_6nte.cpp، که مش را برای یک دامنه مثلثی قائم الزاویه توسط المان مثلثی با شش گره ایجاد می‌کند. ورودی‌های این برنامه ابعاد مثلث و تعداد تقسیمات است که در هر دو جهت برابر فرض می‌شود.

۵- زیربرنامه mesh\_ed\_3nte.cpp، که مش را برای یک دامنه بیضی‌گون توسط المان مثلثی با شش گره ایجاد می‌کند. ورودی‌های این برنامه ابعاد بیضی و تعداد تقسیمات در جهت شعاعی و همچنین تعداد تقسیمات در جهت زاویه‌ای برای لایه اول عناصر در اطراف مرکز بیضی است.

در اجرای روش skyline، لازم است ارتفاع هر ستون به طور جداگانه محاسبه شود و سپس از این ارتفاعات برای یافتن محل ترم‌های مختلف در ماتریس سختی استفاده شود. ارتفاع جداگانه ستون‌ها در این روش توسط زیربرنامه bandwidth.cpp محاسبه می‌شود.

## ۲-۳ - پردازش

واحد پردازش که بخش مرکزی کد اصلی می‌باشد، برای دو موقعیت مختلف برنامه‌ریزی شده است. در حالت اول، ماتریس سختی به صورت متقارن نواری در نظر گرفته شده است. سپس شرایط مرزی بر روی این شکل نواری از ماتریس سختی اعمال می‌شود و سپس روش حذف گاوس برای حل آن فراخوانی می‌شود. در موقعیت دوم، مفهوم skyline فراخوانی می‌شود و عناصر غیر صفر ماتریس سختی در آرایه‌ای ذخیره می‌شوند که از نظر ابعاد فشرده‌تر است. سپس شرایط مرزی بر روی این آرایه اعمال می‌شود و سپس روش تجزیه LU برای به دست آوردن متغیرهای اولیه ناشناخته اجرا می‌شود. واحد پردازش که بیشترین زمان محاسباتی در آن صرف می‌شود، از سه بخش اصلی تشکیل شده است:

۱ - زیربرنامه‌ای برای محاسبه ماتریس‌های المان‌ها و ترکیب آنها در ماتریس سختی نهایی. این کار در زیربرنامه core-3nte.cpp برای المان مثلثی با سه گره و در زیربرنامه core\_6nte.cpp برای المان مثلثی با شش گره انجام می‌شود. مونتاژ و ترکیب ماتریس‌های المان به محض محاسبه آنها انجام می‌شود.

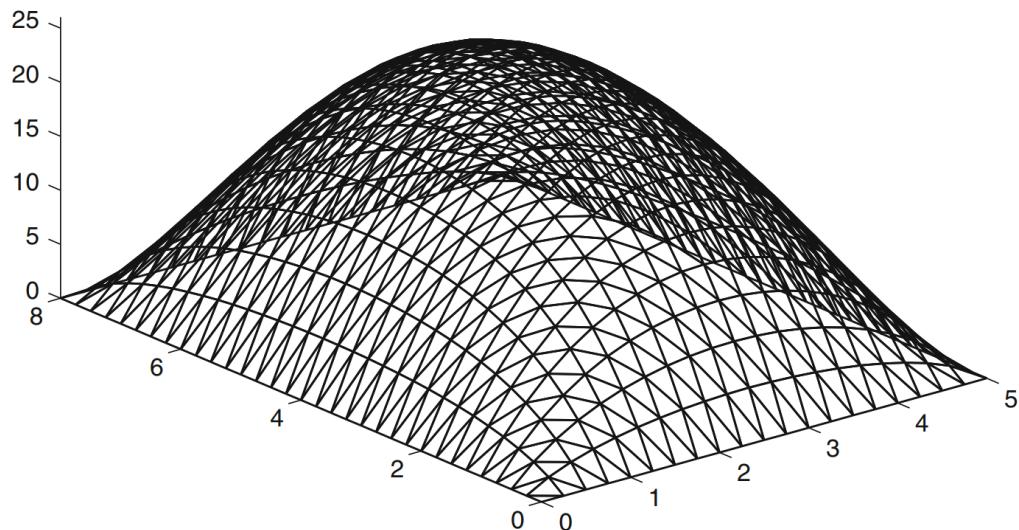
۲ - تعیین شرایط مرزی بر متغیرهای اولیه. این کار در زیربرنامه boundary\_sym\_banded.cpp برای فرم متقارن نواری و در زیربرنامه boundary\_sym\_skyline.cpp برای فرم متقارن skyline انجام می‌شود.

۳ - حل سیستم معادلات خطی برای به دست آوردن متغیرهای اولیه مجھول. مشابه مرحله قبل، دو زیربرنامه مختلف برای این منظور در نظر گرفته شده است. اولین زیربرنامه banded solve\_gauss\_sym\_banded.cpp استفاده می‌کند. دومین زیربرنامه solve\_LUdecom\_sym\_skyline.cpp از روش تجزیه LU برای یک فرم متقارن skyline استفاده می‌کند.

یکی از جنبه‌های مهم اجرای کامپیوتری روش حل حذف گاوس این است که باید از حداقل زمان حل استفاده شود. استفاده از روش skyline، به رغم پیچیدگی بیشتر آن، در مواردی قابل توجیه است که نه تنها عناصر خارج از باند ماتریس سختی صفر هستند، بلکه بسیاری از عناصر داخل باند نیز صفر هستند.

### ۳-۳- پس‌پردازش

پس از حل سیستم خطی معادلات جبری در واحد پردازش، نتایج را باید در قالب مناسب گزارش نمود. در واحد پس‌پردازندۀ نتایج روی فایل‌های خاصی نوشته می‌شوند تا با نرم‌افزار مناسب (مانند MATLAB یا MATHEMATICA) خوانده شوند و به صورت دو بعدی و سه بعدی ترسیم شوند. زیربرنامه file\_out.cpp این کار را انجام می‌دهد و نتایج را در خروجی‌هایی گزارش می‌دهند. نمونه‌ای از نتایج در شکل ۳ - ۴ نشان داده شده است [۱۰].



شکل ۳ - ۴ شکل تغییر شکل غشاء برای مش مثلثی سه گره یک دامنه مستطیلی [۱۰]

## فصل چهارم

### نتایج

## مقدمه

در این بخش ابتدا صحت‌سنجی پژوهش با استفاده از ابزار حل معادلات PDE انجام می‌شود (حل معادله پواسون)، سپس همگرایی پاسخ خیز با تعداد المان‌های مثلثی سه گره و شش گره مورد بررسی قرار می‌گیرد و در انتهای اثر پارامترهای مختلف بر روی خیز ارائه می‌شود.

### ۱-۴- ابزار صحت‌سنجی

برای حالت ابتدایی با غشاء به طول ۸ و عرض ۵، برای دو حالت مشبندی مثلثی با سه و شش گره، صحت‌سنجی و همگرایی تعداد المان‌ها مورد بررسی قرار می‌گیرد. ابزار حل معادلات جزئی که برای صحت‌سنجی مطالعه در این قسمت در نظر گرفته شده است، معادلاتی به فرم زیر را می‌تواند حل نماید.

$$m \frac{\partial^2 u}{\partial t^2} + d \frac{\partial u}{\partial t} - \nabla(c \nabla u) + au = f \quad (1-4)$$

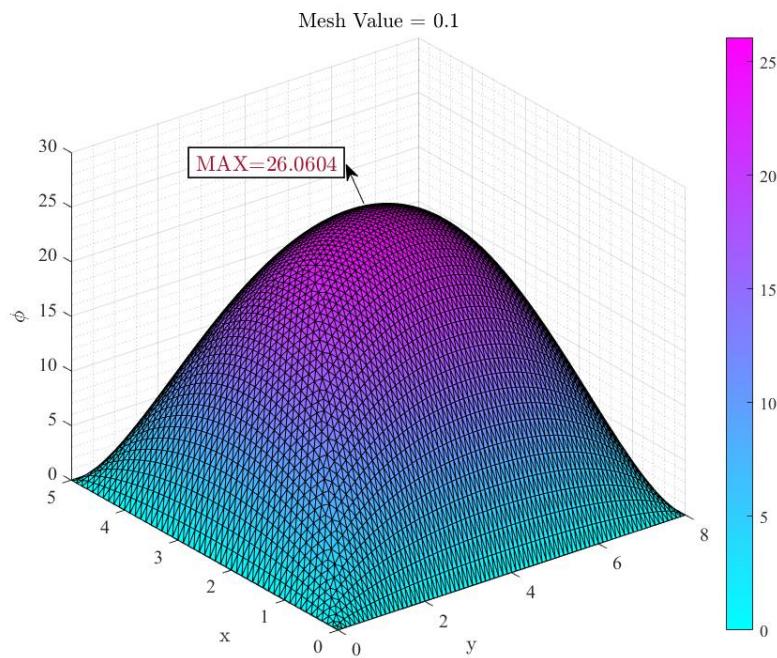
معادله غشاء، معادله پواسون به فرم زیر می‌باشد،

$$-\nabla^2 u = f \quad (2-4)$$

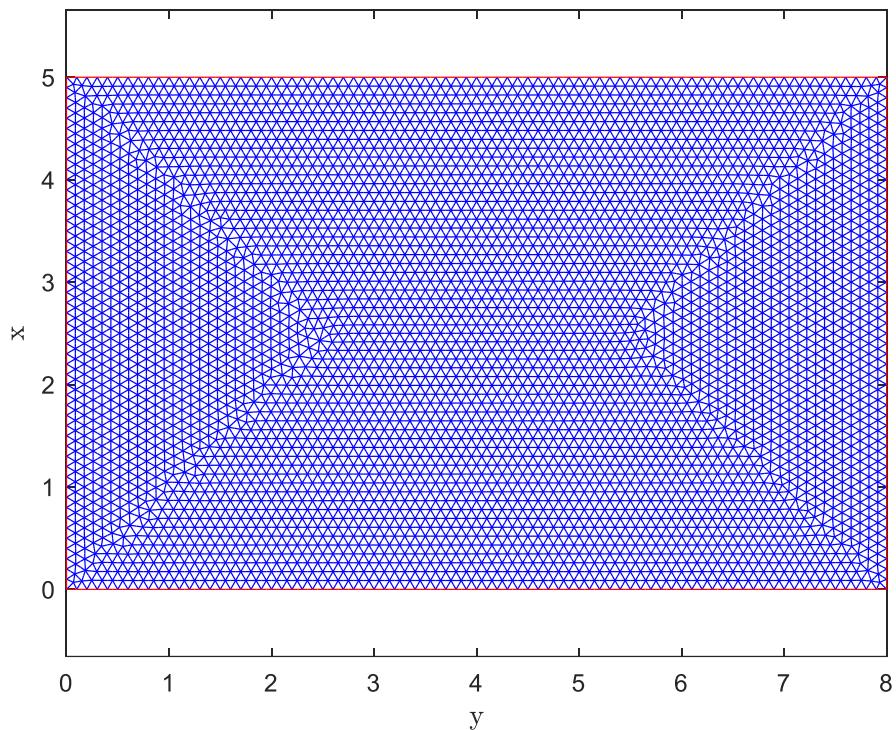
که  $f$  بیانگر فشار نسبت به نیروی کششی می‌باشد که در برنامه کامپیوتوری ابتدایی برابر ده در نظر گرفته شده است. در فصل دوم، معادله ۱-۲ نیز به همین موضوع اشاره دارد. با در نظر گرفتن مقادیر به فرم زیر،

$$m = 0, d = 0, c = -1, f = -10 \quad (3-4)$$

و اندازه مش ۰.۱، و شرط دیریکله و تغییرات صفر برای دورتا دور مستطیل خواهیم داشت،

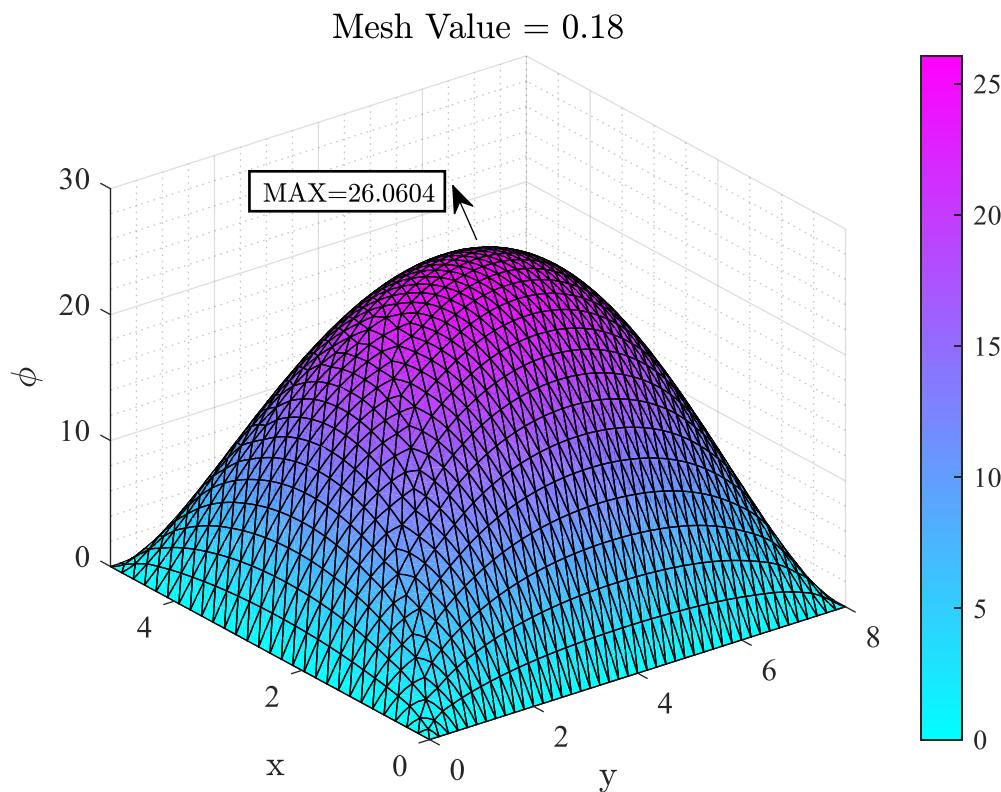


شکل ۴ - ۱ خیز غشاء مستطیلی، محاسبه شده از حل معادله پواسون توسط ابزار حل معادلات جزئی با مش ۰.۱.

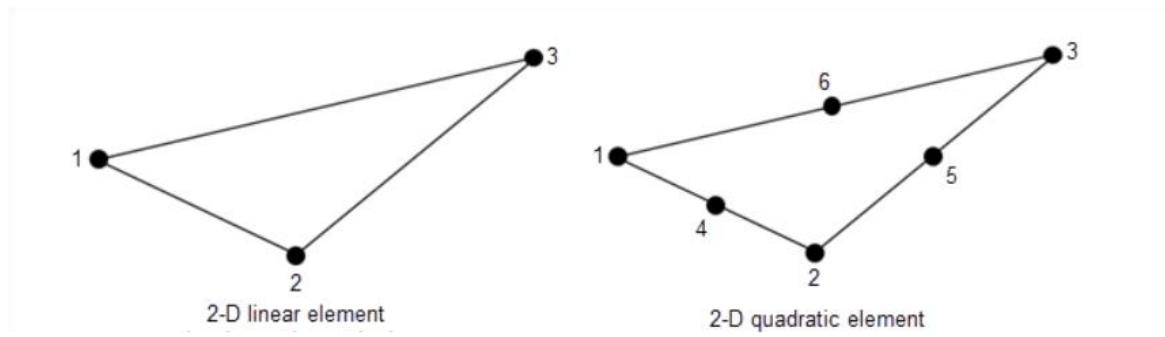


شکل ۴ - ۲ نمایش مش ۰.۱ در محاسبه خیز غشاء مستطیلی، حل معادله پواسون توسط ابزار حل معادلات جزئی.

مشبندی در این حالت در شکل ۴ - ۲ نمایش داده شده است، برای المان‌های بزرگ‌تر، اندازه ۰.۱۸ نیز نتایج مشابهی مشاهده می‌شود، پس می‌توان نتیجه گرفت، اندازه مش ۰.۱۸ برای صحتسنجی از نظر همگرایی نیز مناسب است، شکل ۴ - ۳. در قسمت صحتسنجی با ابزار حل معادلات جزئی، از مش مثلثی کوادراتیک، همانطور که در شکل ۴ - ۴ نشان داده شده است، استفاده شده است.



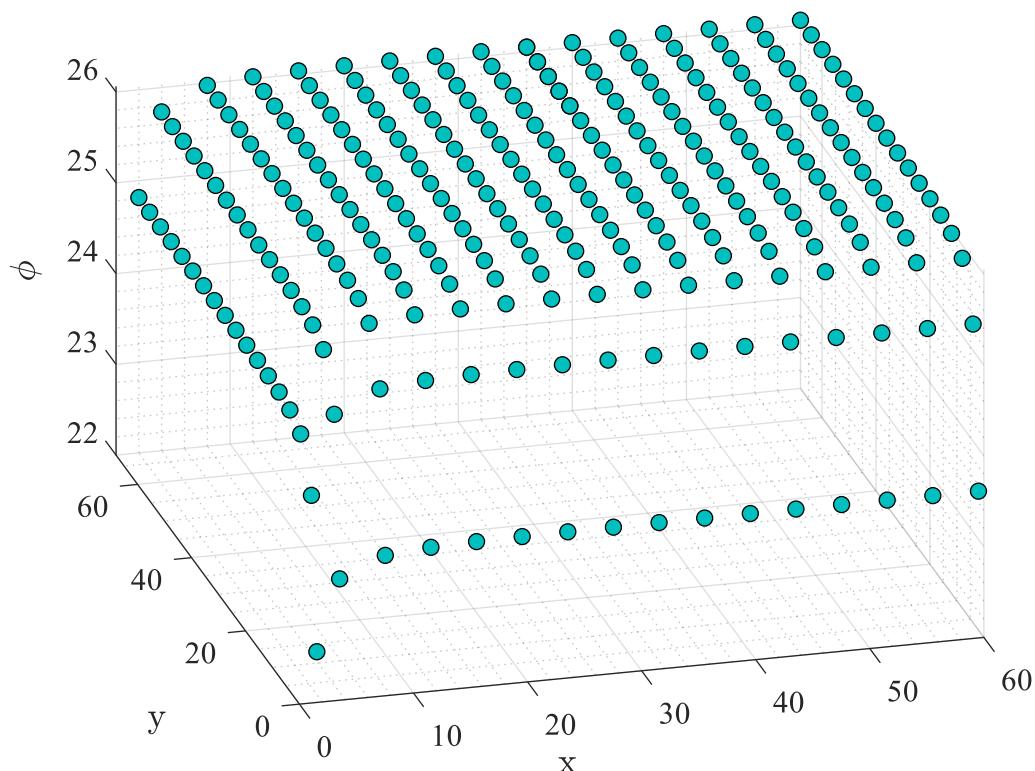
شکل ۴ - ۳ خیز غشاء مستطیلی، محاسبه شده از حل معادله پواسون توسط ابزار حل معادلات جزئی با مش ۰.۱۸.



شکل ۴ - ۴ نمایش انواع مش مثلثی دو بعدی در محاسبه خیز غشاء مستطیلی، حل معادله پواسون توسط ابزار حل معادلات جزئی.

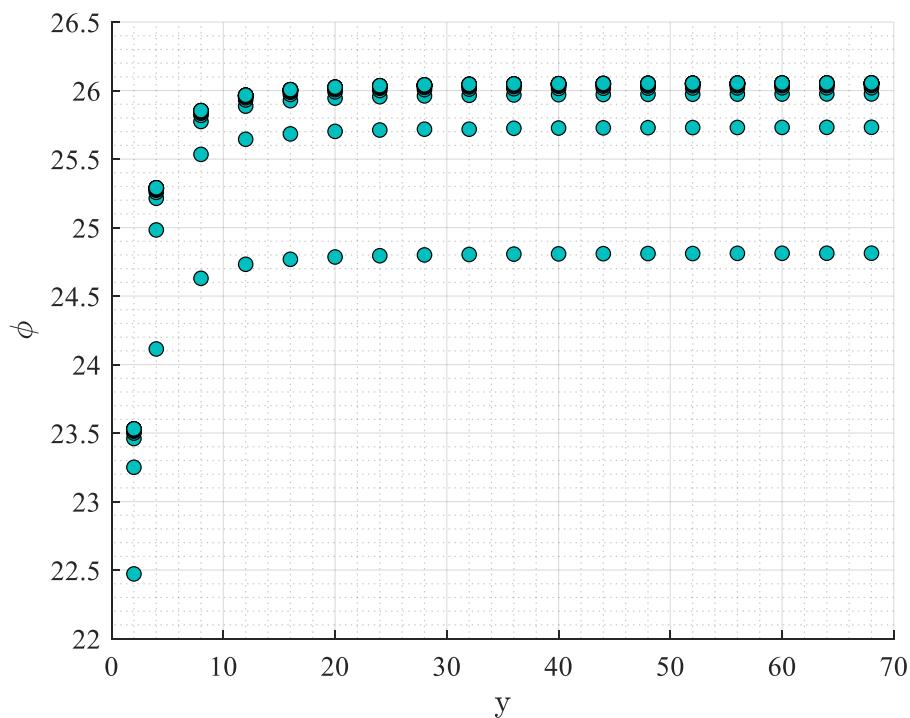
## ۴-۲- مطالعه همگرایی کد کامپیوتروی

تعداد المان‌ها در راستاهای طول و عرض تغییر می‌کنند تا به نتایج مطلوبی برسیم و از آن به بعد بقیه نتایج با تعداد المان‌های مناسب گزارش شوند. در این راستا، شکل ۴ - ۵ ارائه شده است. همانطور که مشخص است با افزایش تعداد المان در هر راستا، باعث بهبود پاسخ و نزدیک شدن به پاسخ اصلی (۲۶.۰۶) می‌شویم.

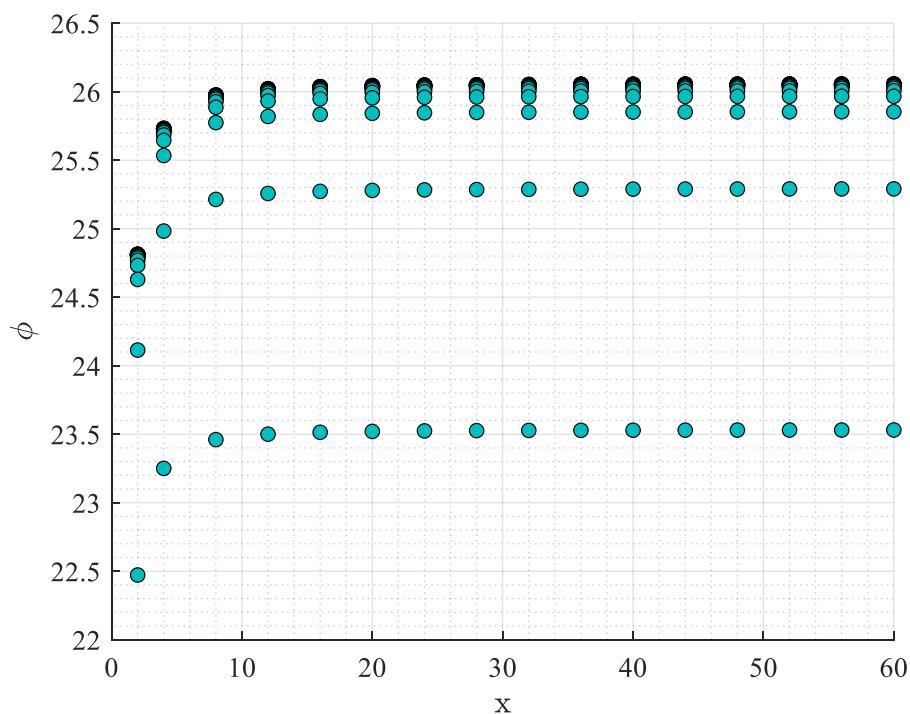


شکل ۴ - ۵ نمایش سه بعدی همگرایی المان مثلثی با ۳ گره تعداد المان‌های طولی و عرضی برای یک غشاء  $8 \times 5$  مستطیلی.

برای همگرایی در راستاهای طولی و عرضی داریم، شکل ۴ - ۶ و شکل ۴ - ۷. همانطور که مشخص است، بعد از تعداد المان ۸ برابر اندازه طول، ۴۰ برای راستای طولی و ۶۴ برای المان عرضی، نتایج به طور مناسب همگرا هستند (با خطای ۰.۰۰٪). جدول همگرایی‌ها و مقدار خطاهای در پیوست ارائه شده‌اند.

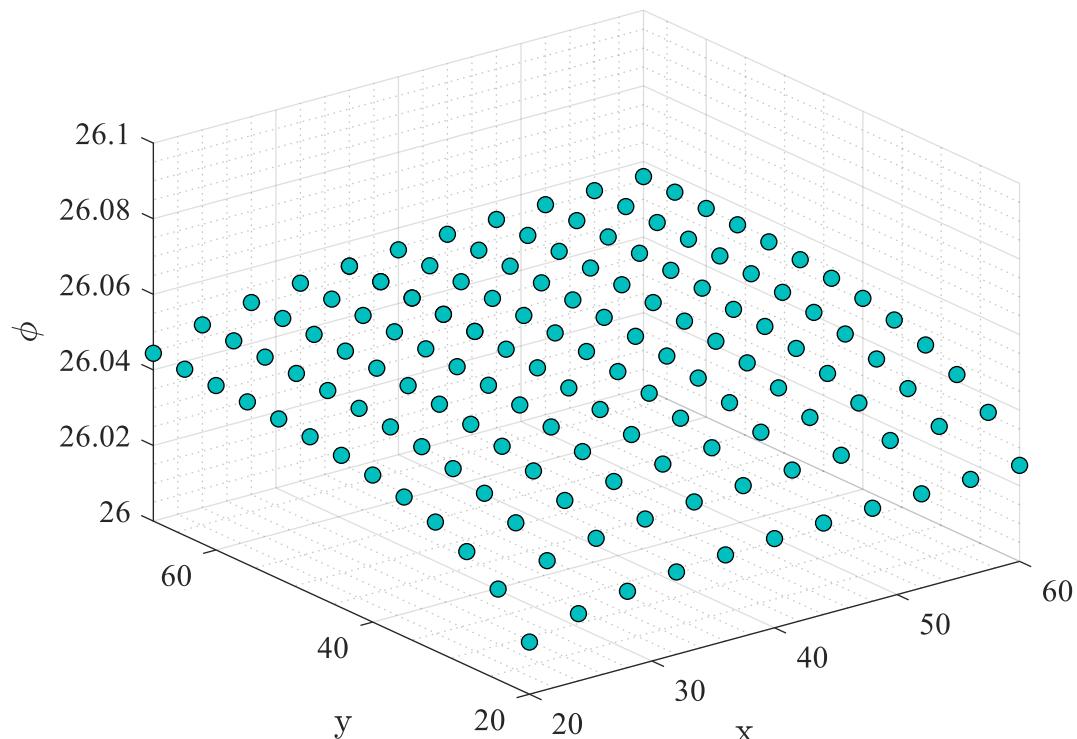


شکل ۴ - ۶ نمایش دو بعدی همگرایی المان مثلثی با ۳ گره تعداد المان های عرضی برای یک غشاء  $8 \times 5$  مستطیلی.

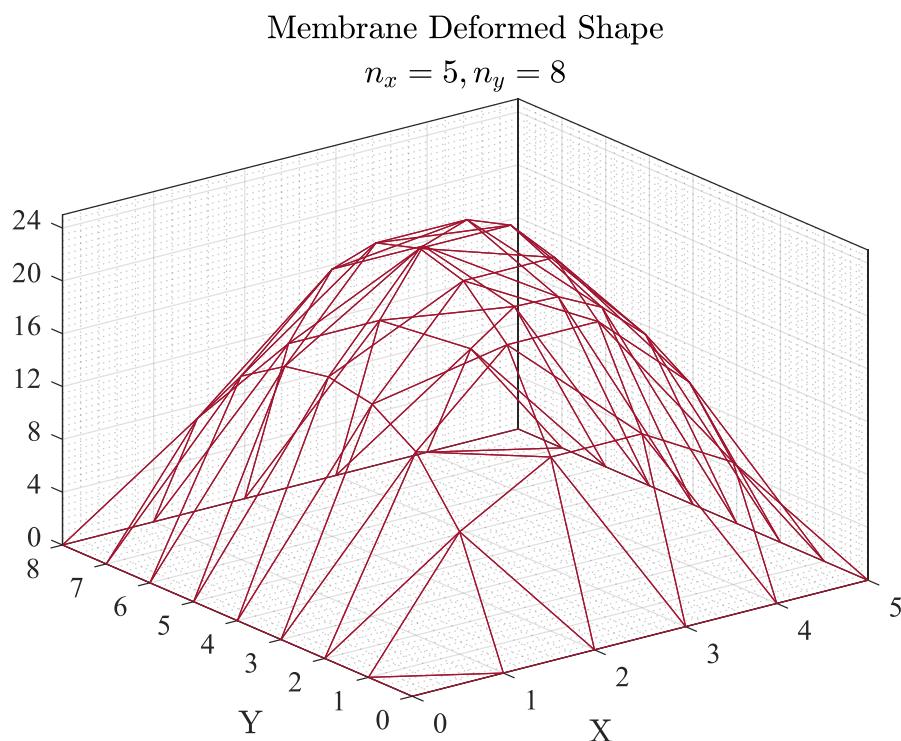


شکل ۴ - ۷ نمایش دو بعدی همگرایی المان مثلثی با ۳ گره تعداد المان های طولی برای یک غشاء  $8 \times 5$  مستطیلی.

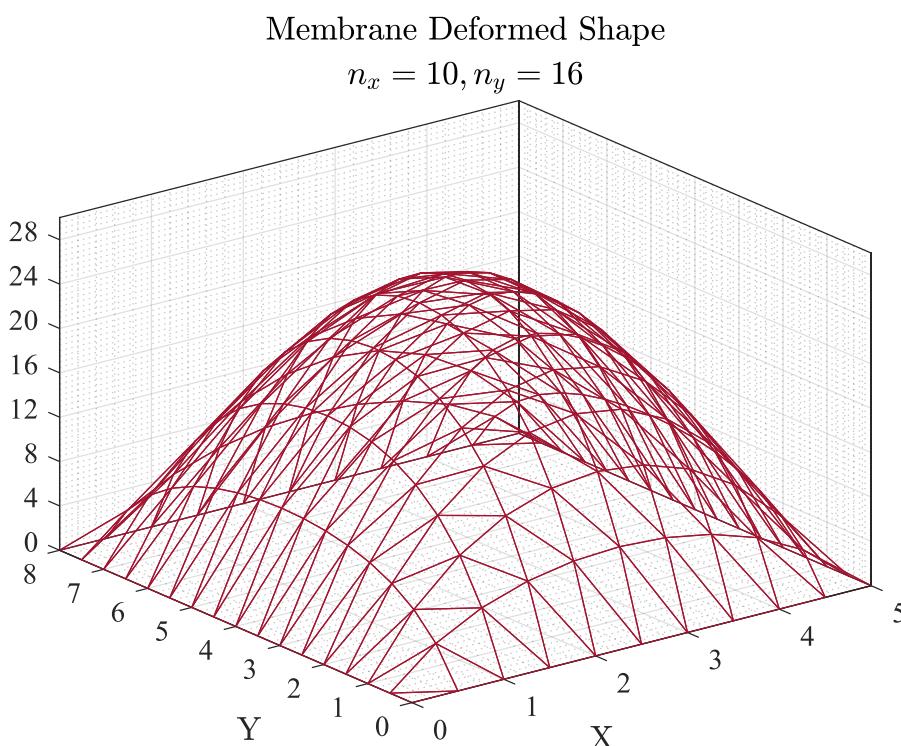
نمای جزئی از مقدار همگرایی در شکل ۴ - ۸ نمایش داده شده است. مشاهده می‌شود با افزایش تعداد المان‌ها، همگرایی به نتیجه مطلوب قابل حصول است، البته باید توجه داشت که زمان محاسبات افزایش می‌یابد، که برای کد غشاء از این مقدار برای تعداد المان کم، به دلیل اینکه زیر یک ثانیه می‌باشند، می‌توان صرف نظر کرد. قابل توجه است که کد C++ موجود، با رایانه‌ای با ویندوز ۱۱ و پردازنده اینتل (Intel(R)) و رم ۱۶ گیگابایت اجرا شده‌اند.



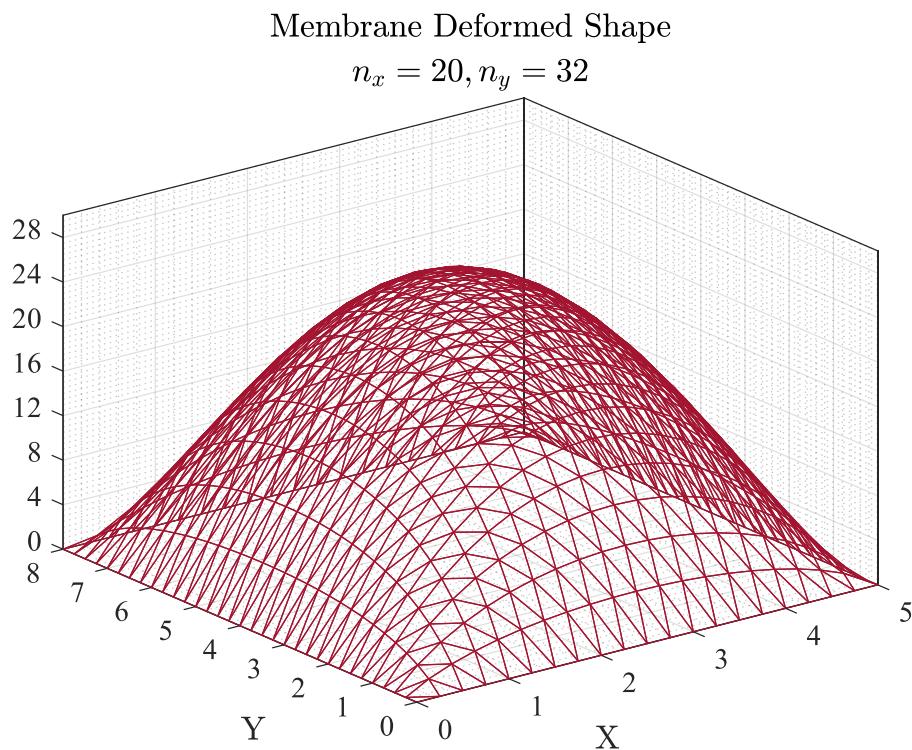
شکل ۴ - ۸ نمای جزئی نمایش سه بعدی همگرایی المان مثلثی با ۳ گره تعداد المان‌های طولی برای یک غشاء  $8 \times 5$  مستطیلی.



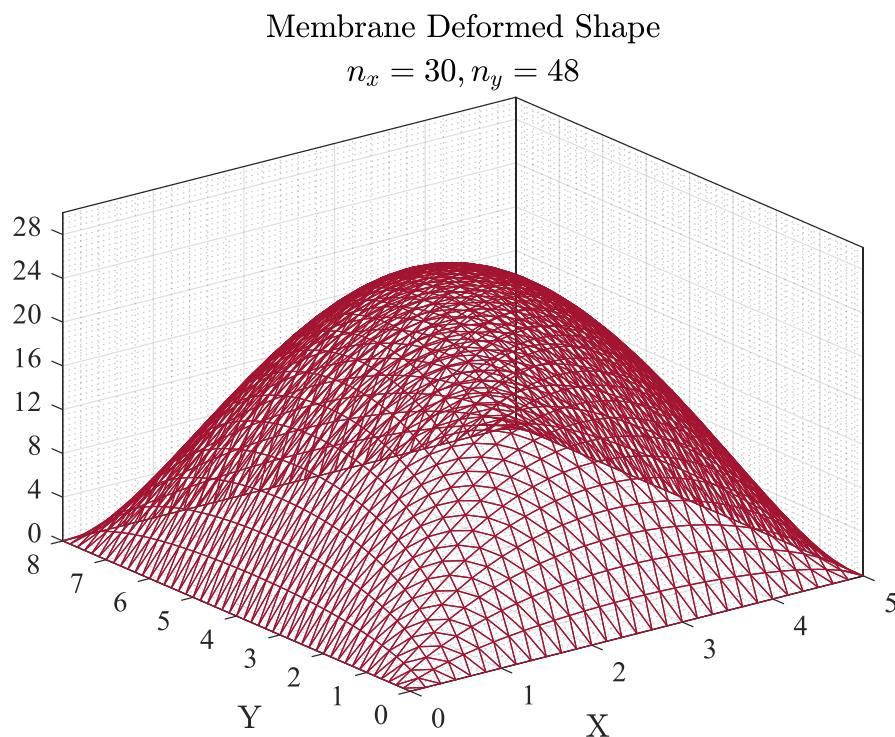
شکل ۴ - ۹ نمای خیز غشاء مستطیلی با المان مش مثلثی سه گره برای ۸



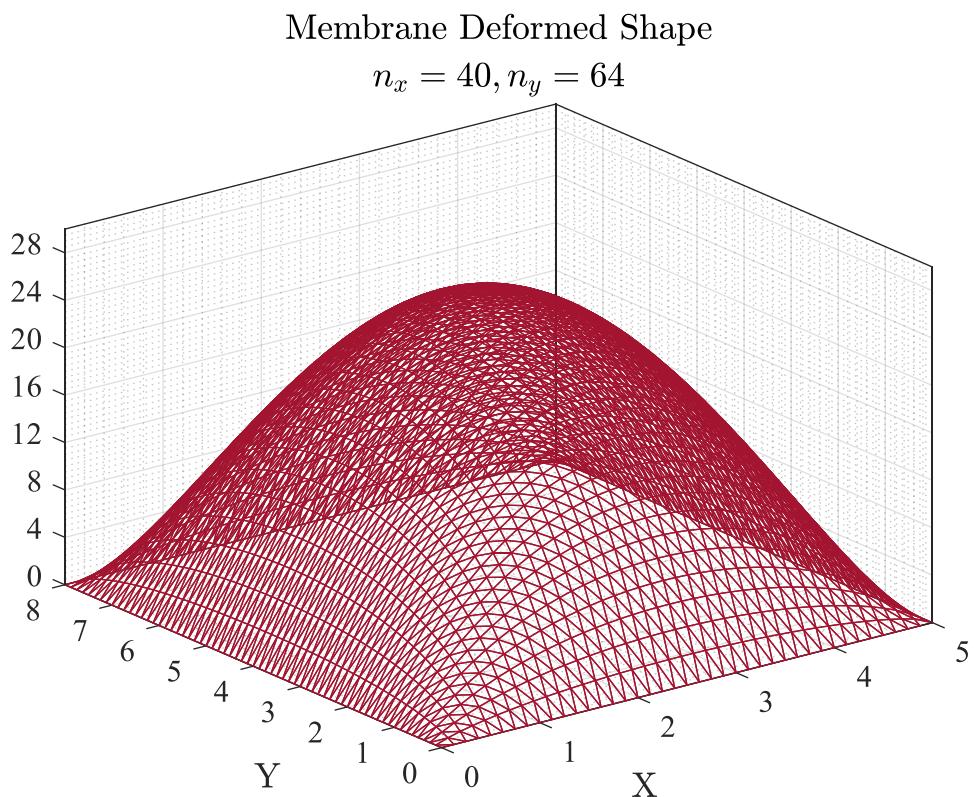
شکل ۴ - ۱۰ نمای خیز غشاء مستطیلی با المان مش مثلثی سه گره برای ۱۶



شکل ۴ - ۱۱ نمای خیز غشاء مستطیلی با المان مش مثلثی سه گره برای  $n_x = 20, n_y = 32$

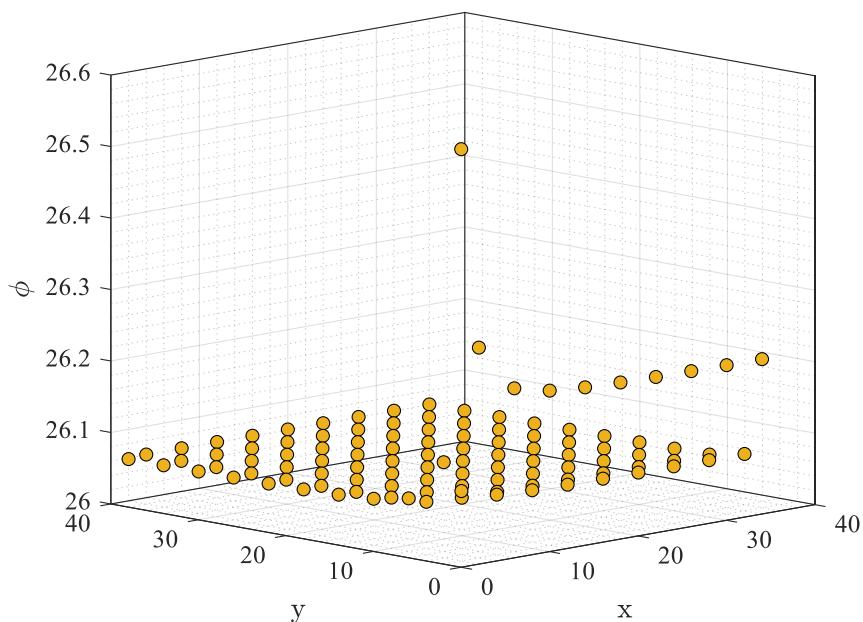


شکل ۴ - ۱۲ نمای خیز غشاء مستطیلی با المان مش مثلثی سه گره برای  $n_x = 30, n_y = 48$

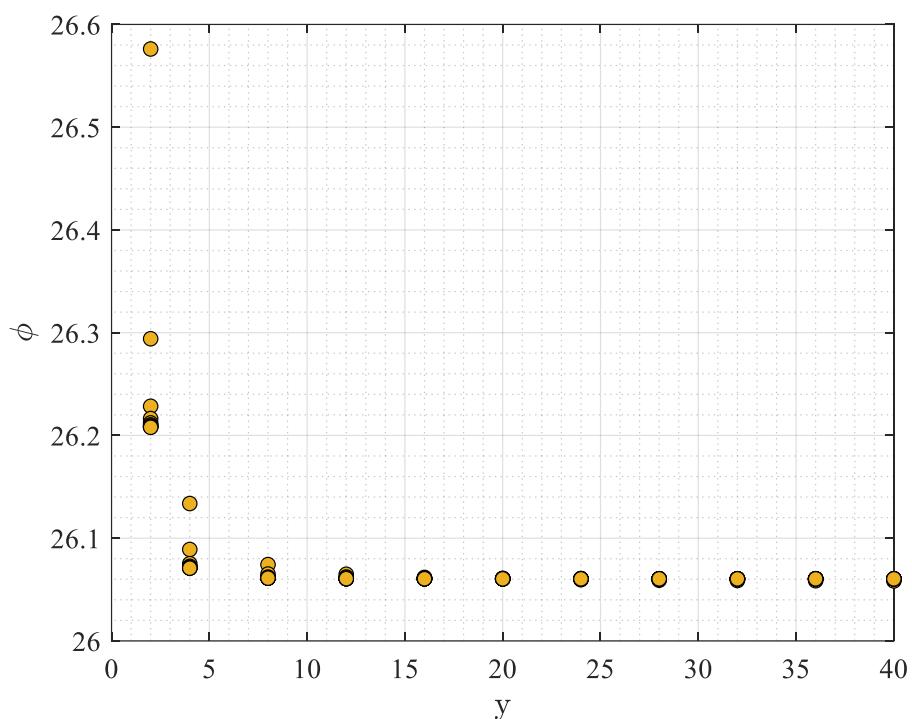


شکل ۴ - ۱۳ نمای خیز غشاء مستطیلی با المان مش مثلثی سه گره برای  $n_x = 40, n_y = 68$

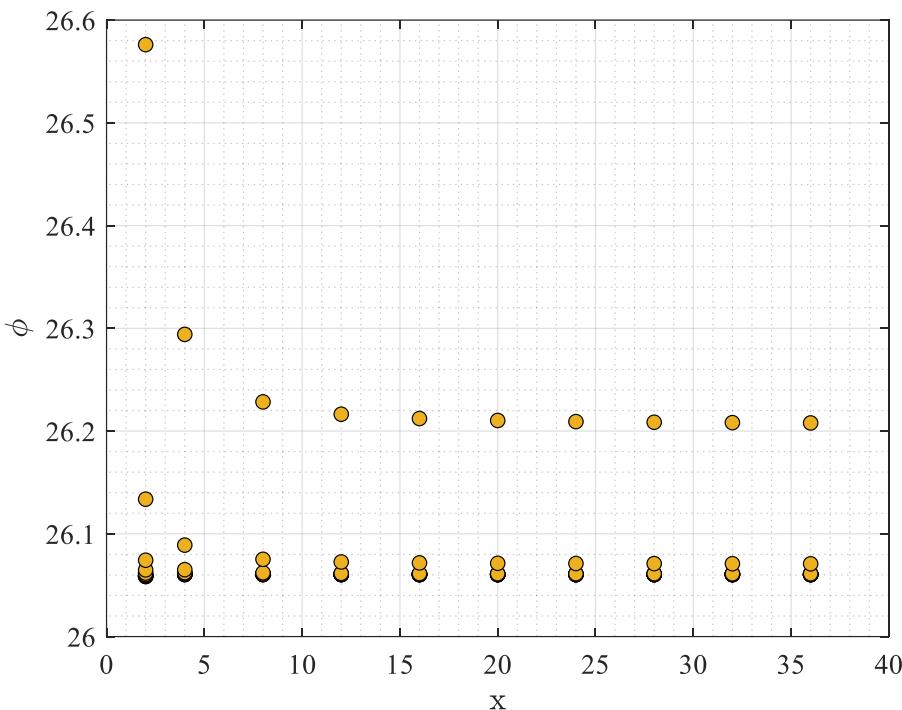
نمودارهای ارائه شده در بخش ۴-۲ تا کنون برای حالت mesh\_rd\_3nte اجرا شده‌اند. برای همگرایی حالت mesh\_rd\_6nte خواهیم داشت، شکل ۴ - ۱۴، شکل ۴ - ۱۵ و شکل ۴ - ۱۶.



شکل ۴ - ۱۴ نمایش سه بعدی همگرایی المان مثلثی با ۶ گره تعداد المان‌های طولی و عرضی برای یک غشاء  $8 \times 5$  مستطیلی.

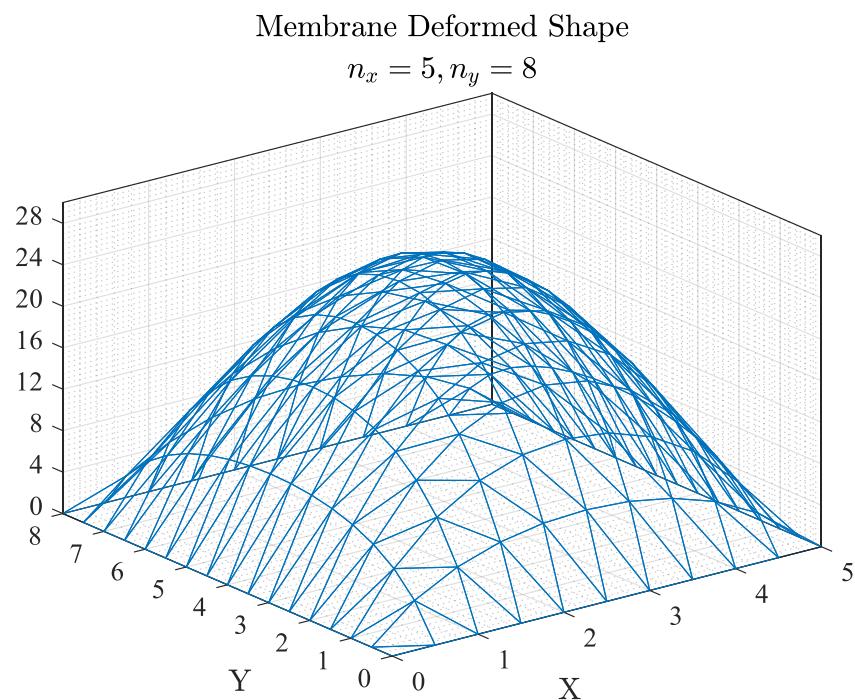


شکل ۴ - ۱۵ نمایش دو بعدی همگرایی المان مثلثی با ۶ گره تعداد المان‌های عرضی برای یک غشاء  $8 \times 5$  مستطیلی.

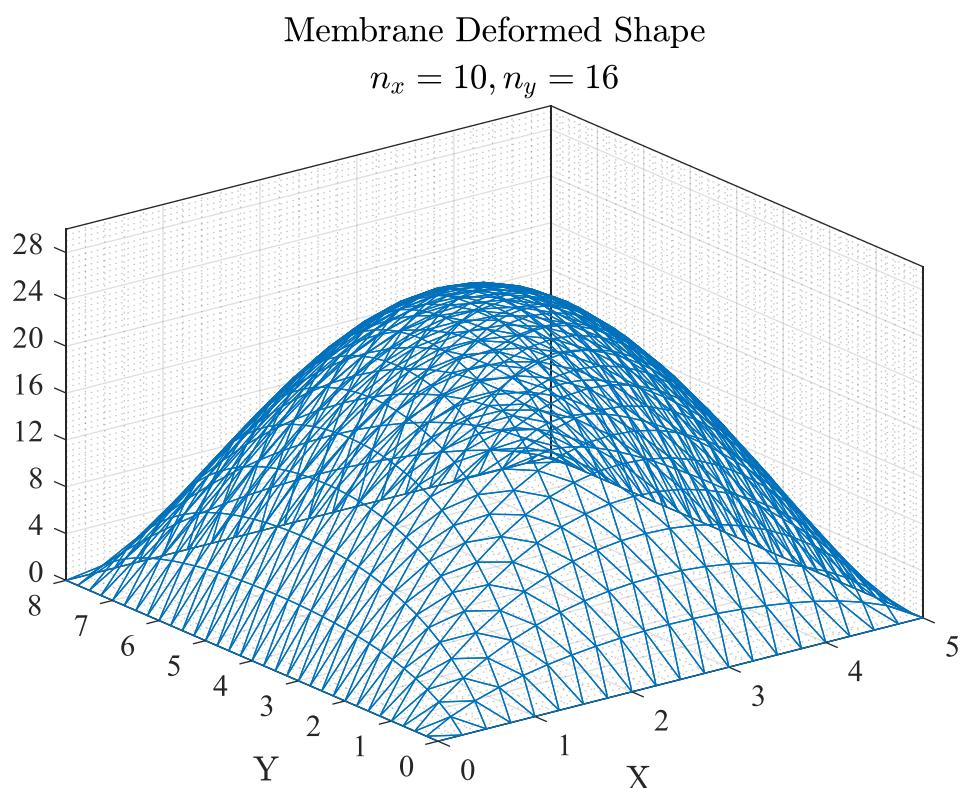


شکل ۴ - ۱۶ نمایش دو بعدی همگرایی المان مثلثی با ۶ گره تعداد المان‌های طولی برای یک غشاء  $8 \times 5$  مستطیلی.

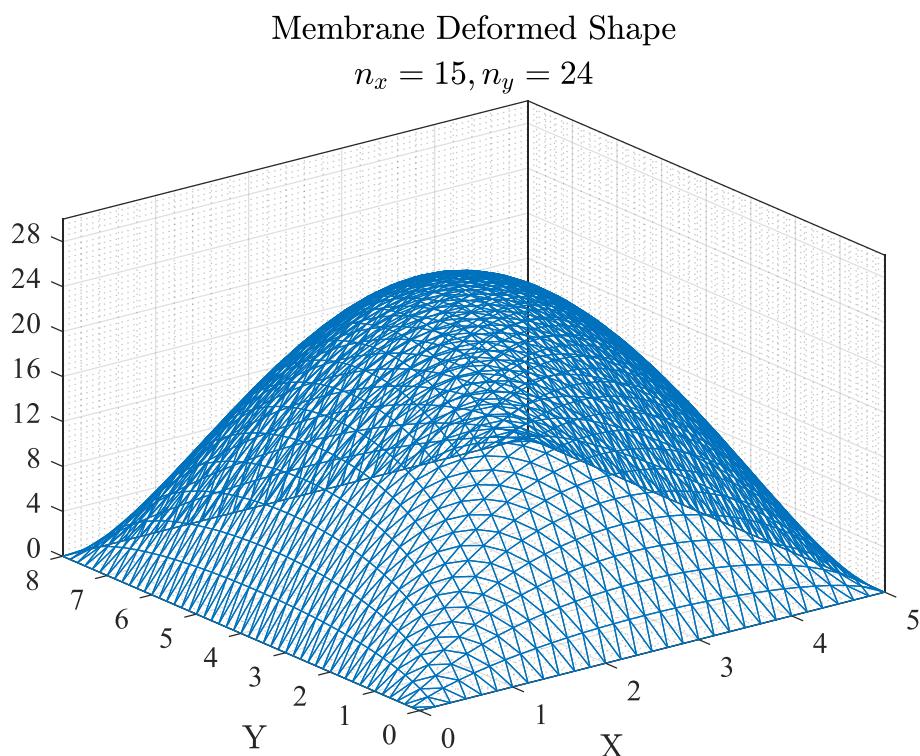
با استفاده از جداول همگرایی می‌توان متوجه شد که اندازه مش  $15 \times 24$ ، یعنی تقسیم هر اندازه به سه قسمت، می‌تواند خیز را به مقدار  $26.0604$  تخمین بزند که اندازه مطلوب ما تا چهار رقم اعشار می‌باشد (خطای زیر  $0.0003$  درصد). نمایهایی از خیزهای حاصل شده با تعداد المان‌های طولی و عرضی متفاوت در شکل‌های ۱۷-۴ تا ۲۰-۴ رسم شده است.



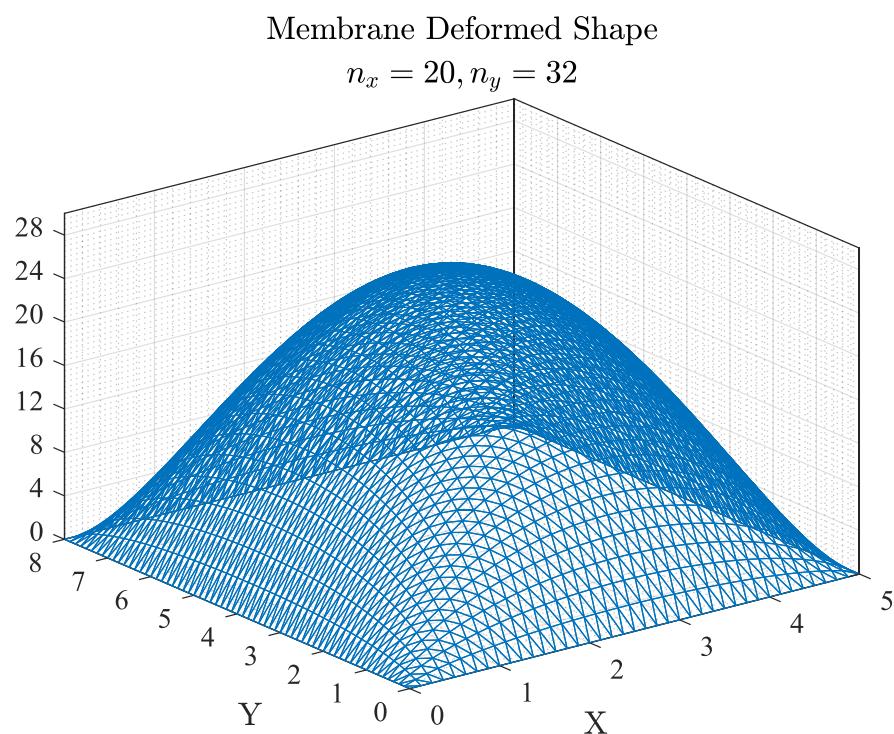
شکل ۴ - ۱۷ نمای خیز غشاء مستطیلی با المان مش مثلثی شش گره برای  $n_x = 5, n_y = 8$



شکل ۴ - ۱۸ نمای خیز غشاء مستطیلی با المان مش مثلثی شش گره برای  $n_x = 10, n_y = 16$



شکل ۴ - ۱۹ نمای خیز غشاء مستطیلی با المان مش مثلثی شش گره برای  $n_x = 15, n_y = 24$

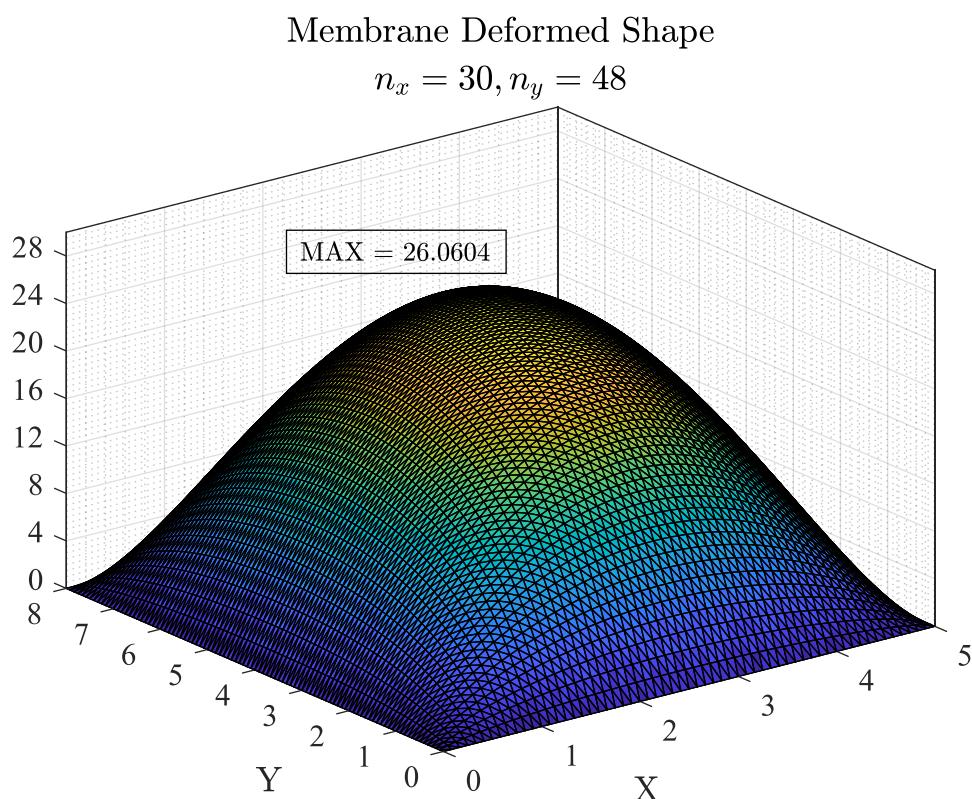


شکل ۴ - ۲۰ نمای خیز غشاء مستطیلی با المان مش مثلثی شش گره برای  $n_x = 20, n_y = 32$

### ۳-۴- مطالعه پارامتری

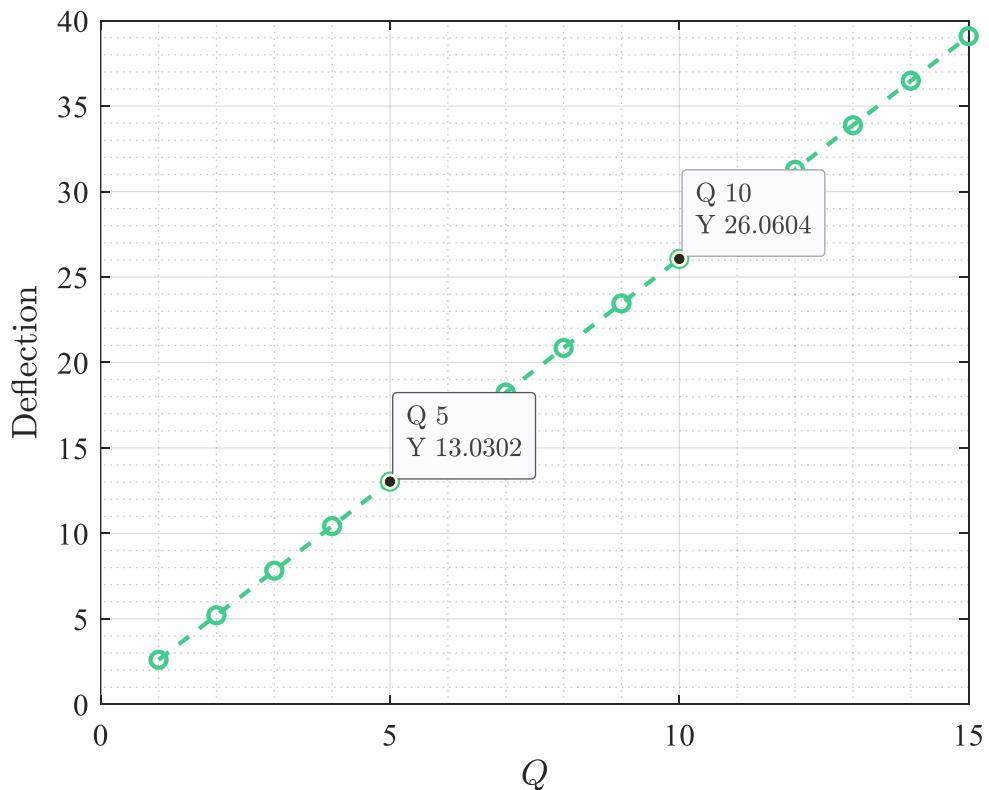
در این بخش به اثر تغییر فشار روی غشاء و همچنین اثر اندازه هندسی بر روی خیز پرداخته می شود. در نمودارهای بخش قبلی، غشاء مستطیلی  $8 \times 5$  با فشار ۱۰ و کشش سطحی ۱ در دو جهت طولی و عرضی محاسبه شد. در این بخش از المان های مثلثی با ۶ گره و همچنین تعداد المان ۵ برابر در هر راستا در نظر گرفته می شود که همگرایی آن در بخش قبل اثبات شد. برای حالت فشار به کشش ۱۰ و سطح مقطع  $8 \times 5$  در به نمایش گذاشته شده است، که اندازه خیز آن برابر  $26.0604$  می باشد، شکل ۴ -

.۲۱



شکل ۴ - ۲۱ تغییرات خیز غشاء مستطیلی در حالت اولیه مقطع  $8 \times 5$  مستطیلی.

در شکل ۴ - ۲۲ نشان داده شده است که افزایش فشار وارد بر غشاء، به افزایش خیز غشاء منجر می شود که این تغییرات به علت استاتیکی بودن رابطه به صورت خطی می باشد.

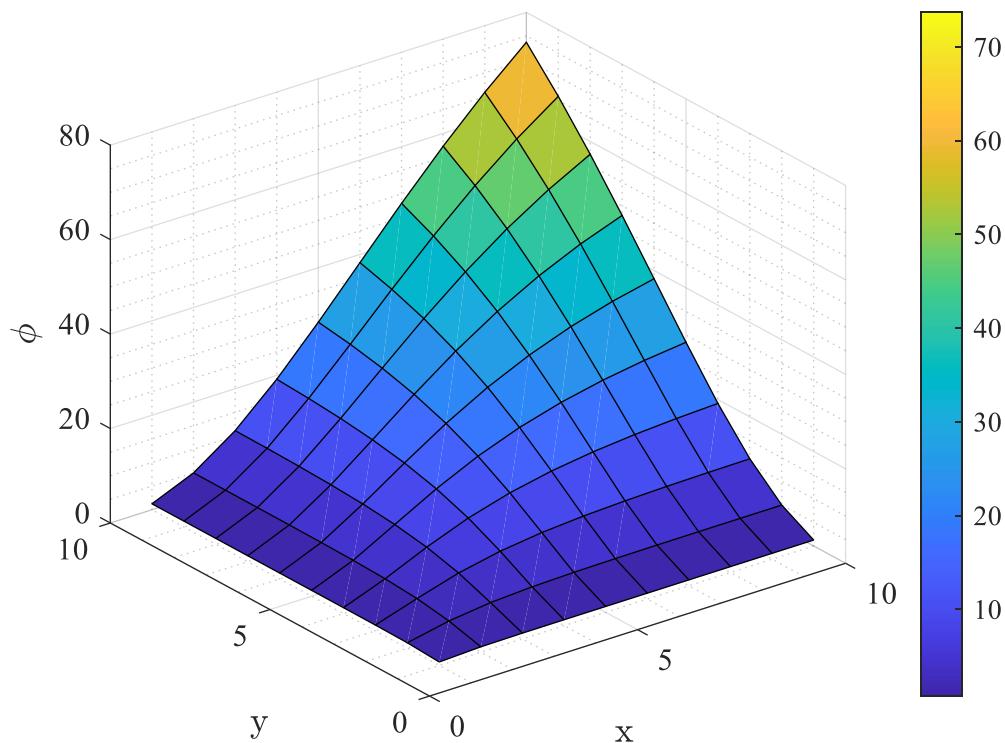


شکل ۴ - ۲۲ تغییرات خیز غشاء مستطیلی با تغییرات فشار وارد بر غشاء.

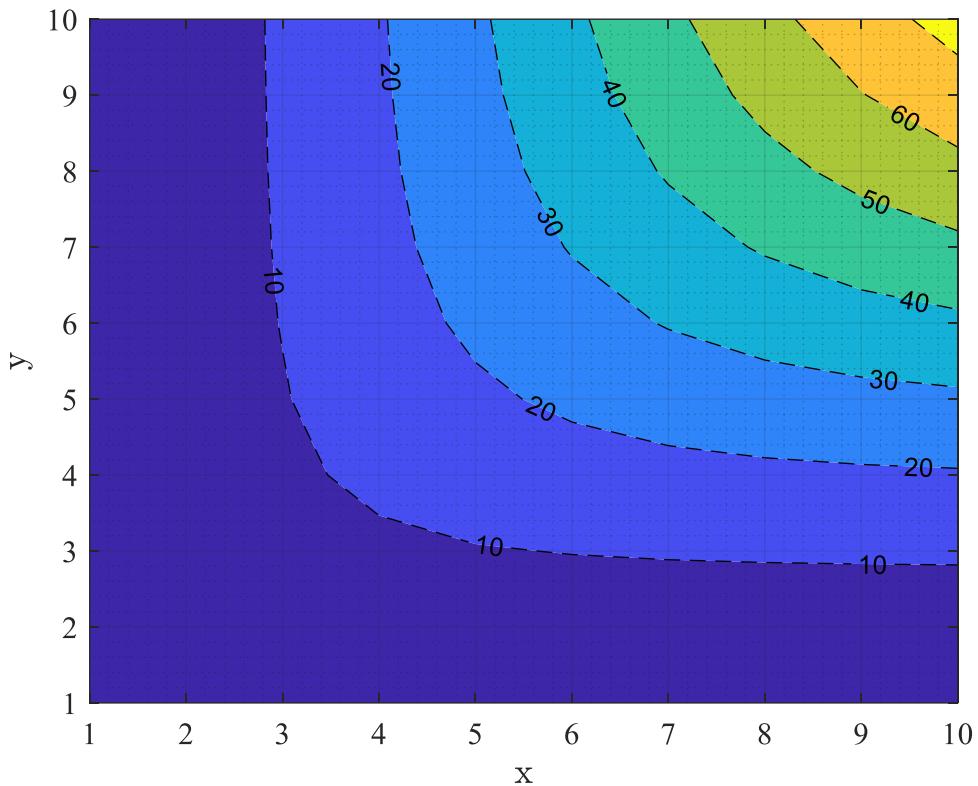
در تحلیل بعدی، جدول ۴ - ۱ نشان داده شده است، افزایش مساحت مقطع باعث افزایش خیز غشاء می شود. برای نمایش بهتر این موضوع شکل ۴ - ۲۳ و شکل ۴ - ۲۴ رسم شده است.

جدول ۴ - ۱ تغییرات خیز با تغییرات طولی و عرضی مقطع مستطیلی.

	<i>x</i>									
	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>8</b>	<b>9</b>	<b>10</b>
<b>1</b>	0.7367	1.1387	1.2268	1.2452	1.249	1.2498	1.25	1.25	1.25	1.25
<b>2</b>	1.1387	2.9468	4.0309	4.5549	4.7967	4.9073	4.9577	4.9807	4.9912	4.996
<b>3</b>	1.2268	4.0309	6.6304	8.4348	9.5653	10.2485	10.656	10.8979	11.0414	11.1264
<b>4</b>	1.2452	4.5549	8.4348	11.7874	14.3216	16.1234	17.3692	18.2195	18.7965	19.187
<b>5</b>	1.249	4.7967	9.5653	14.3216	18.4178	21.6852	24.1867	26.0604	27.4475	28.468
<b>6</b>	1.2498	4.9073	10.2485	16.1234	21.6852	26.5217	30.5237	33.739	36.2777	38.2614
<b>7</b>	1.25	4.9577	10.656	17.3692	24.1867	30.5237	36.099	40.8364	44.7734	47.9992
<b>8</b>	1.25	4.9807	10.8979	18.2195	26.0604	33.739	40.8364	47.1497	52.6229	57.2862
<b>9</b>	1.25	4.9912	11.0414	18.7965	27.4475	36.2777	44.7734	52.6229	59.6738	65.8832
<b>10</b>	1.25	4.996	11.1264	19.187	28.468	38.2614	47.9992	57.2862	65.8832	73.6714

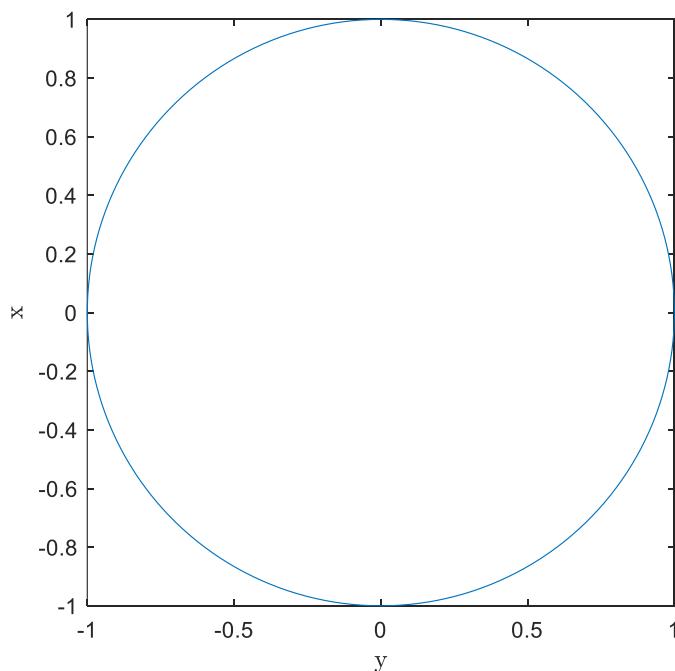


شکل ۴ - ۲۳ شماتیک تغییرات خیز غشاء مستطیلی با تغییرات طولی و عرضی مقطع مستطیلی.

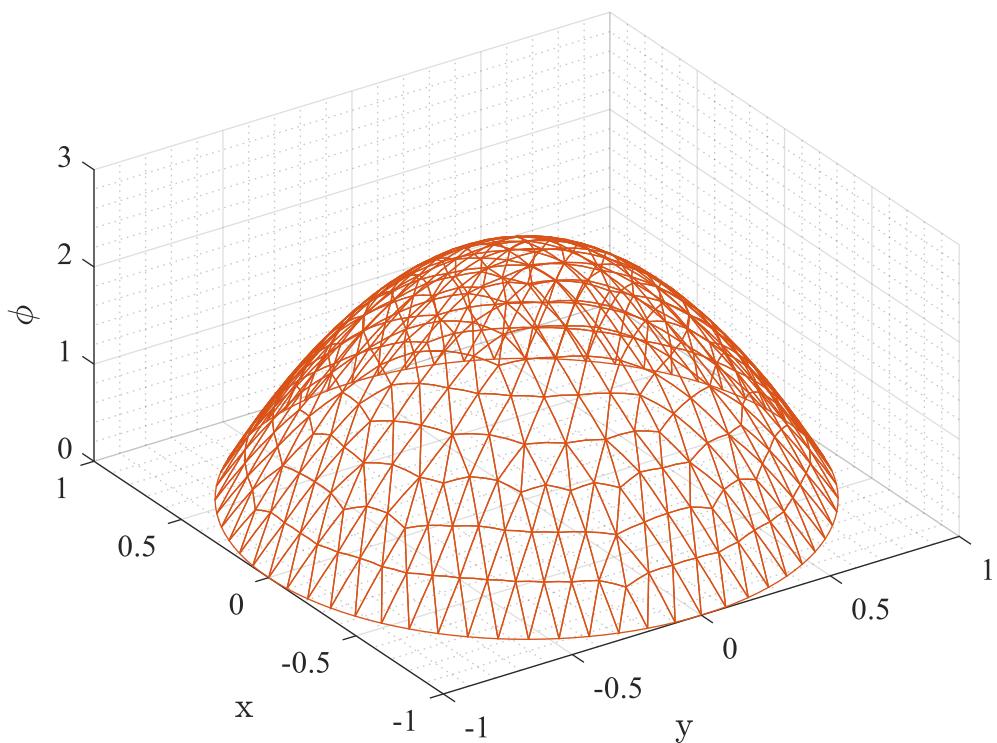


شکل ۴ - ۲۴ شماتیک کانتور تغییرات خیز غشاء مستطیلی با تغییرات طولی و عرضی مقطع مستطیلی.

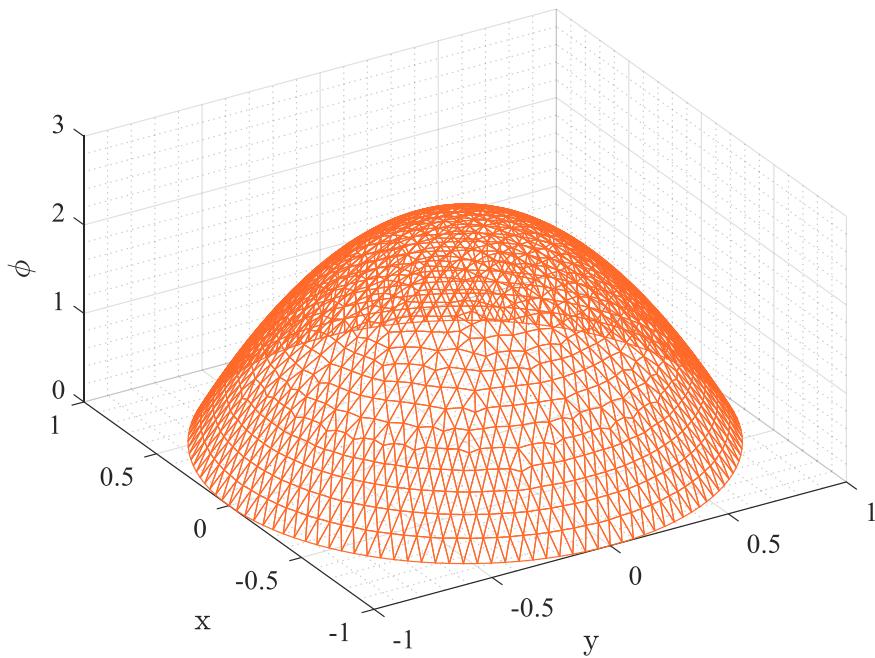
در این قسمت با در نظر گرفتن سطح مقطع دایروی، و المان مثلثی با سه نود (موجود در کتابخانه کد ارائه شده)، به بررسی خیز یک غشاء دایروی در برابر فشار روی غشاء ۱۰ و کشش ۱ می‌باشد. ساختار در نظر گرفته شده در شکل ۴-۲۵ قابل مشاهده می‌باشد. با در نظر گرفتن ۶۴ المان در دور غشاء دایروی، شکل ۴-۲۶ قابل حصول است. با افزایش تعداد المان‌ها دقیق محاسبات بیشتر خواهد شد و شکل ۴-۲۷ تا شکل ۴-۲۸ Smooth تری خواهیم داشت.



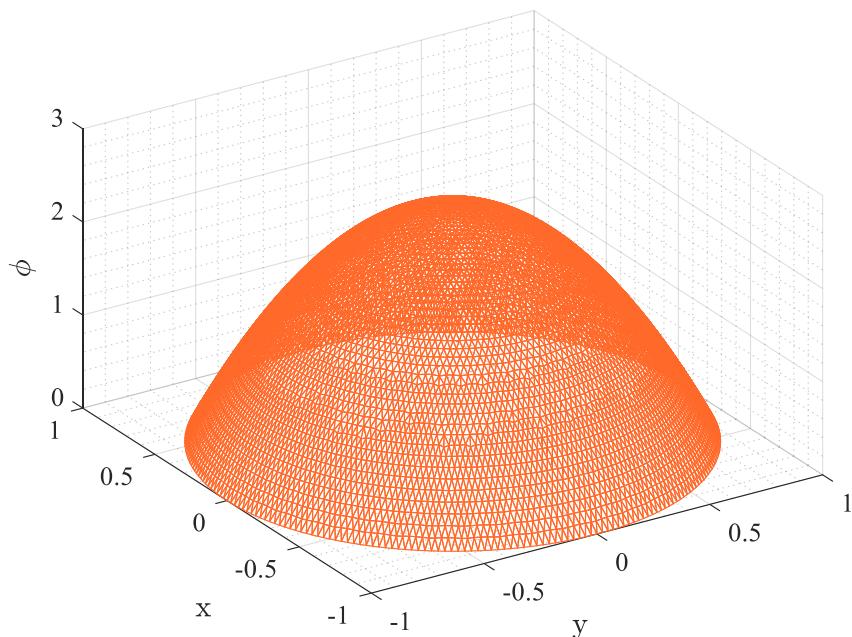
شکل ۴ - ۲۵ شماتیک غشاء دایره‌ای با شعاع ۱.



شکل ۴ - ۲۶ نمای خیز غشاء دایره‌ای با المان مش مثلثی سه گره برای لایه اول خارجی با تعداد المان ۶۴.



شکل ۴ - ۲۷ نمای خیز غشاء دایره‌ای با المان مش مثلثی سه گره برای لایه اول خارجی با تعداد المان ۱۲۸.



شکل ۴ - ۲۸ نمای خیز غشاء دایره‌ای با المان مش مثلثی سه گره برای لایه اول خارجی با تعداد المان ۶۵۶.

## فصل پنجم

### جمع‌بندی و نتیجه‌گیری

## جمع‌بندی

همانطور که ملاحظه شد، در پروژه اول درس المان محدود، به بررسی خیز غشاء به روش المان محدود پرداخته شد. در ابتدا، کلیات غشاء و معادلات حاکم بر مسئله و همچنین شرح توضیحاتی از روش حل و الگوریتم داده شده است. سپس، در فصل چهارم، نتایج حاصل در غالب نمودار و جدول شرح داده شده است، در ابتدا خیز غشاء مستطیلی با المان‌های مثلثی ۳ گره، سپس المان مثلثی ۶ گره محاسبه شد و در انتها نتایج برای یک مسئله غشاء دایروی بسط داده شده است. از دستاوردهای این پروژه می‌توان به این اشاره کرد که استفاده از المان با تعداد گره‌های بیشتر، می‌توان به کاهش تعداد نقاط المان برای همگرایی در جواب نهایی (خیز در مسئله حاضر) اشاره کرد.

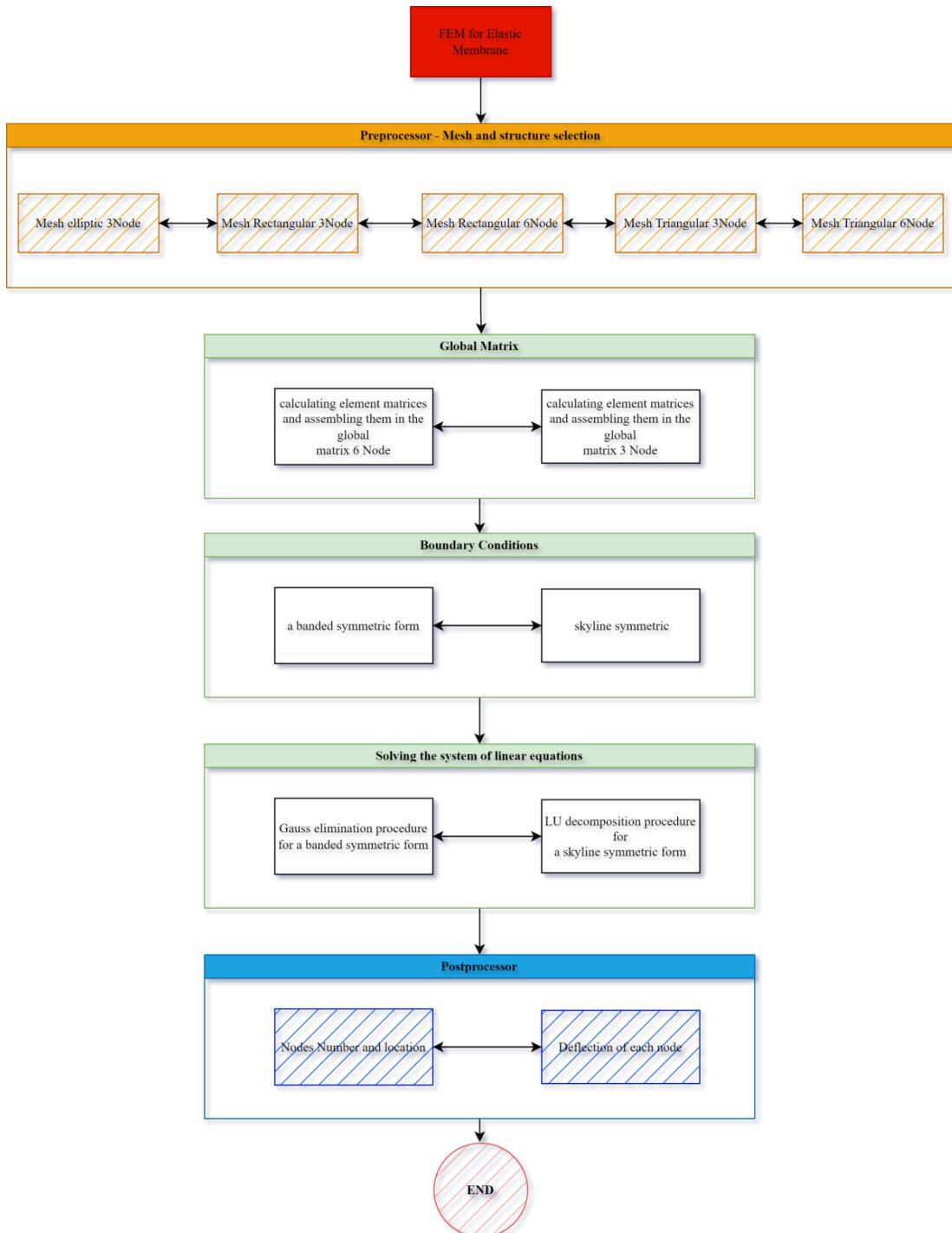
## منابع و مراجع

## منابع

- [١]F. Shaw, N. Perrone, A numerical solution for the nonlinear deflection of membranes.(١٩٥٤)
- [٢]R. Kao, N. Perrone, Large deflections of flat arbitrary membranes, Computers & Structures 2(4) (1972) 535-546.
- [٣]R. Kao, N. Perrone, Large deflections of axisymmetric circular membranes, International Journal of Solids and Structures 7(12) (1971) 1601-1612.
- [٤]P. Seide, Large deflections of rectangular membranes under uniform pressure, International Journal of Non-Linear Mechanics 12(6) (1977) 397-406.
- [٥]D. Maier-Schneider, J. Maibach, E. Obermeier, A new analytical solution for the load-deflection of square membranes, Journal of Microelectromechanical Systems 4(4) (1995) 238-241.
- [٦]G.C. Tsiatas, J.T. Katsikadelis, Large deflection analysis of elastic space membranes, International Journal for Numerical Methods in Engineering 65(2) (2006) 264-294.
- [٧]W. Fichter, Some solutions for the large deflections of uniformly loaded circular membranes, 1997.
- [٨]T.-C. Lim, Large Deflection of Circular Auxetic Membranes Under Uniform Load, Journal of Engineering Materials and Technology 138(4).(٢٠١١)
- [٩]P. Huang, Y. Song, Q. Li, X. Liu, Y. Feng, A Theoretical Study of Circular Orthotropic Membrane Under Concentrated Load: The Relation of Load and Deflection ,IEEE Access 8 (2020) 126127-126137.
- [١٠]M.R. Eslami, Finite elements methods in mechanics, Springer2014.
- [١١]A. Zamani, M.R. Eslami, Implementation of the extended finite element method for dynamic thermoelastic fracture initiation, International Journal of Solids and Structures 47(10) (2010) 1392-1404.
- [١٢]A. Zamani, R. Gracie, M.R. Eslami, Higher order tip enrichment of extended finite element method in thermoelasticity, Computational Mechanics 46 (2010) 851-866.
- [١٣]O.O. Oyekoya, D.U. Mba, A.M. El-Zafrany, Buckling and vibration analysis of functionally graded composite structures using the finite element method, Composite Structures 89(1) (2009) 134-142.
- [١٤]G. Dhatt, G. Touzot, The finite element method displayed.(١٩٨٤)

## پیوست‌ها

در این قسمت کد استفاده شده برای حل غشاء دایره‌ای به اندازه یک در نظر گرفته شده است.



شکل پ - ۱ فلوچارت اجرای کد المان محدود C++ برای غشاء الاستیک.

## جدول پ - ۱ کد C++ مسئله المان محدود شعاع دایره‌ای.

```

/*
   Introducing global variables
*
* nx=number of divisions in x direction
* ny=number of divisions in y direction
* nr=number of divisions in radial direction for an elliptic domain
* nc=number of divisions in angular direction for the first layer of
*     elements around the ellipse center.
* npe=number of nodes per element
* nb=bandwidth
* ne=number of elements
* nn=number of nodes
* node=connectivity
* xv=vector of x values
* yv=vector of y values
* iconv=convection vector
* const1=determines whether to impose primary boundary conditions
* const2=determines the value of primary boundary conditions
* maxh=height of each column for skyline implementation
* gk=global stiffness matrix
* p=the vector which is first used to store external forces and then
*     to store the calculated primary variables from finite element model
* Q=heat generation inside element (or pressure in membrane formulation).
* q=heat flux to the boundary edges
* h=the coefficient of convection heat transfer
* Tinf=temperature at infinity (surrounding media).
* kx,ky=thermal conductivities in x and y directions respectively
*         (or tension in membrane formulation).
*
***** */
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <conio.h>
#include <iostream>
#include <fstream>
#include <float.h>

int Xd=1;
int Yd=1;
using namespace std;

const long nx=30, ny=48, nr=20, nc=64;

//the variables for a rectangular domain and three node triangular element
//(rd_3nte).
//const long npe=3, nb=nx+2, ne=2*nx*ny, nn=(nx+1)*(ny+1);

//the variables for a rectangular domain and six node triangular element
//(rd_6nte).
//const long npe=6, nb=4*nx+3, ne=2*nx*ny, nn=(2*nx+1)*(2*ny+1);

//the variables for a triangular domain and three node triangular element
//(td_3nte).
//const long npe=3, nb=ny+2, ne=ny*ny, nn=(ny+1)*(ny+2)/2;

//the variables for a triangular domain and six node triangular element
//(td_6nte).
//const long npe=6, nb=4*ny+2, ne=ny*ny, nn=(2*ny+1)*(ny+1);

//the variables for an elliptic domain and three node triangular element
//(ed_3nte).
const long npe=3, nb=nc*(2*nr-3), ne=nc*(nr-1)*(nr-1), nn=nc*nr*(nr-1)/2+1;

```

```

long      node[npe*ne+1],  iconv[nn+1], const1[nn+1],maxh[nn+2];
double    gk[nn*nb+1], p[nn+1], xv[nn+1], yv[nn+1], const2[nn+1];
double    Q[ne+1], h[nn+1], q[nn+1], Tinf[nn+1];
double    kx,ky;

void mesh_rd_3nte(double x0,double y0);
void mesh_rd_6nte(double x0,double y0);
void mesh_td_3nte(double x0,double y0);
void mesh_td_6nte(double x0,double y0);
void mesh_ed_3nte(double x0,double y0);
void bandwidth();
void core_3nte(char type);
void core_6nte(char type);
void boundary_sym_banded(long neqns,long nbw,double mat[],double rhs[],
    long cond1[],double cond2[]);
void boundary_sym_skyline(long neqns,long nbw,double mat[],double rhs[],
    long maxh[],long cond1[],double cond2[]);
void solve_gauss_sym_banded(long neqns,long nbw,double mat[],double rhs[]);
void solve_LUdecom_sym_skyline(long neqns,double mat[],double rhs[],long maxh[]);
void file_out();

int main()
{
    char type='b'; //specifies whether to choose the banded or skyline form.
                    //the value 'b' refers to banded form.
                    //the value 's' refers to skyline form.

    mesh_ed_3nte(Xd,Yd);

    switch(type){
        case 'b' :
            core_3nte(type);
            boundary_sym_banded(nn,nb,gk,p,const1,const2);
            solve_gauss_sym_banded(nn,nb,gk,p);
            break;
        case 's' :
            bandwidth();
            core_3nte(type);
            boundary_sym_skyline(nn,nb,gk,p,maxh,const1,const2);
            solve_LUdecom_sym_skyline(nn,gk,p,maxh);
            break;
    }

    file_out();
}

void mesh_rd_3nte(double x0,double y0)//generates the mesh for a rectangular
//domain by three node triangular elements
{
    long i,j,n,m,n1,n2,nx1,ny1;

    nx1=nx+1;
    ny1=ny+1;
    m=-6;
    for (j=1;j<=ny;j++){
        for (i=1;i<=nx;i++){
            m+=6;
            n=nx1*(j-1)+i;

            node[m+1]=n;
            node[m+2]=n+1;
            node[m+3]=n+nx+1;

            node[m+4]=n+1;
            node[m+5]=n+nx+2;
            node[m+6]=n+nx+1;
        }
    }
}

```

```

        }

    }

    for (i=1;i<=nx1;i++){
        for (j=1;j<=ny1;j++){
            n=nx1*(j-1)+i;
            xv[n]=double(i-1)*x0/double(nx);
            yv[n]=double(j-1)*y0/double(ny);
        }
    }

    for (i=1;i<=nn;i++){
        icanv[i]=0;
        const1[i]=0;
        const2[i]=0.0;
    }

    n=nx1*ny;
    for (i=1;i<=nx1;i++){
        n1=i;
        n2=i+n;
        const1[n1]=1;
        const1[n2]=1;

        const2[n1]=0.0;
        const2[n2]=0.0;
    }

    for (j=2;j<=ny;j++){
        n1=nx1*(j-1)+1;
        n2=nx1*j;
        const1[n1]=1;
        const1[n2]=1;

        const2[n1]=0.0;
        const2[n2]=0.0;
    }

    for(i=1;i<=ne;i++){
        Q[i]=10.0;
    }

    kx=1.0;ky=1.0;
}

void mesh_rd_6nte(double x0,double y0)//generates the mesh for a rectangular
//domain by six node triangular elements
{
    long i,j,n,m,n1,n2,nx1,ny1,ny2;

    nx1=2*nx+1;
    ny1=2*ny+1;
    m=-12;
    for (j=1;j<=ny;j++){
        for (i=1;i<=nx;i++){
            m+=12;
            n=2*nx1*(j-1)+2*i-1;

            node[m+1]=n;
            node[m+2]=n+2;
            node[m+3]=n+4*nx+2;
            node[m+4]=n+1;
            node[m+5]=n+2*nx+2;
            node[m+6]=n+2*nx+1;

            node[m+7]=n+2;
            node[m+8]=n+4*nx+4;
        }
    }
}

```

```

        node[m+9]=n+4*nx+2;
        node[m+10]=n+2*nx+3;
        node[m+11]=n+4*nx+3;
        node[m+12]=n+2*nx+2;
    }

    for (i=1;i<=nx1;i++){
        for (j=1;j<=ny1;j++){
            n=nx1*(j-1)+i;
            xv[n]=double(i-1)*x0/double(2*nx);
            yv[n]=double(j-1)*y0/double(2*ny);
        }
    }

    for (i=1;i<=nn;i++){
        iconv[i]=0;
        const1[i]=0;
        const2[i]=0.0;
    }

    n=2*nx1*ny;
    for (i=1;i<=nx1;i++){
        n1=i;
        n2=i+n;
        const1[n1]=1;
        const1[n2]=1;

        const2[n1]=0.0;
        const2[n2]=0.0;
    }

    ny2=ny1-1;
    for (j=2;j<=ny2;j++){
        n1=nx1*(j-1)+1;
        n2=nx1*j;
        const1[n1]=1;
        const1[n2]=1;

        const2[n1]=0.0;
        const2[n2]=0.0;
    }

    for(i=1;i<=ne;i++){
        Q[i]=10.0;
    }

    kx=1.0;ky=1.0;
}

void mesh_td_3nte(double x0,double y0)//generates the mesh for a triangular
//domain by three node triangular elements
{
    long i,j,n,m,n1,n2,n3,ny1;

    ny1=ny+1;
    for (j=1;j<=ny;j++){
        for (i=1;i<=j-1;i++){
            m=3*((j-1)*(j-1)+2*(i-1));
            n=(j-1)*j/2+i;

            node[m+1]=n;
            node[m+2]=n+j+1;
            node[m+3]=n+j;

            node[m+4]=n;
            node[m+5]=n+1;
        }
    }
}

```

```

        node[m+6]=n+j+1;
    }
}

for(j=1;j<=ny;j++){
    n=j*(j+1)/2;
    m=j*j;

    node[3*(m-1)+1]=n;
    node[3*(m-1)+2]=n+j+1;
    node[3*(m-1)+3]=n+j;
}

for (j=1;j<=ny1;j++){
    for (i=1;i<=j;i++){
        n=(j-1)*j/2+i;
        xv[n]=double(i-1)*x0/double(ny);
        yv[n]=double(j-1)*y0/double(ny);
    }
}

for (i=1;i<=nn;i++){
    iconv[i]=0;
    const1[i]=0;
    const2[i]=0.0;
}

n=ny*ny1/2;
for (j=1;j<=ny1;j++){
    n1=j*(j-1)/2+1;
    n2=j+n;
    n3=j*(j+1)/2;

    const1[n1]=1;
    const1[n2]=1;
    const1[n3]=1;

    const2[n1]=0.0;
    const2[n2]=0.0;
    const2[n3]=0.0;
}

for(i=1;i<=ne;i++){
    Q[i]=10.0;
}

kx=1.0;ky=1.0;
}

void mesh_td_6nte(double x0,double y0)//generates the mesh for a triangular
//domain by six node triangular elements
{
    long i,j,n,m,n1,n2,n3,ny1;

    ny1=2*ny+1;
    for (j=1;j<=ny;j++){
        for (i=1;i<=j-1;i++){
            m=6*((j-1)*(j-1)+2*(i-1));
            n=(j-1)*(2*j-1)+2*i-1;

            node[m+1]=n;
            node[m+2]=n+4*j+1;
            node[m+3]=n+4*j-1;
            node[m+4]=n+2*j;
            node[m+5]=n+4*j;
            node[m+6]=n+2*j-1;
        }
    }
}

```

```

        node[m+7]=n;
        node[m+8]=n+2;
        node[m+9]=n+4*j+1;
        node[m+10]=n+1;
        node[m+11]=n+2*j+1;
        node[m+12]=n+2*j;
    }

}

for(j=1;j<=ny;j++){
    n=(2*j-1)*j;
    m=j*j;

    node[6*(m-1)+1]=n;
    node[6*(m-1)+2]=n+4*j+1;
    node[6*(m-1)+3]=n+4*j-1;
    node[6*(m-1)+4]=n+2*j;
    node[6*(m-1)+5]=n+4*j;
    node[6*(m-1)+6]=n+2*j-1;
}

for (j=1;j<=ny1;j++){
    for (i=1;i<=j;i++){
        n=(j-1)*j/2+i;
        xv[n]=double(i-1)*x0/double(2*ny);
        yv[n]=double(j-1)*y0/double(2*ny);
    }
}

for (i=1;i<=nn;i++){
    iconv[i]=0;
    const1[i]=0;
    const2[i]=0.0;
}

n=ny*ny1;
for (j=1;j<=ny1;j++){
    n1=j*(j-1)/2+1;
    n2=j+n;
    n3=j*(j+1)/2;

    const1[n1]=1;
    const1[n2]=1;
    const1[n3]=1;

    const2[n1]=0.0;
    const2[n2]=0.0;
    const2[n3]=0.0;
}

for(i=1;i<=ne;i++){
    Q[i]=10.0;
}

kx=1.0;ky=1.0;
}

void mesh_ed_3nte(double x0,double y0)//generates the mesh for an elliptic
//domain by three node triangular elements
{
    long i,j,m,n,i1,j1,k1,m1,n1,nr1,ntheta;
    double pi=3.141592653589793;
    double sc,a,b,r,theta;

    nr1=nr-1;
}

```

```

m=-3;
n=0;
i1=1;
for(i=1;i<=nr1;i++){
    ntheta=nc*i;
    for(j=1;j<=ntheta;j++){
        m+=3;
        n++;

        if(j==1) node[m+1]=i1;
        else if(div(j-1,i).rem==0) node[m+1]=node[m-2];
        else node[m+1]=node[m-2]+1;

        node[m+2]=n+1;
        node[m+3]=n+2;
        if(j==1){
            i1=node[m+1];
            k1=node[m+2];
        }
    }
    node[m+1]=i1;
    node[m+3]=k1;
    i1=k1;
}

m1=m;
n1=n;
j1=nc+3;
for(i=2;i<=nr1;i++){
    ntheta=nc*(i-1);
    for(j=1;j<=ntheta;j++){
        m+=3;
        n++;

        if(j==1) node[m+2]=j1;
        else if(div(j-1,i-1).rem==0) node[m+2]=node[m-1]+2;
        else node[m+2]=node[m-1]+1;

        node[m+1]=n-n1+1;
        node[m+3]=node[m+1]+1;
        if(j==1) k1=node[m+1];
    }
    node[m+3]=k1;
    j1=node[m+2]+2;
}

xv[1]=0;
yv[1]=0;
i1=1;
for(i=1;i<=nr1;i++){
    sc=double(i)/double(nr1);
    ntheta=nc*i;
    for(j=1;j<=ntheta;j++){
        i1++;
        theta=2.0*pi*double(j-1)/double(ntheta);
        a=sin(theta);
        b=cos(theta);
        r=sc*x0*y0/sqrt(x0*x0*a*a+y0*y0*b*b);
        xv[i1]=r*b;
        yv[i1]=r*a;
    }
}

for (i=1;i<=nn;i++){
    iconv[i]=0;
}

```

```

        const1[i]=0;
        const2[i]=0.0;
    }

    i1=nc*(nr-1);
    n=nc*(nr-1)*(nr-2)/2+1;
    for(i=1;i<=i1;i++){
        n++;
        const1[n]=1;
        const2[n]=0.0;
    }

    for(i=1;i<=ne;i++){
        Q[i]=10.0;
    }

    kx=1.0;ky=1.0;
}

void bandwidth()//calculates the height of each individual column for
                //skyline format
{
    long i,j,num,n1,a;
    long n[npe+1];

    n1=nn+1;
    for(i=1;i<=n1;i++){
        maxh[i]=0;
    }

    for(num=1;num<=ne;num++){

        i=npe*(num-1);
        for(j=1;j<=npe;j++){
            n[j]=node[i+j];
        }

        for(i=1;i<=npe;i++){
            for(j=1;j<=npe;j++){
                if(n[i]>n[j]){
                    n1=n[i]+1;
                    a=n[i]-n[j]+1;
                }
                else{
                    n1=n[j]+1;
                    a=n[j]-n[i]+1;
                }
                if(maxh[n1]<a) maxh[n1]=a;
            }
        }
    }

    maxh[1]=1;
    maxh[2]=2;
    n1=nn+1;
    for(i=3;i<=n1;i++){
        maxh[i]+=maxh[i-1];
    }
}

void core_3nte(char type)//calculates element matrices and assembles them in
                        //the global matrix for three node triangular elements
{
    long      i,j,ii,num;
    double    area,s,Tinfm,hm,qm,conv;
}

```

```

long      n[npe+1];
double    b[npe+1],c[npe+1],x[npe+1],y[npe+1];
double    k1[npe+1][npe+1],k2[npe+1][npe+1],p1[npe+1],p2[npe+1],p3[npe+1];

for (num=1;num<=ne;num++){
    for (i=1;i<=npe;i++){
        p1[i]=0.0;
        p2[i]=0.0;
        p3[i]=0.0;
        for(j=1;j<=npe;j++){
            k1[i][j]=0.0;
            k2[i][j]=0.0;
        }
    }

    i=npe*(num-1);
    for(j=1;j<=npe;j++){
        n[j]=node[i+j];
        x[j]=xv[n[j]];
        y[j]=yv[n[j]];
    }

    b[1]=y[3]-y[2];      b[2]=y[1]-y[3];      b[3]=y[2]-y[1];
    c[1]=x[2]-x[3];      c[2]=x[3]-x[1];      c[3]=x[1]-x[2];

    area=fabs( ((x[1]-x[3])*(y[2]-y[3])- (x[2]-x[3])*(y[1]-y[3])) /2.0);

    for(i=1;i<=npe;i++){
        for(j=i;j<=npe;j++){
            k1[i][j]=(kx*b[i]*b[j]+ky*c[i]*c[j])/(4.0*area);
        }
    }

    for (i=1;i<=npe;i++){
        p1[i]=Q[num]*area/3.0;
    }

    if ((iconv[n[1]]+iconv[n[2]])==2){
        s=sqrt((x[1]-x[2))*(x[1]-x[2])+(y[1]-y[2))*(y[1]-y[2]));
        Tinfm=(Tinf[n[1]]+Tinf[n[2]])/2;
        hm=(h[n[1]]+h[n[2]])/2;
        qm=(q[n[1]]+q[n[2]])/2;
        conv=hm*s/6.0;

        k2[1][1]=2.0*conv;
        k2[1][2]=conv;
        k2[2][2]=k2[1][1];

        p2[1]=hm*Tinfm*s/2.0;
        p2[2]=p2[1];

        p3[1]=qm*s/2.0;
        p3[2]=p3[1];
    }

    if ((iconv[n[2]]+iconv[n[3]])==2){
        s=sqrt((x[2]-x[3))*(x[2]-x[3])+(y[2]-y[3))*(y[2]-y[3]));
        Tinfm=(Tinf[n[2]]+Tinf[n[3]])/2;
        hm=(h[n[2]]+h[n[3]])/2;
        qm=(q[n[2]]+q[n[3]])/2;
        conv=hm*s/6.0;

        k2[2][2]=2.0*conv;
        k2[2][3]=conv;
        k2[3][3]=k2[2][2];
    }
}

```

```

p2[2]=hm*Tinfm*s/2.0;
p2[3]=p2[2];

p3[2]=qm*s/2.0;
p3[3]=p3[2];
}

if ((iconv[n[3]]+iconv[n[1]])==2){
    s=sqrt( (x[3]-x[1])*(x[3]-x[1])+(y[3]-y[1])*(y[3]-y[1]) );
    Tinfm=(Tinfm[n[3]]+Tinfm[n[1]])/2;
    hm=(h[n[3]]+h[n[1]])/2;
    qm=(q[n[3]]+q[n[1]])/2;
    conv=hm*s/6.0;

    k2[1][1]=2.0*conv;
    k2[1][3]=conv;
    k2[3][3]=k2[1][1];

    p2[1]=hm*Tinfm*s/2.0;
    p2[3]=p2[1];

    p3[1]=qm*s/2.0;
    p3[3]=p3[1];
}

switch(type){
    case 'b' :
        for(i=1;i<=npe;i++){
            p[n[i]]+=p1[i]+p2[i]+p3[i];
            for(j=i;j<=npe;j++){
                if(n[i]>n[j]) ii=(n[j]-1)*nb+n[i]-n[j]+1;
                else ii=(n[i]-1)*nb+n[j]-n[i]+1;
                gk[ii]+=k1[i][j]+k2[i][j];
            }
        }
        break;
    case 's' :
        for(i=1;i<=npe;i++){
            p[n[i]]+=p1[i]+p2[i]+p3[i];
            for(j=i;j<=npe;j++){
                if(n[i]>n[j]) ii=maxh[n[i]]+n[i]-n[j];
                else ii=maxh[n[j]]+n[j]-n[i];
                gk[ii]+=k1[i][j]+k2[i][j];
            }
        }
        break;
    }
}

void core_6nte(char type)//calculates element matrices and assembles them in
//the global matrix for six node triangular elements
{
    long      i,j,ii,num;
    double    J11,J12,J21,J22,area,a11,a12,a22,s,Tinfm,hm,qm,conv;
    long      n[npe+1];
    double   x[npe+1],y[npe+1],k1[npe+1][npe+1],k2[npe+1][npe+1];
    double   p1[npe+1],p2[npe+1],p3[npe+1];

    for (num=1;num<=ne;num++){

        for (i=1;i<=npe;i++){
            p1[i]=0.0;
            p2[i]=0.0;
            p3[i]=0.0;
            for(j=1;j<=npe;j++){
}

```

```

        k1[i][j]=0.0;
        k2[i][j]=0.0;
    }
}

i=npe*(num-1);
for(j=1;j<=npe;j++){
    n[j]=node[i+j];
    x[j]=xv[n[j]];
    y[j]=yv[n[j]];
}

J11=x[1]-x[3];      J12=y[1]-y[3];
J21=x[2]-x[3];      J22=y[2]-y[3];

area=fabs( (J11*J22-J12*J21) /2.0);

a11=(kx*J22*J22+ky*J21*J21)/2.0/area;
a12=-(kx*J22*J12+ky*J21*J11)/2.0/area;
a22=(kx*J12*J12+ky*J11*J11)/2.0/area;

k1[1][1]=0.5*a11;
k1[1][2]=-a12/6.0;
k1[1][3]=(a11+a12)/6.0;
k1[1][4]=2.0/3.0*a12;
k1[1][5]=0.0;
k1[1][6]=-2.0/3.0*(a11+a12);
k1[2][2]=0.5*a22;
k1[2][3]=(a12+a22)/6.0;
k1[2][4]=2.0/3.0*a12;
k1[2][5]=-2.0/3.0*(a12+a22);
k1[2][6]=0.0;
k1[3][3]=0.5*(a11+2*a12+a22);
k1[3][4]=0.0;
k1[3][5]=-2.0/3.0*(a12+a22);
k1[3][6]=-2.0/3.0*(a11+a12);
k1[4][4]=4.0/3.0*(a11+a12+a22);
k1[4][5]=-4.0/3.0*(a11+a12);
k1[4][6]=-4.0/3.0*(a12+a22);
k1[5][5]=4.0/3.0*(a11+a12+a22);
k1[5][6]=4.0/3.0*a12;
k1[6][6]=4.0/3.0*(a11+a12+a22);

for (i=4;i<=npe;i++){
    p1[i]=Q[num]*area/3.0;
}

if ((iconv[n[1]]+iconv[n[2]])==2){
    s=sqrt( (x[1]-x[2])*(x[1]-x[2])+(y[1]-y[2])*(y[1]-y[2]) );
    Tinfm=(Tinf[n[1]]+Tinf[n[4]]+Tinf[n[2]])/2;
    hm=(h[n[1]]+h[n[4]]+h[n[2]])/2;
    qm=(q[n[1]]+q[n[4]]+q[n[2]])/2;
    conv=hm*s/30.0;

    k2[1][1]=4.0*conv;
    k2[1][2]=-conv;
    k2[1][4]=2.0*conv;
    k2[2][2]=k2[1][1];
    k2[2][4]=k2[1][4];
    k2[4][4]=16.0*conv;

    p2[1]=hm*Tinfm*s/6.0;
    p2[2]=p2[1];
    p2[4]=2.0*p2[1];

    p3[4]=qm*s/3.0;
}

```

```

if ((iconv[n[2]]+iconv[n[3]])==2){
    s=sqrt( (x[2]-x[3])*(x[2]-x[3])+(y[2]-y[3])*(y[2]-y[3]) );
    Tinfm=(Tinf[n[2]]+Tinf[n[5]]+Tinf[n[3]])/2;
    hm=(h[n[2]]+h[n[5]]+h[n[3]])/2;
    qm=(q[n[2]]+q[n[5]]+q[n[3]])/2;
    conv=hm*s/30.0;

    k2[2][2]=4.0*conv;
    k2[2][3]=-conv;
    k2[2][5]=2.0*conv;
    k2[3][3]=k2[2][2];
    k2[3][5]=k2[2][5];
    k2[5][5]=16.0*conv;

    p2[2]=hm*Tinfm*s/6.0;
    p2[3]=p2[2];
    p2[5]=2.0*p2[2];

    p3[5]=qm*s/3.0;
}

if ((iconv[n[3]]+iconv[n[1]])==2){
    s=sqrt( (x[3]-x[1])*(x[3]-x[1])+(y[3]-y[1])*(y[3]-y[1]) );
    Tinfm=(Tinf[n[3]]+Tinf[n[6]]+Tinf[n[1]])/2;
    hm=(h[n[3]]+h[n[6]]+h[n[1]])/2;
    qm=(q[n[3]]+q[n[6]]+q[n[1]])/2;
    conv=hm*s/30.0;

    k2[1][1]=4.0*conv;
    k2[1][3]=-conv;
    k2[1][6]=2.0*conv;
    k2[3][3]=k2[1][1];
    k2[3][6]=k2[1][6];
    k2[6][6]=16.0*conv;

    p2[1]=hm*Tinfm*s/6.0;
    p2[3]=p2[1];
    p2[6]=2.0*p2[1];

    p3[6]=qm*s/3.0;
}

switch(type){
    case 'b' :
        for(i=1;i<=npe;i++){
            p[n[i]]+=p1[i]+p2[i]+p3[i];
            for(j=i;j<=npe;j++){
                if(n[i]>n[j]) ii=(n[j]-1)*nb+n[i]-n[j]+1;
                else ii=(n[i]-1)*nb+n[j]-n[i]+1;
                gk[ii]+=k1[i][j]+k2[i][j];
            }
        }
        break;
    case 's' :
        for(i=1;i<=npe;i++){
            p[n[i]]+=p1[i]+p2[i]+p3[i];
            for(j=i;j<=npe;j++){
                if(n[i]>n[j]) ii=maxh[n[i]]+n[i]-n[j];
                else ii=maxh[n[j]]+n[j]-n[i];
                gk[ii]+=k1[i][j]+k2[i][j];
            }
        }
        break;
}
}

```

```

void boundary_sym_banded(long neqns,long nbw,double mat[],double rhs[],
    long cond1[],double cond2[])
//imposes the boundary conditions on primary variables for
//a banded symmetric form
{
    long ii,i1,i2,i3,j;
    for (ii=1;ii<=neqns;ii++){
        if (cond1[ii]!=0){
            i1=ii;
            i2=ii;
            i3=(ii-1)*nbw;
            for (j=2;j<=nbw;j++){
                i1--;
                i2++;
                if(i1>=1){
                    rhs[i1]-=mat[(i1-1)*nbw+j]*cond2[ii];
                    mat[(i1-1)*nbw+j]=0.0;
                }
                if(i2<=neqns){
                    rhs[i2]-=mat[i3+j]*cond2[ii];
                    mat[i3+j]=0.0;
                }
            }
            rhs[ii]=cond2[ii];
            mat[i3+1]=1.0;
        }
    }
}

void boundary_sym_skyline(long neqns,long nbw,double mat[],double rhs[],
    long maxh[],long cond1[],double cond2[])
//imposes the boundary conditions on primary variables for
//a skyline symmetric form
{
    long ii,i1,i2,j,j1,nbw1;

    nbw1=nbw-1;
    for (ii=1;ii<=neqns;ii++){
        if (cond1[ii]!=0){
            i1=ii;
            i2=maxh[ii];
            j1=maxh[ii+1]-maxh[ii]-1;
            for(j=1;j<=j1;j++){
                i1--;
                rhs[i1]-=mat[i2+j]*cond2[ii];
                mat[i2+j]=0.0;
            }
            i1=ii;
            for(j=1;j<=nbw1;j++){
                i1++;
                if(ii<neqns && j<=maxh[ii+1]-maxh[ii]-1){
                    rhs[i1]-=mat[maxh[ii]+j]*cond2[ii];
                    mat[maxh[ii]+j]=0.0;
                }
            }
            rhs[ii]=cond2[ii];
            mat[i2]=1.0;
        }
    }
}

```

```

void solve_gauss_sym_banded(long neqns,long nbw,double mat[],double rhs[])
//solves the system of equations in banded symmetric form by
//Gauss elimination method
{
    long      i,j,ii,jj,m,n,n1,n2,n3;
    double     factor;

    m=neqns-1;
    for(n=1;n<=m;n++){
        n1=n+1;
        n2=n+nbw-1;
        n3=(n-1)*nbw;
        if (n2>neqns) n2=neqns;
        for(i=n1;i<=n2;i++){
            j=i-n1;
            factor=mat[n3+j]/mat[n3+1];
            for(j=i;j<=n2;j++){
                ii=j-i+1;
                jj=j-n+1;
                mat[(i-1)*nbw+ii]-=factor*mat[n3+jj];
            }
            rhs[i]-=factor*rhs[n];
        }
    }

    // Back substitution

    for(n=neqns;n>=2;n--){
        n1=n-1;
        n2=n-nbw+1;
        rhs[n]/=mat[(n-1)*nbw+1];
        if (n2<1) n2=1;
        for(i=n1;i>=n2;i--){
            j=n-i+1;
            rhs[i]-=mat[(i-1)*nbw+j]*rhs[n];
        }
    }

    rhs[1]/=mat[1];
}

void solve_LUdecom_sym_skyline(long neqns,double mat[],double rhs[],long maxh[])
//solves the system of equations in skyline symmetric form by
//LU decomposition method
{
    long  n,m,m1,m2,m3,m4,i,i1,i2,i3,ii,j,k;
    double a,b;

    for(n=1;n<=neqns;n++){
        m=maxh[n];
        m1=m+1;
        m2=maxh[n+1]-1;
        m3=m2-m1;
        if(m3>0){
            i=n-m3;
            i1=0;
            m4=m2;
            for(j=1;j<=m3;j++){
                i1++;
                m4--;
                i2=maxh[i];
                i3=maxh[i+1]-i2-1;
                if(i3>0){
                    ii=min(i1,i3);
                    a=0.0;
                    for(k=1;k<=ii;k++){
                        mat[i][k]=0;
                    }
                }
            }
        }
    }
}

```

```

        a+=mat[i2+k]*mat[m4+k];
    }
    mat[m4]-=a;
}
i++;
}
if(m3>=0){
    i=n;
    b=0.0;
    for(ii=m1;ii<=m2;ii++){
        i--;
        i2=maxh[i];
        a=mat[ii]/mat[i2];
        b+=a*mat[ii];
        mat[ii]=a;
    }
    mat[m]-=b;
}
//Reduation of right-hand-side veator

for(n=1;n<=neqns;n++){
    m1=maxh[n]+1;
    m2=maxh[n+1]-1;
    if(m2-m1>=0){
        i=n;
        a=0.0;
        for(ii=m1;ii<=m2;ii++){
            i--;
            a+=mat[ii]*rhs[i];
        }
        rhs[n]-=a;
    }
}

//back substitution

for(n=1;n<=neqns;n++){
    i=maxh[n];
    rhs[n]/=mat[i];
}
n=neqns;
for(k=2;k<=neqns;k++){
    m1=maxh[n]+1;
    m2=maxh[n+1]-1;
    if(m2-m1>=0){
        i=n;
        for(ii=m1;ii<=m2;ii++){
            i--;
            rhs[i]-=mat[ii]*rhs[n];
        }
    }
    n--;
}
}

void file_out()
//writes the results on some files
{
    long i,n;
    n=npe*ne;

    ofstream out1;
    out1.open("out1.dat");
    for (i=1;i<=nn;i++){

```

```

        out1<<p[i]<<"  ";
    }

ofstream out2;
out2.open("out2.dat");
for (i=1;i<=n;i++){
    out2<<node[i]<<"  ";
}

ofstream out3;
out3.open("out3.dat");
for (i=1;i<nn;i++){
    out3<<xv[i]<<"  ";
}

ofstream out4;
out4.open("out4.dat");
for (i=1;i<nn;i++){
    out4<<yv[i]<<"  ";
}

ofstream out5;
out5.open("out5.dat");
for (i=1;i<nn;i++){
    out5<<xv[i]<<","<<yv[i]<<endl;
    out5<<"  ";
}
ofstream out6;
out6.open("out6.dat");
for (i=1;i<nn;i++){

    out6<<xv[i]<<"  ";
}

ofstream outnx;
outnx.open("outnx.dat");
{
    outnx<<nx+1;
}
ofstream outny;
outny.open("outny.dat");
{
    outny<<ny+1;
}
ofstream outxd;
outxd.open("outxd.dat");
{
    outxd<<Xd;
}

ofstream outyd;
outyd.open("outyd.dat");
{
    outyd<<Yd;
}
}

```