دانشگاه صنعتی شاهرود
دانشکدهٔ مهندسی مکانیک

پایان نامه برای اخذ درجهٔ کارشناسی ارشد

موضوع:

تدوین نرم افزار جهت تحلیل جریان برشی ، دو بعدی ، گذرا و غیرقابل تراکم

استاد راهنما:

جناب آقای دکتر محمد جواد مغربی

ارائه دهنده:

محمد حسین دیبائی بناب

گرایش تبدیل انرژی

مهرماه ۱۳۸۵

چکیده

با پیشرفت علم کامپیوتر و بوجود آمدن کامپیوترهایی با سرعت بالا استفاده از حلهای عددی در مسائل مختلف علوم از جمله مهندسی فراگیر شده است. حلهای عددی نه تنها هزینه روشهای آزمایشگاهی و تجربی را ندارند، بلکه در مدت زمان کمتری نیز می توانند نتایج مفید با جزئیات زیادی را بدست آورند. در این پایان نامه نیز سعی شده است تا جریانهای برشی بوسیلهٔ روش شبیه سازی مستقیم عددی حل شود. مزیت این روش نسبت به سایر روشهای عددی این است که نیاز به مدل خاصی برای مدلسازی توربولانس ندارد ولی یکی از معایب آن این است که حجم محاسبات بالا بوده و در نتیجه حل با استفاده از این روش زمان‌بر خواهد بود.

جریانهای برشی حالتهایی از جریان هستند که استفاده صنعتی زیادی دارند و همچنین در مدلسازی مسائل پیچیده نیز به کار می‌روند؛ لیکن حل دقیق این جریانها می‌تواند در موارد فوق مشکل گشا باشد.

در این تحقیق فرم چرخشی معادلات ناویر- استوکس برای جریانهای برشی با استفاده از روشهای عددی و تفاضلات محدود فشرده[1] در جهت اصلی جریان (x) و تفاضلات فشردهٔ تطبیقی در جهت عمود بر جریان حل شده‌اند. دامنهٔ حل مسئله در جهت جریان دارای طول محدود و در جهت عمود بر جریان به سمت $\pm\infty$ میل می نماید.

از نگاشت[2] یک به یک کتانژانت $y = -\beta\cot\zeta$ برای مرتبط نمودن شبکهٔ فیزیکی (y) با شبکهٔ محاسباتی (ζ) به طول واحد استفاده شده است. در مرز ورودی شرط مرزی دیریشله برای

[1] Padé compact finite difference
[2] Mapping

سرعت و در مرز خروجی این شرط با استفاده از مدل انتقالی تولید و اعمال شده است. در این مرزها علاوه بر شرط فوق ، شرط نیومن که از معادلهٔ پیوستگی بوجود می آید بر روی $\frac{\partial u}{\partial x}$ اعمال شده است. محاسبات در دامنهٔ زمان با استفاده از روش فشردهٔ رانگ کوتای مرتبهٔ سوم انجام شده است.

نتایج بدست آمده از این تحقیق نشان می دهد وقتی که نرم افزار با پارامترهای مناسب اجرا گردد، نتایج بدست آمده از دقت بسیار خوبی برخوردارند. ضمناً تحلیل جوابها در دستگاه مختصات خودمشابه، خود تشابهی ترمهای سرعت و گردابه را بخوبی نمایان می نماید.

کلمات کلیدی: روش مستقیم عددی، جریان برشی، روش تفاضلات محدود فشرده، تفاضلات محدود فشردهٔ تطبیقی ، معادلهٔ پواسون، خود تشابهی

فهرست مطالب

تقدیر و تشکر ..... ب

چکیده ..... ث

مقدمه ۱

فصل ۱- مطالعهٔ جریان برشی ۳

۱-۱-مقدمه: ۳

۱-۲- جریان های برشی آزاد ۳

۱-۳-تحلیل جریان جت ایده آل دو بعدی ۴

۱-۴- تحلیل جریان دنباله ۱۲

۱-۵- تحلیل جریان لایه اختلاطی ۱۶

فصل ۲-مقدمه ای بر روش مستقیم عددی ۱۹

۲-۱-مقدمه: ۱۹

۲-۲-کاربردهای روش مستقیم عددی ۱۹

فصل ۳-روش‌های تقسیم دامنه ۲۲

۳-۱-مقدمه: ۲۲

۳-۲-شبکه بندی ۲۳

۳-۳- روش تک بلوکی یا تک ناحیه‌ای ۲۵

۳-۴- روش چند ناحیه‌ای ۲۶

۳-۵- روش چندبلوکی ۲۶

فصل ۴-محاسبه مشتقات جزئی به روش عددی ۳۸

۴-۱-مقدمه ۳۸

۴-۲-روش تفاضلات محدود فشرده برای محاسبهٔ مشتقات ۳۸

۴-۳- نگاشت یک به یک جبری(Algebraic mapping) ۴۱

۴-۴-استخراج اپراتورهای مشتق در حالت استفاده از تابع نگاشت ۴۳

۴-۵-ارزیابی و تست کدهای نوشته شده برای محاسبهٔ مشتقات ۴۴
۴-۶-الگوی پیشروی در زمان ۵۲
فصل ۵-معادلات دیفرانسیل حاکم و الگوریتم حل عددی آنها ۵۵
۵-۱-مقدمه ۵۵
۵-۲-شکل چرخشی معادلهٔ ناویر- استوکس ۵۵
۵-۳- روش حل عددی معادلات ۵۷
۵-۴-روش گسسته‌سازی معادلهٔ پواسون دو بعدی ۵۸
۵-۵- حل معادلهٔ ماتریسی AX+XB=C ۵۹
۵-۶- نحوهٔ اعمال شرایط مرزی ۶۴
۵-۷- شرط اولیه ۶۵
۵-۸- الگوریتم حل عددی ۶۶
فصل ۶- معرفی نرم افزار و ارائه نتایج ۶۸
۶-۱- مقدمه ۶۸
۶-۲- معرفی نرم افزار و قسمتهای مختلف آن ۶۸
۶-۳- گردابه های استوارت ۷۵
۶-۴-نتایج حل عددی برای جت ۷۶
۶-۵-نتایج حل عددی برای دنباله ۷۹
۶-۶-نتایج حل عددی برای لایهٔ اختلاطی ۸۳
ضمیمهٔ الف) تحلیل خطا ۸۶
ضمیمهٔ ب) ماتریس هسنبرگ ۸۷
ضمیمهٔج) لایهٔ مرزی روی صفحهٔ تخت ۸۸
ضمیمهٔ د) کد مستقیم عددی جریان برشی ۱۰۲
فهرست مراجع: ۱۷۲
Abstract ۱۷۴

فهرست اشکال

موضوع	صفحه
شکل۱-۱: نمایی از جت، دنباله و لایهٔ اختلاطی و نوع پروفیل سرعت	۴
شکل۱-۲: پروفیل سرعت جت در پائین دست از سوراخ جت	۱۲
شکل۱-۳ : نمودار خود تشابهی جت	۱۲
شکل۱-۴ : جریان دنباله پشت صفحهٔ تخت	۱۳
شکل ۱-۵: پروفیل سرعت در لایه اختلاطی.	۱۶
شکل ۱-۶: هندسه لایه اختلاطی توسعه یافته مکانی	۱۷
شکل ۱-۷: توزیع سرعت در جریان اختلاطی در دستگاه مختصات خود تشابه برای ایستگاههای مختلف جریان	۱۷
شکل ۳-۱: تولید شبکه روی هندسه‌های پیچیده با استفاده از روش‌های، الف: باسازمان،ب: بی‌سازمان.	۲۵
شکل۳-۲ : نمایش انواع مختلف شبکه‌های بلوکی.	۳۲
شکل۳-۳: تولید شبکه سازمان یافته به روش‌های مختلف، الف: شبکه باسازمان، ب: شبکه بلوکی منطبق، ج: شبکه بلوکی وصله‌ای، د: شبکه بلوکی هم‌پوشان .	۳۵
شکل۳-۴: تقسیم ناحیه ما بین دو ایرفویل به پنج بلوک.	۳۶
شکل۳-۵: انواع مرزهای بلوکی. الف: مرز با پیوستگی شبکه، ب: مرز باپیوستگی متریک،ج: مرز با ناپیوستگی متریک، د: مرز با ناپیوستگی شبکه	۳۷
شکل۴-۱: توزیع شبکه در یک شبکه بندی کشیده شدهٔ جبری با $N_y = 50$, $L_y = 6$, $y_0 = 0$ با استفاده از تابع نگاشت (۴-۱۴)	۴۲
شکل۴-۲:نمودار تابع $f(x) = x^3 \cos x + 2x \sin(4x)$	۴۵
شکل۴-۳: نمودار مشتق اول تابع $f(x) = x^3 \cos x + 2x \sin(4x)$ (۲۰۰ گره)	۴۶

موضوع	صفحه
شکل۴-۴: نمودار تابع $f(x)=\sum_{i=1}^{30} a_i(\sin x + ix\cos(ix))$	۴۷
شکل۴-۵: نمودار مشتق اول تابع $f(x)=\sum_{i=1}^{30} a_i(\sin x + ix\cos(ix))$ (با ۱۰۰۰گره)	۴۷
شکل۴-۶: نمودار مشتق دوم تابع $f(x)=e^{x^2}$ (با ۱۰۰گره)	۴۸
شکل۴-۷: نمودار خطای نسبی مشتق دوم تابع $f(x)=e^{x^2}$ (با ۱۰۰گره)	۴۹
شکل۴-۸: نمودار مشتق اول تابع $f(x)=e^{x^2}$ با اعمال نگاشت رابطهٔ۴-۲۵(با ۱۰۰گره)	۵۰
شکل۴-۹: نمودار مشتق دوم تابع $f(x)=e^{x^2}$ با اعمال نگاشت۴-۲۵(با ۱۰۰گره)	۵۰
شکل۴-۱۰: نمودار مشتق اول تابع $f(x)=e^{x^2}$ با اعمال نگاشت رابطه ۴-۲۶(با ۱۰۰گره)	۵۱
شکل۴-۱۱: نمودار مشتق دوم تابع $f(x)=e^{x^2}$ با اعمال نگاشت رابطه ۴-۲۶(با ۱۰۰گره)	۵۲
شکل۴-۱۲: مرتبه دقت الگوی پیشروی زمان	۵۴
شکل۶-۱: منوی اصلی نرم افزار	۶۹
شکل۶-۲ : وارد کردن پارامترهای جدید برای حل عددی	۶۹
شکل۶-۳ : بزرگ بودن عدد CFL از معیار تعیین شده توسط کاربر	۷۰
شکل۶-۴: کوچک بودن عدد CFL از مقدار Min CFL	۷۱
شکل۶-۵: گزارش ذخیره اطلاعات حاصل از حل عددی در بازه‌های مشخص شده	۷۲
شکل۶-۶: فرمت فایل ذخیره شده توسط نرم افزار	۷۲
شکل۶-۷: ادامهٔ حل از فایل تاریخچه با پسوند dib.*	۷۳
شکل۶-۸: مقادیر سرعت در جهت y در گره‌های مختلف	۷۴
شکل ۶-۹: ماکزیمم خطا در u و v برای تست گردابه های استوارت به صورت تابعی از زمان	۷۶
شکل ۶-۱۰: پارامترهای ورودی برای شبیه سازی جت بدون اغتشاش ورودی	۷۶
شکل ۶-۱۱: پروفیل سرعت u در مختصات خود تشابه برای شبیه سازی جت بدون اغتشاش ورودی	۷۷

موضوع **صفحه**

شکل ۶-۱۲: پروفیل ω در مختصات خود تشابه برای شبیه سازی جت بدون اغتشاش ورودی ۷۷

شکل ۶-۱۳: سرعت خط مرکزی برای شبیه سازی جت بدون اغتشاش ورودی ۷۸

شکل ۶-۱۴: ضخامت نیم عرض جت و مقایسه با روش تحلیلی برای شبیه سازی جت بدون اغتشاش ورودی ۷۸

شکل ۶-۱۵: پارامترهای ورودی برای شبیه سازی دنباله بدون اغتشاش ورودی ۷۹

شکل ۶-۱۶: پروفیل سرعت u در مختصات خود تشابه برای شبیه سازی دنباله بدون اغتشاش ورودی ۷۹

شکل ۶-۱۷: پروفیل ω در مختصات خود تشابه برای شبیه سازی دنباله بدون اغتشاش ورودی ۸۰

شکل ۶-۱۸: بردار سرعت برای شبیه سازی دنباله بدون اغتشاش ورودی ۸۰

شکل ۶-۱۹: سرعت خط مرکزی برای شبیه سازی دنباله بدون اغتشاش ورودی ۸۱

شکل ۶-۲۰: ضخامت نیم عرض دنباله و مقایسه با روش تحلیلی برای شبیه سازی دنباله بدون اغتشاش ورودی ۸۱

شکل ۶-۲۱: تاریخچهٔ زمانی سرعت u در ایستگاههای مختلف برای شبیه سازی دنباله بدون اغتشاش ورودی ۸۲

شکل ۶-۲۲: تاریخچهٔ زمانی سرعتv در ایستگاههای مختلف برای شبیه سازی دنباله بدون اغتشاش ورودی ۸۲

شکل ۶-۲۳: تاریخچهٔ زمانی گردابه در ایستگاههای مختلف برای شبیه سازی دنباله بدون اغتشاش ورودی ۸۳

شکل ۶-۲۴: پروفیل سرعت u در مختصات خود تشابه برای شبیه سازی لایهٔ اختلاطی بدون اغتشاش ورودی ۸۴

شکل ۶-۲۵: پروفیل ω در مختصات خود تشابه برای شبیه سازی لایهٔ اختلاطی بدون اغتشاش ورودی ۸۴

موضوع **صفحه**

شکل ۶-۲۶: سرعت خط مرکزی برای شبیه سازی لایهٔ اختلاطی بدون اغتشاش ورودی ۸۵

شکل ج-۱: لایه مرزی روی صفحهٔ تخت ۸۸

شکل ج-۲: نمودار توابع f, f', f'' ۹۲

شکل ج-۳: نمودار مولفهٔ $\frac{u}{U_\infty}$ سرعت و مقایسهٔ ان با نتایج نیکورادزه ۹۲

شکل ج-۴ : نمودار نتایج حل عددی معادلهٔ بلازیوس به روش Shooting ۱۰۱

شکل ج-۵ : حل عددی معادلهٔ بلازیوس ، نمودارهای f, f', f'', f''' و $f''' + \frac{1}{2} f f''$ ۱۰۱

فهرست جداول

موضوع	صفحه
جدول ۱-۱: بالانس نرخ حجم و مومنتوم در جهت x برای سطح کنترل شکل۱-۴	۱۴
جدول ۴-۱:الگوی پیشروی در زمان رانج – کوتا مرتبهٔ سوم	۵۲
جدول ج-۱: نتایج حل عددی بلازیوس به روشShooting	۹۹

مقدمه

همانطور که می دانیم برای بررسی جریان یکی از روشهای متداول آزمایش و استفاده از داده های تجربی است. با وجود اینکه داده های تجربی و آزمایشگاهی بسیار کارآمد هستند ولی هزینه بسیار بالا و سرسام آور آن باعث شده است تا دانشمندان به دنبال روشهایی باشند تا بتوانند نتایج یک رخداد فیزیکی را بدون آزمایش و با هزینه کمتر پیش بینی کنند. یکی از این روشها که جدیداً در همهٔ علوم علی الخصوص مهندسی مکانیک استفاده می شود ، استفاده از روشهای عددی است. این روشها نه تنها هزینهٔ سخت افزاری کمتری لازم دارد بلکه محدودیت آزمایش را ندارد و می توان برای شرایط مرزی مختلف از آن استفاده کرده و مسئله را تحلیل کرد. یکی از روشهای عددی مرسوم، روش حل مستقیم عددی است؛ در این روش معادلات ناویر- استوکس بدون هیچ مدل توربولانس حل می شوند . چون در این روش معادلات ناویر- استوکس به صورت مستقیم حل می شوند؛برای داشتن دقت کافی باید تعداد گره ها در دامنه به مقدار قابل توجهی زیاد باشد که در نتیجه برای ارضای عدد CFL باید گام زمانی آن نیز کوچک باشد. به این دلیل هزینهٔ محاسبات در روش مستقیم عددی بالاست.

در این پایان نامه سعی شده است تا جریان برشی شامل جت – دنباله و لایه اختلاطی به وسیلهٔ یکی از روشهای مستقیم حل شود. در این حل مشتقات مکانی در جهت x و y به ترتیب به روش تفاضلات محدود فشرده و تفاضلات فشردهٔ تطبیقی محاسبه شده است در جهت عمود بر جریان از یک نگاشت یک به یک برای مرتبط نمودن توزیع گره ها در مختصات فیزیکی و مختصات محاسباتی استفاده شده است.

درفصل اول مطالعات انجام شده قبلی بر روی جریانهای برشی بررسی خواهد شد. این فصل بیشتر شامل حل تحلیلی و دقیقی است که به روش تشابه ارائه شده است.

در فصل دوم ، مختصری راجع به روش مستقیم عددی بحث خواهیم کرد. درفصل بعد راجع به روشهای تقسیم دامنه بحث خواهد شد. در این فصل انواع مختلف تقسیم دامنه و همچنین مزایا و معایب هر کدام بحث خواهد شد. در فصل چهارم روش محاسبهٔ مشتقات جزئی مکانی و زمانی بحث خواهد شد. مشتقات مکانی به روش تفاضلات محدود فشرده محاسبه شده است این روش دقت بالایی دارد وبرای حل به روش مستقیم عددی یک روش مناسب است. مشتقات زمانی به روش رانگ – کوتای مرتبهٔ سوم محاسبه شده است در این روش برای حل معادلهٔ دیفرانسیل شامل زمان در یک پیشروی به اندازهٔ Δt، مسئله در سه زیر بازهٔ زمانی حل خواهد شد که این روش نیز از دقت مناسبی برخوردار است. در فصل پنجم معادلات دیفرانسیل حاکم بر حرکت سیال که همان معادلات ناویر-استوکس هست بررسی خواهد شد. ابتدا معادلهٔ مومنتوم را به فرمی تبدیل خواهیم کرد که ترم فشار از معادلهٔ مومنتوم حذف شود با اعمال این تغییرات فشار از معادلهٔ مومنتوم حذف می‌شود ولی مرتبهٔ مشتقات جزئی بیشتر می‌شود که این به نوبهٔ خود باعث می‌شود که هزینهٔ محاسبات بیشتر شده و اعمال شرایط مرزی سخت‌تر شود. در ادمه نحوهٔ گسسته سازی معادلات را بررسی خواهیم کرد که در نتیجهٔ گسسته سازی معادلهٔ پواسون به معادلهٔ ماتریسی $AX+XB=C$ می‌رسیم که نحوهٔ حل آن نیز بررسی خواهد شد.

در فصل ششم نرم افزار ارائه شده و قابلیت‌های آن معرفی شده است. و در آخر نتایج حاصل از حل عددی ارائه شده است و نتایج مورد بحث و بررسی قرار گرفته است.

فصل ۱- مطالعهٔ جریان برشی

۱-۱-مقدمه:

در این فصل ابتدا انواع جریانهای برشی آزاد بررسی شده و دسته بندی می‌شوندو سپس حلهای تحلیلی و مطالعات قبلی که روی این جریانها انجام شده است مورد بررسی قرار خواهد گرفت.

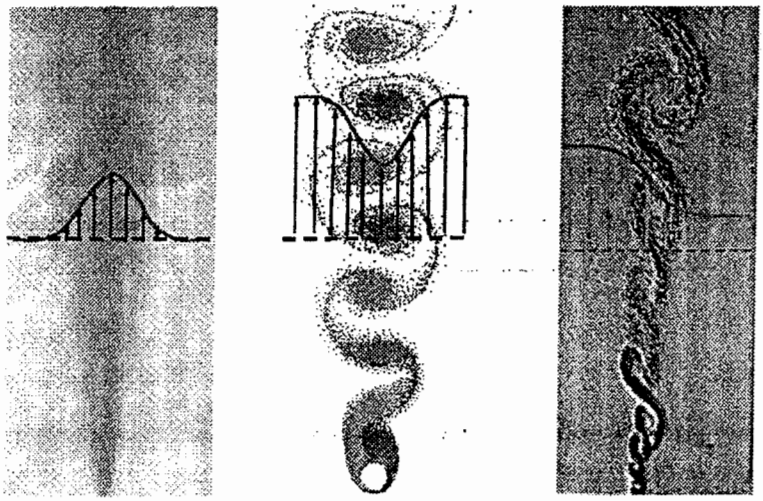
۱-۲- جریان های برشی آزاد

در جریان های برشی آزاد سیال هیچ تماس فیزیکی مرزی با محیط خود ندارد. به همین علت به آن جریانهای برش آزاد می گویند. از انواع جریانهای برشی آزاد می توان دنباله[۱]، جت[۲]، لایه های اختلاطی[۳] و پلوم[۴] را نام برد درشکل ۱-۱ می توان انها را دید.

در اینگونه جریانها با پیشروی برش آزاد در محیط، سیال محیط را با خود حمل می کند و محیط گردابه ای تولید می کند. با بی بعد کردن ابعاد با یک طول مشخصه خاص و سرعت با مشخصهٔ سرعت، پروفیل های سرعت و توزیع تنش های رینولدز در ایستگاههای مختلف در دامنهٔ وسیعی به خوبی به روی هم منطبق می شوند. به این پدیده خودتشابهی می گویند.[16]

---

[1] Wake
[2] Jet
[3] Mixing Layer
[4] Plum



شکل۱-۱: نمایی از جت، دنباله و لایهٔ اختلاطی و نوع پروفیل سرعت [۷]

جریانهای برشی یکی از اشکال اصلی جریان محسوب می شود که به طور وسیع در بسیاری از کاربردهای عملی استفاده می شود. به عنوان مثال جریان جت درمحفظهٔ پاشش سوخت، جلوبندی در موتور جت و ... به کاربرده می‌شود. یکی از مهمترین بررسی‌های انجام شده برای جریان جت فرآیند اختلاط و حمل سیال محیط یک جت با توجه به محیط خودش است. به عنوان مثال پاشش سوخت باعث اختلاط یکنواخت‌تر شده و در نتیجه بازده افزایش می‌یابد و کارکرد موتور بهتر می‌شود. از کاربردهای جریان دنباله می توان در طراحی زیردریایی و کشتیها و ایرفویلها اشاره کرد. لایه های اختلاطی در طراحی توربینهای گازی و محاسبات لبهٔ فرار پره توربین ها و یا واکنش‌های شیمیایی استفادهٔ‌گسترده‌ای دارد.

۱-۳-تحلیل جریان جت ایده آل دو بعدی

در این قسمت یک جت دو بعدی و صفحه‌ای را که بوسیلهٔ تئوری لایهٔ مرزی دو بعدی مدل شده است و دارای یک حل تحلیلی و دقیق است، ارائه می‌دهیم.

در این قسمت یک جت دو بعدی و پایا را مطالعه می کنیم که در آن سیال از یک سوراخ به یبرون می‌جهد. فشار در داخل محیط (هوا) ثابت فرض می‌شود. اگر فرض کنیم که سرعت هـوا در بینهایت صفر باشد آنگاه سرعت جریان ایده‌آل در بیرون از جت صفر خواهد بود.

در عمل جریان جت با اعداد رینولدز بالا توربولانت است. اما در این قسمت فرض مـی‌کنـیم که حالت لامینار داریم. این در حالی است که روش ریاضی برای جت توربولانـت بـر اسـاس رفتـار آرام مسئله بنا می‌شود.

در ابتدا ، جت مقداری از سیال محیط را که در حال سکون است با خود حمل می کند ولی این حرکت به علت اصطکاک در پیرامون جت از بین رفتـه و سـاکن مـی‌شـود . در نتیجـه تلفـات ویسکوزیته در داخل جت اهمیت زیادی دارد.

در یک ایستگاه مناسب در فاصلهٔ x که در آن جت باریک است ، گرادیان تندی در جهـت y وجود دارد یعنی :

$$\frac{\partial}{\partial y} >> \frac{\partial}{\partial x} \qquad (۱-۱)$$

برای سادگی یک سوراخ بی‌نهایت باریک را در نظر می‌گیریم ، اما ایـن فـرض نیازمنـد ایـن است که سرعت بی‌نهایت بزرگی در سوراخ داشته باشیم (حالت نقطهٔ منفرد) تا یک حجم محدود و مومنتوم محدود در جت داشته باشیم. معادلات بعـددار لایـه مـرزی را در ایـن قـسمت یـادآوری می کنیم:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \qquad (۲-۱)$$ پیوستگی

$$u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = u_e \frac{du_e}{dx} + \upsilon \frac{\partial^2 u}{\partial y^2} \qquad (۳-۱)$$ معادلهٔ مومنتوم

در روابط بالا $u_e(x)$ سرعت جریان ایده‌آل در بیرون از لایهٔ مرزی است.

سوالی که در اینجا مطرح مـی‌شـود ایـن اسـت کـه شـرایط مـرزی لازم بـرای ایـن روابـط چیست؟(در این قسمت هیچ مرز جامدی برای اعمال اصل عدم لغزش وجود ندارد.)

شرایط مرزی به شرح زیر است:

- ✓ $u_e(x) = 0$ ، در نتیجه در معادلهٔ مومنتوم داریم: $u_e du_e / dx = 0$

✓ فرض می‌شود که جت در y=0 متقارن است بنا براین

$$u(y)=u(-y)$$
$$\left(\frac{\partial u}{\partial y}\right)_{y=0^+}=-\left(\frac{\partial u}{\partial y}\right)_{y=0^-} \quad \Rightarrow \quad \left(\frac{\partial u}{\partial y}\right)_{y=0}=0 \qquad (۴-۱)$$
$$v(y=0^+)=-v(y=0^-) \quad \Rightarrow \quad v(y=0)=0$$

✓ آخرین شرط مرزی این است که این است که کل انتقال مومنتوم در جهت x مقداری ثابت است(و بنابراین مستقل از x است.)به طور فیزیکی جت به صورت واگرا نسبت به x افزایش می‌یابد حال آنکه مقدار مومنتوم و سرعت آن در مرکز کاهش می‌یابد.

می خواهیم نشان دهیم که

$$M=\rho\int_{-\infty}^{+\infty}u^2dy=\text{Constant} \qquad (۵-۱)$$

که ρ در این رابطه دانسیتهٔ سیال مورد نظر است که ثابت فرض می شود.

معادلهٔ (۱-۳) را می توان به صورت زیر نوشت:

$$u\frac{\partial u}{\partial x}+\frac{\partial}{\partial y}(uv)-u\frac{\partial v}{\partial y}=\upsilon\frac{\partial^2 u}{\partial y^2} \qquad (۶-۱)$$

معادلهٔ پیوستگی را باز نویسی کرده به صورت زیر می نویسیم:

$$\frac{\partial u}{\partial x}=-\frac{\partial v}{\partial y} \qquad (۷-۱)$$

با جاگذاری رابطهٔ (۱-۷) در (۱-۶) معادلهٔ مومنتوم به صورت زیر در می آید:

$$u\frac{\partial u}{\partial x}+\frac{\partial}{\partial y}(uv)+u\frac{\partial u}{\partial x}=\upsilon\frac{\partial^2 u}{\partial y^2} \qquad (۸-۱)$$

و یا:

$$\frac{\partial}{\partial x}(u^2)+\frac{\partial}{\partial y}(uv)=\upsilon\frac{\partial^2 u}{\partial y^2} \qquad (۹-۱)$$

با انتگرال گیری در یک ایستگاه مشخص از $y=-\infty$ تا $y=+\infty$ و اعمال تقارن جت به رابطهٔ زیر می رسیم:

$$\frac{d}{dx}\int_{-\infty}^{+\infty}u^2dy+2[vu]_0^{+\infty}=2\upsilon\left[\frac{\partial u}{\partial y}\right]_0^{+\infty} \qquad (۱۰-۱)$$

روابـط $v(0)=0$ و $\frac{\partial u}{\partial y}(y=0)=0$ و همچنـین $u(\infty)=0$ ، $\frac{\partial u}{\partial y}(y=\infty)=0$ را داریـم ؛

بنابراین روابط زیر را می توان استخراج کرد:

$$\frac{d}{dx}\int_{-\infty}^{+\infty}u^2dy=0 \qquad (۱۱-۱)$$

و

$$\frac{d}{dx}\int_{-\infty}^{+\infty}u^2dy=\text{constant}=\frac{M}{\rho} \qquad (۱۲-۱)$$

M یک پارامتر معلوم از مسئله اسـت کـه نمایـانگر نـرخ انتقـال مومنتـوم در یـک ایـستگاه مشخص است.

بعد M: $[M]=\rho U^2L$ که U و L نمایـانگر سـرعت مشخـصه و طـول مشخـصه اسـت. در نتیجه می توانیم عدد رینولدز را بر اساس M تعریف کنیم:

$$\text{Re}=\frac{UL}{\upsilon}=\left(\frac{M}{\rho L}\right)^{1/2}\frac{L}{\upsilon}=\left(\frac{ML}{\rho\upsilon^2}\right)^{1/2} \qquad (۱۳-۱)$$

برای اعداد $\text{Re}<<1$، تئوری لایهٔ مرزی آرام را اعمال می کنیم تا بتوانیم معادلات حاکم زیر را حل کنیم :

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}=0 \qquad (۱۴-۱)$$

$$u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}=u_e\frac{du_e}{dx}+\upsilon\frac{\partial^2u}{\partial y^2} \qquad (۱۵-۱)$$

که شرایط مرزی عبارتند از :

$$u \to 0 \quad \text{as} \quad y \to \infty$$
$$\frac{\partial u}{\partial y} = 0 \quad \text{on} \quad y = 0 \qquad (۱۶-۱)$$
$$\rho \int_{-\infty}^{+\infty} u^2 dy = M$$

دنبال یک راه حل به صورت زیر هستیم:

$$u(x,y) = x^p f'\left(\frac{y}{x^q}\right) \qquad (۱۷-۱)$$

که متغیر تشابهی به صورت زیر است :

$$\eta = \frac{y}{x^q} \qquad (۱۸-۱)$$

برای اینکه سومین شرط مرزی را ارضا کنیم ، باید داشته باشیم:

$$x^{2p} x^q = x^0 \qquad (۲۰-۱)$$

یا :

$$2p + q = 0 \qquad (۲۱-۱)$$

همچنین برای اینکه ترم اینرسی و ویسکوزیته بتوانند همدیگر را در معادلهٔ مومنتوم بالانس کنند ، باید داشته باشیم:

$$u\frac{\partial u}{\partial x} \sim \frac{u}{x} \sim x^{2p-1} \sim \upsilon \frac{\partial^2 u}{\partial y^2} \sim \frac{u}{y^2} \sim \frac{x^p}{x^{2q}} \qquad (۲۲-۱)$$

و یا :

$$p + 2q = 1. \qquad (۲۳-۱)$$

بنابراین دو معادلهٔ خطی بر حسب p و q داریم که می توانیم آنها را حل کنیم :

$$p = -\frac{1}{3}, \quad q = \frac{2}{3} \qquad (۲۴-۱)$$

با استفاده از این آنالیز توانستیم رابطهٔ x و متغیر تشابهی η را بدست آوریم اکنون می توانیم بنویسیم :

$$\eta = \frac{1}{3\upsilon^{1/2}} \frac{y}{x^{2/3}}, \quad \zeta = x \qquad (۲۵-۱)$$

در این قسمت معادلهٔ خط جریان را به صورت زیر در نظر می گیریم:

$$\mathrm{Re}=\frac{UL}{\upsilon}=\left(\frac{M}{\rho L}\right)^{1/2}\frac{L}{\upsilon}=\left(\frac{ML}{\rho\upsilon^{2}}\right)^{1/2} \qquad (۲۶-۱)$$

بنابراین $u=\dfrac{\partial\Psi}{\partial y}$ دارای وابستگی با x به صورت $x^{-1/3}$ است.

تغییر مختصات منجر به روابط زیر می شود:

$$\frac{\partial}{\partial x}=\frac{\partial}{\partial\zeta}-\frac{2\eta\partial}{3x\partial\eta}$$
$$\frac{\partial}{\partial y}=\frac{x^{-2/3}}{3\upsilon^{1/2}}\frac{\partial}{\partial\eta} \qquad (۲۷-۱)$$

عبارتهای نظیر u و v در مختصات جدید به صورت زیر محاسبه می شود:

$$u=\frac{\partial\Psi}{\partial y}=\frac{x^{-1/3}}{3}f'(\eta)$$
$$and \qquad (۲۸-۱)$$
$$v=-\frac{\partial\Psi}{\partial x}=-\frac{\upsilon^{1/2}}{3}x^{-2/3}\{f-2\eta f'\}$$

با جاگذاری در معادلهٔ مومنتوم زیر :

$$u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}=\upsilon\frac{\partial^{2}u}{\partial y^{2}} \qquad (۲۹-۱)$$

بدست می آوریم:

$$\upsilon\frac{x^{-2/3}}{3\upsilon^{1/2}}\frac{\partial}{\partial\eta}\left\{\frac{x^{-2/3}}{3\upsilon^{1/2}}\frac{\partial}{\partial\eta}\left(\frac{x^{-1/3}}{3}f'\right)\right\}=\frac{x^{-1/3}}{3}f'\left\{\frac{\partial}{\partial x}-\frac{2\eta}{3x}\frac{\partial}{\partial\eta}\right\}\left(\frac{x^{-1/3}}{3}f'\right)$$
$$-\frac{\upsilon^{1/2}}{3}x^{-2/3}\{f-2\eta f'\}\frac{x^{-2/3}}{3\upsilon^{1/2}}\frac{\partial}{\partial\eta}\left(\frac{x^{-1/3}}{3}f'\right) \qquad (۳۰-۱)$$

با ضرب طرفین رابطهٔ بالا در $27x^{5/3}$ و ساده سازی به رابطهٔ زیر می رسیم:

$$f''' + ff'' + f'^2 = 0 \qquad (۳۱-۱)$$

کـه معادلـه بـالا دارای شـرایط مـرزی $f'(\infty) \to 0$ و $f''(0) = 0$ و همچنـین از تقـارن $v(y=0) = 0$ بنابراین $f(0) = 0$، است.

معادلهٔ (۱-۳۱) را می توان به صورت زیر نوشت:

$$f''' + \frac{d}{d\eta}(ff') = 0 \qquad (۳۲-۱)$$

یا :

$$f'' + ff' = C \qquad (۳۳-۱)$$

که C یک ثابت است و چون $f(0) = f''(0) = 0$. در نتیجه C=0.

در این مرحله بعد از انتگرالگیری مجدد می توان مستقیماً f را استخراج کرد، اما یـک روش زیبا با تغییر متغیر وجود دارد:

$$\chi = a\eta \quad , \quad f(\eta) = 2aF(\chi) \qquad (۳۴-۱)$$

که a یک ثابت آزاد است. معادلهٔ (۱-۳۳) به صورت زیر تبدیل می شود:

$$2a^3F'' + 4a^3FF' = 0 \qquad (۳۵-۱)$$

یا:

$$F'' + 2FF' = 0 \qquad (۳۶-۱)$$

که شرایط مرزی این معادله به صورت زیر است:

$$\begin{aligned} &\chi = 0 \ , \ \mathrm{F} = 0 \\ &\chi = \infty \ , \ \mathrm{F}' = 0 \end{aligned} \qquad (۳۷-۱)$$

با یک بار انتگرال گیری از معادلهٔ بالا رابطهٔ زیر حاصل می شود:

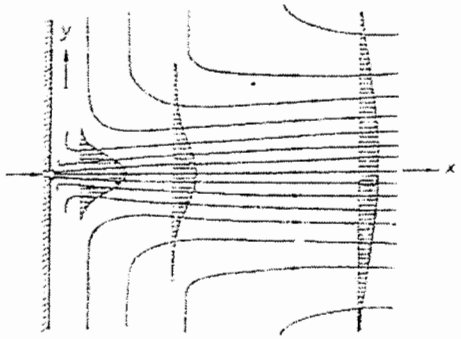
$$F' + F^2 = 1. \qquad (۳۸-۱)$$

بدون اینکه لطمه ای به فرضیات مـسئله وارد شـود ثابـت انتگرالگیـری برابـر یـک انتخـاب می شود چرا که هنوز مقدار a تعیین نشده است(مقداری که معادل با $F'(0) = 1$ باشد.)

و سرانجام رابطه ای برای aبدست می آید:

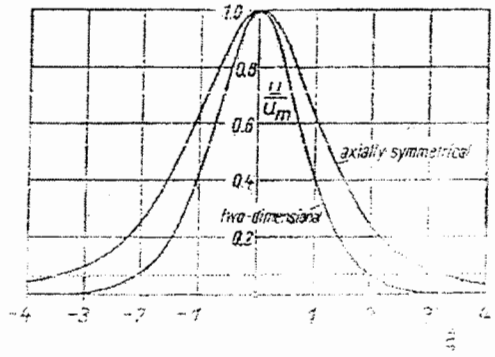
$$a=\left(\frac{9}{16}\right)^{1/3}\left(\frac{M}{\rho v^{1/2}}\right)^{1/3} \qquad (۴۶-۱)$$

براساس نتایج بالا، می توان عبارتهایی برای uو v بدست آورد.



شکل۱-۲: پروفیل سرعت جت در پائین دست از سوراخ جت[10]

و شکل۱-۳ نمودار تشابهی سرعت جت را نشان می دهد.



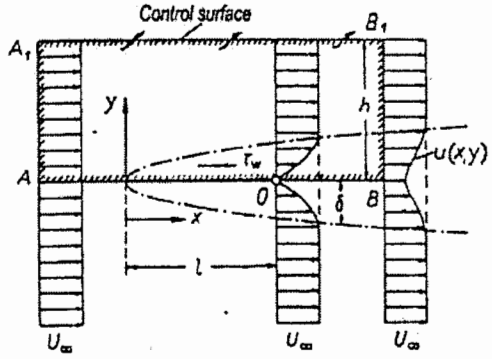
شکل۱-۳ : نمودار خود تشابهی جت[10]

۱-۴- تحلیل جریان دنباله

همانطور که می دانیم استفاده از معادلات لایهٔ مرزی الزاماً برای جریان روی دیوارهٔ ثابت متمرکز نشده است. شکل(۱-۴) جریان دنباله پشت یک صفحهٔ تخت را نشان می دهد. پروفیل

سرعت دنباله با افزایش x کامل تر شده و مقدار نقصان آن کمتر می شود در این قسمت فرض می‌شود که اندازهٔ نقصان سرعت یعنی:

$$u_1(x,y) = U_\infty - u(x,y) \qquad (۴۷-۱)$$



شکل۱-۴ : جریان دنباله پشت صفحهٔ تخت[10]

وقتی $x \to \infty$ میل می کند u_1 در مقایسه با U_∞ مقدار ناچیزی دارد. بنابراین می توان از عبارتهای مرتبهٔ دوم u_1 و v_1 صرف نظر کرد. با توجه به اینکه در پائین دست جریان فشار کم است، می توان معادلات لایهٔ مرزی را با صرف نظر از عبارتهای مرتبهٔ دوم به صورت زیر نوشت:

$$U_\infty \frac{\partial u_1}{\partial x} = \nu \frac{\partial^2 u_1}{\partial y^2} \qquad (۴۸-۱)$$

که شرایط مرزی آن به صورت زیر است:

$$\frac{\partial u_1}{\partial y}(y=0) = 0 \quad , \quad y \to \infty \Rightarrow u_1 \to 0 \qquad (۴۹-۱)$$

این معادله یک معادلهٔ خطی با مشتقات جزئی است . این خطی سازی بوسیلهٔ فرضیاتی در ترم u_1 یعنی اغتشاشات انجام شده است. حال اگر متغیر تشابهی η را به صورت زیر تعریف کنیم:

$$u_1 = U_\infty C\left(\frac{x}{l}\right)^{-m} F(\eta), \eta = \frac{y}{2}\sqrt{\frac{U_\infty}{\nu x}} \qquad (۵۰-۱)$$

آنگاه به معادلهٔ دیفرانسیل زیر خواهیم رسید:

$$F'' + 2\eta F' + 4mF = 0 \qquad (۵۱-۱)$$

که شرایط مرزی به صورت زیر است:

$$\eta = 0 : F' = 0; \eta \to \infty : F = 0 \qquad (۵۲-۱)$$

بالانس نرخ حجم و مومنتوم در جهت x برای سطح کنترل در جدول(۱-۱) نشان داده شده است:

جدول ۱-۱: بالانس نرخ حجم و مومنتوم در جهت x برای سطح کنترل شکل۱-۴

$cross-section$	Volume flux	x momentum
AB	0	0
AA_1	$b\int_0^h U_\infty dy$	$\rho b\int_0^h U_\infty^2 dy$
BB_1	$-b\int_0^h u dy$	$-\rho b\int_0^h u^2 dy$
A_1B_1	$-b\int_0^h (U_\infty - u)dy$	$-\rho b\int_0^h U_\infty (U_\infty - u)dy$
$\sum$ = control surface	$\sum$ volume flux = 0	$\sum$ momentum flux = drag

مقدار پارامتر مجهول m را می توان از بالانس مومنتوم حول جسم در شکل(۱-۴) بدست آورد. سطح کنترل مستطیلی AA_1BB_1 به اندازهٔ کافی دور از جسم در نظر گرفته شده است به طوریکه اغتشاشات فشار نداشته باشیم، در نتیجه فشار در کل سطح کنترل ثابت بوده و در نتیجه نیروی حاصل از فشار در بالانس مومنتوم اثری ندارد. به این دلیل که باید معادلات پیوستگی ارضا شوند در نتیجه سیال باید از بالا و پائین سطح کنترل خارج شود. مقدار کمیت سیال خارج شده از A_1B_1 باید با اختلاف سیال ورودی از AA_1 و سیال خارج شده از BB_1 برابر باشد. همانطور که در جدول (۱-۱) نشان داده شده است، نرخ حجم ورودی مثبت و نرخ حجم خروجی منفی در نظر گرفته شده است. حال می‌توانیم نیروی دراگ را بوسیلهٔ نرخ کل مومنتوم حساب کنیم:

$$D = b\rho\int_{-\infty}^{+\infty} u(U_\infty - u)dy \qquad (۵۳-۱)$$

که حدود انتگرال بجای $y=\pm h$ باید با $y=\pm\infty$ جاگذاری شود با فرضیات قبلی می‌توان معادلهٔ (۱-۵۳) را به صورت زیر نوشت

$$D \approx be\int_{-\infty}^{+\infty} U_\infty u_1 dy = 2b\rho U_\infty^2 C\left(\frac{x}{l}\right)^{-m}\sqrt{\frac{\nu x}{U_\infty}}\int_{-\infty}^{+\infty} F(\eta)d\eta \qquad (۱-۵۴)$$

با توجه به اینکه بالانس مستقل از x است می‌توان مقدار پارامتر m را در رابطهٔ(۱-۸۸) بدست آورد که برابر m=0.5 است. در نتیجه رابطهٔ (۱-۵۰) به صورت زیر تبدیل می‌شود:

$$F'' + 2\eta F' + 2F = 0 \qquad (۱-۵۵)$$

با یک بار انتگرال گیری به رابطهٔ زیر می رسیم:

$$F' + 2\eta F = 0 \qquad (۱-۵۶)$$

حل نهایی به صورت زیر است:

$$F(\eta) = e^{-\eta^2} \qquad (۱-۵۷)$$

با توجه به رابطهٔ انتگرالی زیر

$$\int_{-\infty}^{+\infty} F(\eta)d\eta = \int_{-\infty}^{+\infty} e^{-\eta^2} d\eta = \sqrt{\pi} \qquad (۱-۵۸)$$

می توان ضریب دراگ را به صورت زیر استخراج کرد:

$$c_D = \frac{D}{\frac{\rho}{2}U_\infty^2 bl} = \frac{4\sqrt{\pi}C}{\sqrt{\frac{U_\infty l}{\nu}}} \qquad (۱-۵۹)$$

در نتیجه حل نهایی برای ترم u_1 برای جریان دنباله از رابطهٔ زیر استخراج می‌شود که C_D ضریب دراگ است.

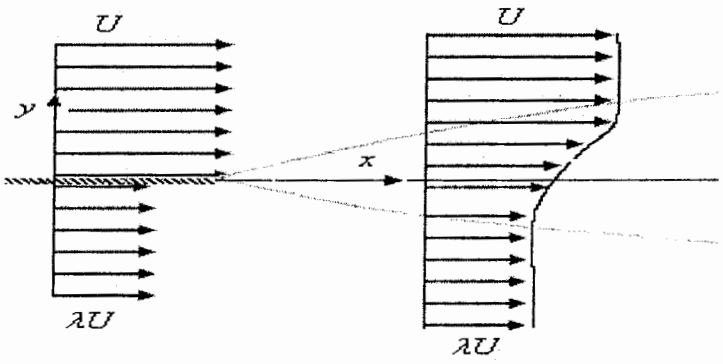
$$\frac{u_1(x,y)}{U_\infty} = \frac{c_D}{4\sqrt{\pi}}\sqrt{\frac{U_\infty l}{\nu}}\left(\frac{x}{l}\right)^{-\frac{1}{2}}\exp\left(-\frac{y^2 U_\infty}{4x\nu}\right) \qquad (۱-۶۰)$$

از رابطهٔ (۱-۶۰) می توان نیم ضخامت دنباله را به صورت زیر استخراج کرد:

$$y_{0.5} = 1.7\sqrt{\frac{\nu x}{U_\infty}} \qquad (۱-۶۱)$$

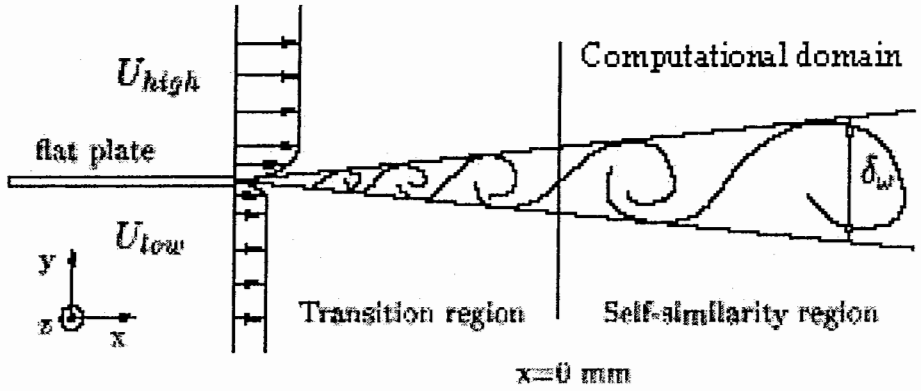
۱-۵- تحلیل جریان لایه اختلاطی

شلیختینگ [10] حل معادله لایه مرزی را برای لایه اختلاطی دو بعدی بین دو جریان با سرعتهای مختلف را به صورت تحلیلی انجام داد. وی فرض کرد که ابتدا دو جریان بصورت موازی و با سرعتهای مختلف $\lambda U, U$ در $x = 0$ با یکدیگر مخلوط می‌شوند. جریانها بدون اغتشاش در نظر گرفته شده‌اند. برای مقادیر کم لزجت ν ، انتقال مومنتوم بین سرعتها، در یک ناحیه اختلاطی باریک رخ می‌هد، بطوریکه در این لایه مولفه سرعت در جهت عمود بر جریان v ، در مقایسه با مولفه سرعت در جهت اصلی جریان u ، کوچک می‌باشد(شکل ۱-۵). بنابراین معادله لایه مرزی بدون مولفه فشار، معتبر خواهد بود. بنابراین با بی بعد سازی متغیرها، یک حل تشابهی برای این مساله وجود دارد(شکل ۱-۶).



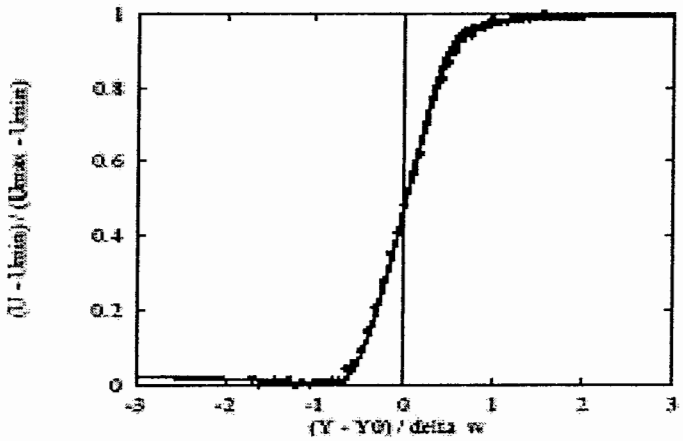
شکل ۱-۵: پروفیل سرعت در لایه اختلاطی.

پروفیل سرعت ورودی جریان لایه اختلاطی در اولین مراحل از تکامل لایه، بر روی دامنه محاسباتی اعمال می‌شود $(x = 0mm)$. در این ناحیه بدلیل توسعه یافته بودن جریان، شاهد رفتار خودتشابهی برای جریان خواهیم بود.



شکل ۱-۶: هندسه لایه اختلاطی توسعه یافته مکانی[17]

شکل۱-۷ توزیع سرعت جریان لایه اختلاطی را در دستگاه مختصات خودتشابه نشان می‌دهد. در اینجا از δ_ω و $\Delta U = U_{high} - U_{low}$ به ترتیب برای بی بعدسازی طول و سرعت استفاده شده است.



شکل ۱-۷: توزیع سرعت در جریان اختلاطی در دستگاه مختصات خود تشابه برای ایستگاههای مختلف جریان [17]

برای شبیه سازی جریان لایه اختلاطی بدون اغتشاشات ورودی، دامنه محاسباتی بصورت $0 \le x \le 50$ و $-\infty < y < +\infty$ در نظر گرفته شده‌است.

دلایل مستند زیادی در مورد تشابه پروفیل‌های سرعت متوسط جریان لایه اختلاطی در مقاطع مختلف وجود دارد. برای جریان لایه‌اختلاطی با پروفیل سرعت تانژانت هایپربولیک، وقتی‌که سرعت ها به‌وسیله اختلاف سرعت دو جریان، ΔU ، و فواصل (ابعاد) به وسیله ضخامت ورتیسیته، δ_ω ، بی بعد می‌شوند، پروفیل‌های سرعت دارای تشابه هندسی با هم می‌شوند. لازم به یادآوری است که رشد ضخامت ورتیسیته بصورت زیر تعریف می‌شود

$$\delta_\omega = \frac{\Delta U}{\left(\frac{\partial \overline{u}}{\partial y}\right)_{\max}} \qquad (۶۲-۱)$$

فصل ۲–مقدمه ای بر روش مستقیم عددی

۲-۱-مقدمه:

حل عددی کامل معادلات ناویر- استوکس وابسته به زمان ، شبیه سازی مستقیم عددی[1] نامیده می شود، که بطور کلی در آن همه پارامترهای مهم جریان را بدون استفاده از مدلهای آشفتگی محاسبه می‌کنند.

۲-۲-کاربردهای روش مستقیم عددی

شبیه سازی مستقیم عددی برای مطالعه فیزیک آشفتگی، توسعه مدلها و تئوریهای آن و در موارد خاص، برای پیش بینی و تحلیل جریان های مورد علاقه در مهندسی استفاده می گردد همچنین شبیه سازی مستقیم عددی در بررسی ساختار آشفتگی ومقایسه داده های تجربی استفاده می شود. اطلاعات بدست آمده ازحل مستقیم عددی در مدلسازی (به خصوص دراعداد رینولدز پایین) و کالیبره کردن وسایل اندازه گیری به کار گرفته می شود.

شبیه سازی مستقیم عددی در حال حاضر یکی از کاربردهای ابر کامپیوترها می باشد. مشکلات موجود در انجام این محاسبات به دلیل وجود محدوده بسیار وسیع مقیاسهای طولی وزمانی در جریان آشفته است که باید در محاسبات در نظر گرفته شوند.

برای بدست آوردن نتایج دقیق در دینامیک جریان آشفته، لازم است تغییرات جریان در مقیاسهای بسیار کوچک محاسبه شوند. بسیاری ازپدیده هایی که دریک جریان آشفته اتفاق

---

[1] DNS

می‌افتد، فرکانسهای در حدود ۴ تا ۱۰ کیلو هرتز دارند ومقیاس مکانی این تغییرات در محدوده بسیار کوچکی در حدود ۱۰ تا ۱۰۰ میکرومتر می باشد.

با درنظرگرفتن این تغییرات نیاز به شبکه محاسباتی بسیار ریز وگامهای زمانی بسیار کوتاه می باشد، که به این ترتیب احتیاج به انجام محاسبات طولانی برای این امرمشخص می شود.

انجام آزمایشات فیزیکی یکی از روشهای معمول در بدست آوردن اطلاعات لازم برای مطالعه جریان سیال است. در این میان استفاده از شبیه سازی مستقیم عددی مزیتهای بسیاری نسبت به انجام این نوع آزمایشات دارد. یکی از مهمترین مزیتها در این میان بدست آوردن اطلاعات برای مطالعه جریان در نزدیکی دیواره است . در حالیکه انجام آزمایشات و اندازه گیری پارامترهای جریان در این ناحیه بسیار مشکل و با خطا همراه است. نتایج مستقیم عددی می تواند اطلاعات کاملی ازاین پارامترها بدست آورد، ازطرف دیگر در مدلسازی آشفتگی نیاز است که اطلاعات مختلفی از جریان در یک نقطه ودر یک زمان مشخص در دست باشد، که درآزمایشات نمی توان در یک لحظه ودریک نقطه همه پارمترهای مختلف جریان را اندازه گیری کرد. مشکل دیگر عدم امکان اندازه گیری بعضی از پارامترهای موجود در مدلهاست که نمی توان برای آن، کمیت فیزیکی قابل اندازه گیری در نظر گرفت که جملات اتلاف تنش وتوزیع مجدد از این دسته می باشند.

با این وجود به دلیل محدودیتهای موجود در توان محاسباتی کامپیوترهای امروزی، محاسبات به روش مستقیم عددی تنها منحصر به بعضی جریانهای با هندسه ساده واعداد رینولدز پایین است وبنابراین استفاده از روش مستقیم عددی در جریان آشفته منحصر به مطالعه بنیادی و کاربرد در مدلسازی می باشد.

به عنوان مثال برای انجام محاسبات در شبیه سازی مستقیم عددی برای یک جریان ساده در کانال به ابعاد۱/۱۰×۱/۱۰×۱/۱۰ متر و دررینولدز بالا گردابه های به ابعاد۱۰- ۱۰۰ میکرومتر تشکیل خواهند شد و بنابراین به شبکه محاسباتی با $۱۰^{۶}$ تا $۱۰^{۱۲}$ گره احتیاج است که بتواند همه فرآیند آشفتگی جریان را مشخص کند، از طرف دیگر سریعترین تغییرات در چنین جریانی فرکانسی در مرتبه۱۰ کیلو هرتز دارد که نیاز است معادلات در بستر زمان با استفاده از قدمهای زمانی ۱۰۰ میکروثانیه گسسته سازی شوند. همچنین برای محاسبهٔ شبیه سازی مستقیم عددی

داخل لوله در عدد رینولدز ۵۰۰۰۰۰ به کامپیوتری احتیاج است که توان محاسباتی ۱۰ میلیون بار بیشتراز سریعترین کامپیوترهای Cray موجود داشته باشد بـه ایـن ترتیـب احتیـاج بـه مدلـسازی جریان آشفته و استفاده از کامپیوتردر انجام محاسبات تا سالها ادامه خواهد داشت ونیازبه تـصحیح وطراحی مدلهای بهتر همچنان احساس می شود.

فصل ۳-روش‌های تقسیم دامنه

۳-۱-مقدمه:

امروزه با پیشرفت کامپیوترها، توسعه الگوریتم‌های عددی برای حل معادلات حاکم بر حرکت سیال با استقبال زیادی روبرو گردیده است. بی شک تولید شبکه مهمترین مبحث در حل معادلات دیفرانسیل جزئی می‌باشد. بدون داشتن یک شبکه مناسب رسیدن به یک حل قابل قبول امکان‌پذیر نخواهد بود. بنابراین اولین قدم در حل معادلات دیفرانسیل حاکم بر فیزیک مسئله، این است که هندسه مورد نظر به درستی شبکه‌بندی گردد. معادلات کلی حاکم بر حرکت سیال، معادلات ناویر- استوکس می‌باشند. از نقطه‌نظر ریاضی، این معادلات جزء معادلات غیر خطی طبقه‌بندی می‌شوند که حل تحلیلی آنها موجود نمی‌باشد. یکی از راه‌های حل این معادلات استفاده از روش‌های عددی می‌باشد. به منظور حل عددی میدان جریان در هندسه‌های پیچیده، هم می‌توان از شبکه‌های محاسباتی بی‌سازمان[1] و هم از شبکه‌های باسازمان[2] با تمهیدات خاص استفاده نمود. چنانچه بتوان با استفاده از روش‌های شبکه‌بندی باسازمان شبکه عددی مناسب را تولید کرد، استفاده از آنها به جای روش‌های بی‌سازمان ترجیح داده می‌شود. در شبکه‌های باسازمان معمولاً از سه روش زیر جهت تولید شبکه محاسباتی در دامنه هندسی استفاده می‌شود که هر کدام از این روش‌ها معایب و مزایای خاص خود را دارا می‌باشند:

---

[1] Unstructured Grid
[2] Structured Grid

روش تک بلوکی یا تک ناحیه‌ای[۳]

روش چند ناحیه‌ای[۴]

روش چند بلوکی[۵]

که در ادامه به توضیح هر یک پرداخته می‌شود.

۳-۲-شبکه بندی

برای آن‌که بتوانیم یک میدان فیزیکی مورد نظر را به درستی مدل کنیم باید یک شبکه مناسب روی میدان تولید کنیم. روش‌های مختلفی برای تولید شبکه وجود دارد:

۱. شبکه‌های باسازمان یا منظم

۲. شبکه‌های بی‌سازمان یا نامنظم

۳. شبکه‌های ترکیبی[۶]

هر کدام از این شبکه‌ها دارای معایب و محاسن خاص خود بوده و انتخاب هر کدام از آنها حل جریان را تحت تأثیر قرار می‌دهد.

تولید شبکه باسازمان تک بلوکی باعث می‌شود تا الگوریتم حل جریان بسیار راحت گردد. اما این روش از مشکلات خاصی برخوردار است. به‌عنوان مثال اگر قرار باشد قسمتی از حل با شبکه ریز انجام دهیم ناگزیریم تا تمام میدان را با شبکه ریز مدل کنیم در حالی که ممکن است نیاز به چنین شبکه ریزی در سراسر میدان نداشته باشیم. همچنین در حالاتی که هندسه جسم پیچیده می‌گردد، تولید شبکه باسازمان به سختی صورت می‌گیرد و حتی گاهی عملاً تولید شبکه باسازمان امکان‌پذیر نخواهد بود.

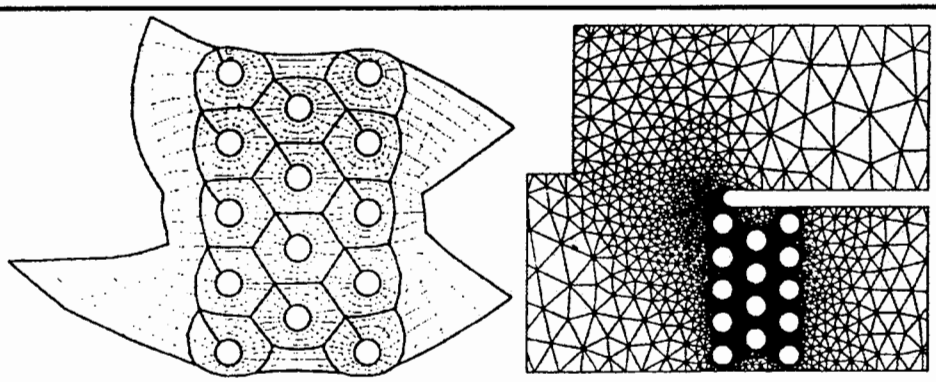
---

[3] Single-Block/Zone
[4] Multi-Zone
[5] Multi-Block
[6] Hybrid Grid

شبکه بی‌سازمان بهترین روش برای تولید نقاط در دامنه محاسباتی می‌باشد. این روش بسیار انعطاف‌پذیر بوده و به راحتی می‌توان در جایی که نیاز است از شبکه ریز و در جایی که نیاز نیست از شبکه درشت استفاده کرد. اما تولید شبکه بی‌سازمان باعث می‌شود تا در جایی که جریان غیرهمگن داریم مانند لایه مرزی، حل از دقت کافی برخوردار نباشد. درثانی توسعه روش‌های مرتبه بالا روی شبکه‌های بی‌سازمان کار دشواری می‌باشد. همچنین شبکه بی‌سازمان به معنی بالا رفتن حافظه مورد نیاز، بالا رفتن پیچیدگی ساختار داده‌ای و مشکلات دیگری از قبیل سخت بودن اعمال روش‌هایی مانند چند شبکه‌ای و دیگر الگوریتم‌های پیشرفته می‌باشد. در حال حاضر، روش ترکیبی که از مزایای هر دو روش باسازمان و بی‌سازمان به طور هم زمان سود می‌برد مورد توجه محققان قرار گرفته است.

شبکه های بلوکی می‌توانند ترکیبی از شبکه‌های باسازمان و بی‌سازمان باشند. بدین صورت که در هر بلوک به صورت محلی از شبکه باسازمان یا بی‌سازمان استفاده می‌شود، اما خود بلوک‌ها می‌توانند به صورت بی‌سازمان در کل میدان پخش شده و آن را بپوشانند. این خاصیت باعث افزایش کاربرد و توانایی روش شبکه‌های بلوکی شده و استفاده از آنها را مورد توجه محققان قرار داده است.

در شکل(۳-۱) تولید شبکه بی‌سازمان و باسازمان بلوکی روی تعدادی استوانه نمایش داده شده است. این اشکال نشان می‌دهند که هر دو روش قادر به تولید شبکه روی مرزهای پیچیده می‌باشند و می‌توان ازآنها برای تولید هندسه‌های پیچیده استفاده کرد.



الف ب

شکل ۳-۱: تولید شبکه روی هندسه‌های پیچیده با استفاده از روش‌های، الف: باسازمان،ب: بی‌سازمان.

۳-۳- روش تک بلوکی یا تک ناحیه‌ای

اولین قدم برای حل معادلات دیفرانسیل حاکم بر فیزیک مسئله، شبکه‌بندی درست هندسه می‌باشد. بنابراین اگر هندسه کمی پیچیده‌تر شود یکی از روش‌های شبکه‌بندی دامنه هندسی جهت حل عددی معادلات حاکم بر حرکت سیال و تحلیل عددی جریان، روش تک بلوکی یا تک ناحیه‌ای است. از مزایای این روش، ساده بودن الگوریتم برنامه‌نویسی آن برای هندسه‌های ساده می‌باشد، با توجه به دشوار بودن تولید شبکه به صورت تک بلوکی برای هندسه‌های پیچیده، این روش مناسب نبوده و هندسه به خوبی مدل نمی‌شود. برای حل عددی از دو روش صریح[۷] و ضمنی[۸] استفاده می‌شود. دقت و پایداری حل روش‌های ضمنی نسبت به روش‌های صریح که روش حل آن نقطه به نقطه و پیشروی به جلو است، بیشتر می‌باشد.

---

[7] Explicit
[8] Implicit

۳-٤- روش چند ناحیه‌ای

یکی دیگر از روش‌های تقسیم دامنه هندسی به منظور حل عددی معادلات حاکم بر حرکت سیال وتحلیل عددی جریان، روش چند ناحیه‌ای است. در روش چند ناحیه‌ای با توجه به این که هندسه به نواحی ساده تقسیم‌بندی شده و در هر ناحیه به طور مستقل و با روش دلخواه شبکه‌بندی صورت می‌گیرد. همچنین چون حل معادلات حاکم در هر ناحیه به طور مستقل صورت می‌گیرد، بنابراین این روش فقط برای جریان‌های مافوق‌صوت کاربرد دارد که سیال از وجود اطلاعات پایین‌دست جریان بی‌اطلاع می‌باشد.

شبکه‌های چند ناحیه‌ای، فرآیند تولید شبکه باسازمان و تحلیل عددی جریان حول هندسه‌های پیچیده را میسر و روان می‌سازد. در این نوع شبکه‌بندی مرزهای جدیدی بین نواحی همسایه بنام مرز ناحیه‌ای[۹] به وجود می‌آید. حل مرز ناحیه‌ای محتاج یک روش درونیابی پردقت، پایدار و قابل تعمیم به سیستم‌های مختصات منحنی‌الخط است.روشی مطلوب با خواص فوق توسط «رآی» برای هندسه‌های دوبعدی ابداع شده است. بکار گیری ایده‌های چند ناحیه‌ای موجب تقویت و توسعه کاربرد شبکه‌های باسازمان شده است. شبکه‌های چند ناحیه‌ای اغلب به دو دسته شبکه‌های انطباقی و شبکه‌های وصله‌ای تقسیم می‌شوند که دسته اول نیاز به درون‌یابی ندارد ولی برای دسته دوم می‌توان اگر تعداد نقاط مرزی یکی نباشد از درون‌یابی استفاده کرد که در ادامه به توضیح هر یک از این دو دسته پرداخته می‌شود.

۳-۵- روش چندبلوکی

تاکنون الگوریتم‌های متعددی به منظور تولید شبکه وتقسیم میدان حل برای تحلیل جریان به صورت عددی ارائه و به انجام رسیده است که هر کدام از آنها دارای قابلیت‌های مخصوص خود و همچنین معایب خاص خود می‌باشند. از میان تمام این الگوریتم‌ها، بی‌شک یکی از

---

[۹] Zone Boundary

کاربردی‌ترین آنها، روش چندبلوکی می‌باشد. این روش که امروزه با استقبال بسیار زیادی روبرو شده است، از انعطاف‌پذیرترین الگوریتم‌ها بوده و با اعمال آن، قابلیت برنامه‌هایی که به منظور شبیه سازی میدان جریان به کار می‌روند به نحو چشمگیری افزایش می‌یابد تا به این حد که اعمال روش چندبلوکی، باعث می‌شود تا این برنامه‌ها از حالت تحقیقاتی صرف به یک برنامه کاملاً کاربردی و صنعتی تبدیل گردد. مشاهده مقالات ارائه شده، مشخص می‌سازد که تمام محققان درصدد دستیابی به الگوریتم‌های چند بلوکی می‌باشند، به‌خصوص مقالاتی که در آنها تحلیل جریان روی هندسه‌های حقیقی و صنعتی صورت گرفته، بدون شک دارای الگوریتم چند بلوکی می‌باشند. در این روش، هندسه به چندین بلوک کوچک‌تر و ساده تقسیم می‌گردد. این نواحی طوری انتخاب می‌گردند که حتی‌المقدور ساده‌ترین شکل هندسی را داشته باشند. انتخاب این بلوک‌ها دست برنامه‌نویس است. بعد از تقسیم دامنه هندسه به قسمت‌های ساده، تولید شبکه سطحی روی این بلوک‌های ساده صورت می‌گیرد و حل میدان جریان ساده در هر بلوک به طور جداگانه انجام می‌گردد. از آنجا که هندسه کل، از مجموعه تمام این بلوک‌ها تشکیل شده است، لازم است تا تبادل اطلاعات هر بلوک با بلوک‌های مجاور صورت گیرد. به طور کلی می‌توان جسم را به بلوک‌های عرضی و طولی تقسیم نمود و این تقسیم‌بندی کاملاً دلخواه می‌باشد. استقبال از روش‌های چند بلوکی از جهات مختلفی قابل بررسی می‌باشدکه در ذیل به ذکر برخی از این دلایل پرداخته می‌شود.

۳-۵-۱- نیاز به حافظه کافی برای جریان روی هندسه کاربردی

در تحلیل جریان روی هندسه حقیقی و کاربردی، می‌بایست شبکه‌ای با ابعاد بزرگ و نقاط بسیار ریز تولید شود. این موضوع سبب می‌شود تا شبکه تولیدی بسیار بزرگ بوده به طوری که برای حل جریان در آن با کمبود حافظه مواجه خواهیم شد. جهت تحلیل جریان حول هندسه‌های پیچیده، عملاً حافظه کافی برای محاسبات کامپیوتر در دسترس نخواهد بود. اولین انگیزه که باعث

شد روش‌های چند بلوکی مورد مطالعه قرار گیرد، غلبه بر مشکل فوق بود. بدین‌گونه که با تجزیه هندسه به بلوک‌های مناسب، در هر لحظه تنها به حل یک بلوک پرداخته شده و اطلاعات مربوط به بلوک‌های دیگر در حافظه جانبی ذخیره می‌شود. گرچه این موضوع باعث می‌شود تا سرعت اجرای برنامه کم شده، ولی به عنوان تنها راه غلبه بر مشکل حافظه، بسیار سودمند می‌باشد.

۳-۵-۲- تولید شبکه روی هندسه‌های پیچیده

از لحاظ تاریخی می‌توان گفت که اولین الگوریتم‌های عددی برای حل جریان به کمک روش‌های تفاضل محدود صورت گرفته است. برای این که به روش تفاضل محدود بتوان جریان درون یک دامنه هندسی را مدل کرد، نیاز است تا یک شبکه منظم روی بدنه و حول جسم تولید نمود. همین مسئله یکی از بزرگترین مشکلات موجود برای حل جریان می‌باشد. هنگامی که هندسه پیچیده می‌شود، مخصوصاً در حالت‌هایی که جسم سه بعدی می‌باشد، تولید شبکه منظم یا باسازمان روی بدنه و همچنینّ تولید شبکه سه بعدی روی میدان به سختی صورت می‌گیرد.

بر خلاف شبکه‌های منظم، شبکه‌های نامنظم یا بی‌سازمان پاسخ‌گوی این مسئله می‌باشند. شبکه‌های نامنظم دارای این خاصیت هستند که به کمک آنها می‌توان هر گونه هندسه سه‌بعدی را توسط آنها شبکه‌بندی کرد. مشکل اصلی این است که باکمک این شبکه‌ها دیگر نمی‌توان از الگوریتم‌های تفاضل محدود استفاده کرد. برای این منظور الگوریتم‌های حل معادلات سیال به کمک روش‌های المان محدود و حجم محدود تدوین و توسعه یافته‌اند. به کمک این الگوریتم‌ها و شبکه نامنظم می‌توان هر هندسه‌ای را شبکه‌بندی و حل نمود، ولی در عمل مشاهده شده است که استفاده از شبکه‌های منظم حتی هنگامی که از الگوریتم‌های حجم محدود و المان محدود استفاده می‌شود، از دقت بیشتری برخوردارند. لذا همچنان سعی می‌گردد تا بتوان روش‌های مناسبی با

تولید شبکه منظم روی هندسه‌های پیچیده و سه بعدی به دست آورد. بی‌شک تنها راه غلبه بر مشکل موجود، استفاده از روش چند بلوکی می‌باشد.

۳-۵-۳- حل حاصل از اندرکنش جریان خارجی با جریان داخلی

در حالاتی که بخواهیم تاثیر جریان داخلی را روی جریان خارجی حساب کنیم، ناگزیریم تا از روش چندبلوکی استفاده‌کنیم. به عنوان مثال جریان روی هواپیما را در نظر بگیرید. برای آنکه تاثیر نیروی ناشی از موتور جت را روی ضرایب آیرودینامیکی هواپیما به دست بیاوریم، ناگزیریم تا جریان داخل نازل موتور را که یک جریان داخلی است را با یک بلوک حل کرده ونتایج آن را به شبکه حول هواپیما داده و تاثیر آن را روی ضرایب هواپیما به دست بیاوریم.

۳-۵-۴- حل جریان در مسائلی که دارای مقیاس‌های طولی متفاوت می‌باشند

در بسیاری از مسائل فیزیکی حقیقی، به مقیاس‌های طولی متفاوتی برخورد می‌کنیم که از جمله آنها می‌توان به جریان‌های با سرعت زیاد، با گرادیان بالا و وجود گردابه‌ها اشاره کرد. در چنین مسائلی لازم است تا در هر کجا که مقیاس‌ها عوض می‌شوند شبکه محاسباتی را فشرده یا در صورت نیاز از شبکه درشت‌تری استفاده کرد. تولید چنین شبکه‌ای به وسیله شبکه باسازمان تک‌بلوکی کاری بسیار سخت و پر هزینه می‌باشد و اغلب با تقریب‌های زیادی همراه است. این دلیل نیز اهمیت روش چندبلوکی را بالا می‌برد و کارهای متعددی توسط محققان مختلف روی آن انجام شد. همانطور که گفته شد در این روش، هندسه مورد نظر به چندین بلوک تقسیم می‌گردد به گونه‌ای که مجموع بلوک‌ها، کل هندسه را تشکیل می‌دهند و در هر بلوک به طور جداگانه می‌توان شبکه‌بندی مورد نیاز را انجام داد به طوری که بسته به نیاز هر ناحیه بتوان مقیاس‌های طولی مورد نیاز را ارضا کرد.

۳-۵-۵- امکان پردازش موازی

در دهه گذشته محاسبات زیاد و طولانی تنها روی ابر کامپیوترها امکان‌پذیر بود. که این به نوبه خود مساله بسیار بزرگی محسوب می‌گردد زیرا ابر کامپیوترها دارای هزینه‌های بسیار زیادی می‌باشند. امروزه پردازش موازی به عنوان یکی از مهمترین ابزارها برای تحلیل جریان مطرح شده است، تا چندی پیش به علت ضعف کامپیوترها و وسایل ارتباط اطلاعات، امکان موازی‌سازی کامپیوترهای شخصی فراهم نبود. این مشکل با توسعه سخت افزارها و امکان تبادل سریع اطلاعات بین کامپیوترهای شخصی مرتفع گردیده است. یکی دیگر از مزایای اصلی روش چندبلوکی، قابلیت ذاتی این روش برای پردازش موازی می‌باشد. بدین‌گونه کافی است که هر کدام از بلوک‌های ایجاد شده، روی یک کامپیوتر جداگانه تحلیل شوند و اطلاعات مورد نیاز در روی مرزها بین کامپیوترها منتقل شوند.

۳-۵-۶- مشکلات موجود در زمینه روش چند بلوکی

۱. پیچیده‌تر شدن الگوریتم حل عددی.

۲. تبادل اطلاعات بین بلوک‌ها، بزرگترین مساله در روش چند بلوکی می‌باشد. روش‌های موجود در تبادل اطلاعات به شرح زیر می‌باشد:

الف- در بلوک‌های منطبق: این کار با جایگزینی مقادیر نقاط مرزی از بلوک همسایه به بلوک حاضر امکان‌پذیر است(عدم نیاز به میان‌یابی).

ب- استفاده از میان‌یابی خطی: در جاهایی که گرادیان متغیرهای جریان کم است و ناپیوستگی‌هایی نظیر موج ضربه‌ای، برگشت جریان و ... وجود ندارد.

ج- استفاده از روش میان‌یابی با دقت بالاتر: در جاهایی که گرادیان متغیرهای جریان زیاد است، نظیر لایه مرزی و موج ضربه‌ای، کارآمد می‌باشد.

د- استفاده از میان‌یابی بقایی[10] در جاهایی که ناپیوستگی‌هایی نظیر موج ضربه‌ای، برگشت جریان و ... داریم: میان‌یابی بقایی به منظور ارضاء کردن معادلات بقاء در مرزها می‌باشند. مثلاً با مساوی قرار دادن جرم، اندازه حرکت و انرژی عبوری از یک بلوک به بلوک دیگر مقادیر متغیرهای جریان روی مرزها به‌دست می‌آید.

۳-۵-۷- بلوک‌بندی

بلوک‌بندی مناسب میدان جریان از اهمیت بسیاری برخودار می‌باشد و بیشترین زمان لازم برای تحلیل عددی جریان حول اجسام پیچیده، صرف ایجاد شبکه می‌شود. در یکی از روشهای خودکار، ابتدا میدان جریان را به صورت بی‌سازمان [11] بلوک‌بندی کرده سپس در هر بلوک شبکه باسازمان ایجاد می‌نمایند. هر چند که روش موثر برای حل میدان جریان روی هندسه‌های پیچیده، استفاده از شبکه بی‌سازمان تک‌بلوکی است، ولی ترکیب بلوک‌بندی جریان با استفاده از شبکه‌های باسازمان جانشین مناسب‌تری برای شبکه‌های بی‌سازمان می‌باشد. در دهه اخیر تحقیقات گسترده‌ای در زمینه تولید و استفاده از شبکه‌های چندبلوکی، روش‌های تولید شبکه، روش‌های دسته‌بندی اطلاعات و همچنین تهیه نرم‌افزارهای تولید شبکه، برای کاهش نقش کاربر و زمان تولید شبکه بویژه برای شکل‌های پیچیده هندسی انجام گرفته‌است.

به طور کلی دو نوع شبکه بلوکی وجود دارد:

۱. شبکه‌های منطبق[12]

۲. شبکه‌های غیرمنطبق[13] که خود به سه دسته زیر تقسیم می‌شود:

شبکه‌های وصله‌ای[14]

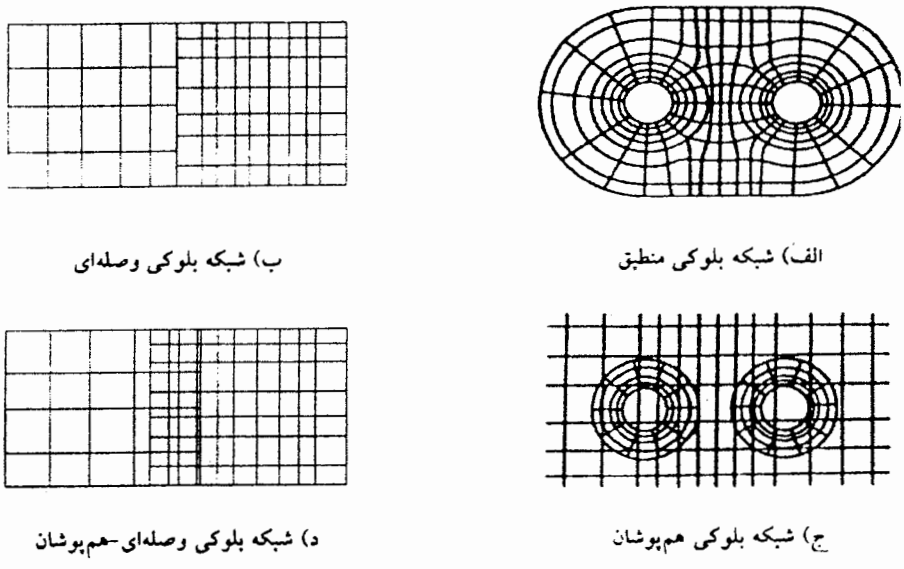
---

[10] Conservative Interpolation
[11] Unstructured
[12] Matched Grids
[13] Unmatched Grids

شبکه‌های هم‌پوشان[۱۵]

شبکه‌های وصله‌ای-هم‌پوشان[۱۶]

شکل(۳-۲) انواع مختلف شبکه‌های بلوکی را نشان می‌دهد.



الف) شبکه بلوکی منطبق

ب) شبکه بلوکی وصله‌ای

ج) شبکه بلوکی هم‌پوشان

د) شبکه بلوکی وصله‌ای-هم‌پوشان

شکل۳-۲ : نمایش انواع مختلف شبکه‌های بلوکی.

۳-۵-۷-۱- روش انطباقی

در شبکه‌های منطبق بلوک‌های مجاور در مرز و نقاط مرزی با یکدیگر مشترک می باشند و هیچ‌گونه هم‌پوشانی ندارند. این نوع شبکه‌ها در مسائل هوافضایی و توربوماشین بسیار سودمند می‌باشند. در این روش در مقایسه با روش‌های شبکه‌بندی باسازمان و شبکه‌بندی بی‌سازمان، هندسه براحتی مدل می‌گردد و قوانین بقایی را در مرزها براحتی می‌توان اعمال کرد. اما این روش دارای معایب زیر می‌باشد:

---

[14] Patched Grids
[15] Chimera Grids
[16] Patched- Chimera Grids

۱. از آنجا که بلوک‌ها در مرز و نقاط مرزی با یکدیگر مشترک می‌باشند، انعطاف‌پذیری این روش در تولید هندسه‌های پیچیده کم می‌شود. این قید باعث می‌شود تا بلوک‌ها هم در تعداد نقاط به یکدیگر وابسته بوده و هم در مکان نقاط روی مرز دارای قید باشند. این قید ممکن است به پایین آمدن کیفیت شبکه منتهی شود و حتی در مرزها شرط تعامد شبکه را نتوان به درستی اعمال کرد.

۲. از آنجا که تعداد بلوک‌ها به بلوک‌های مجاور وابسته می‌باشد، باید از الگوریتم مناسبی جهت تعیین تعداد نقاط مناسب در هر بلوک تعیین گردد تا در مناطق مورد نیاز فشردگی شبکه رعایت شود.

۳-۵-۷-۲- روش غیر انطباقی

برخلاف شبکه‌های منطبق، شبکه‌های غیرمنطبق از انعطاف‌پذیری بیشتری برخوردارند. شبکه‌های غیرمنطبق را عموماً به دو دسته تقسیم‌بندی می‌کنند:

۱. شبکه‌های وصله‌ای

شبکه‌های وصله‌ای شبکه‌های هستند که در آنها دو بلوک مجاور تنها در یک مرز با یکدیگر مشترک هستند و نقاط مرزی آنها لزوماً روی یکدیگر قرار ندارند.

۲. شبکه‌های هم‌پوشان

شبکه‌های هم‌پوشان شبکه‌هایی هستند که در آنها دو بلوک مجاور هم، با یکدیگر هیچ ارتباط هندسی ندارند و کاملاً آزادانه می‌توانند روی یکدیگر حرکت کنند. بنا به این خاصیت، با استفاده از شبکه‌های هم‌پوشان می‌توان به راحتی هرگونه هندسه را شبکه‌بندی کرده و جریان را روی آنها حل نمود. اما انتقال اطلاعات بین بلوک‌ها به سختی صورت می‌گیرد و ارضای شارها در نقاط مرزی بسیار مشکل خواهد بود.

اگرچه فعالیت‌های زیادی روی شبکه‌های غیر منطبق صورت گرفته‌است، اما همچنان مشکلات زیادی وجود دارد که از جمله می‌توان به انتقال اطلاعات بین بلوک‌ها، تولید شبکه به صورت اتوماتیک و مسائل با مرز متحرک اشاره کرد. در شکل(۳-۳- الف) تا (۳-۳- د) انواع شبکه‌های تولید شده توسط روش‌های مختلف مشاهده می‌شود. شکل(۳-۳- الف) میدان حل را به‌وسیله یک بلوک باسازمان شبکه‌بندی کرده است. همانطور که ملاحظه می‌شود این شبکه از کیفیت مناسب برخوردار نمی‌باشد. شکل(۳-۳- ب) همان میدان را به‌وسیله بلوک‌های منطبق شبکه‌بندی کرده‌است. همان‌طور که در این شکل ملاحظه می‌شود، بلوک‌ها هم در مرز و هم در نقاط مرزی با یکدیگر مشترک هستند. شکل (۳-۳ ج) همان میدان را به‌وسیله بلوک‌های وصله‌ای شبکه‌بندی کرده‌است. در این حالت بلوک‌ها تنها در مرزها با یکدیگر در اشتراکند و نقاط مرزی به صورت دلخواه روی مرزها تقسیم شده‌اند. همان‌طور که ملاحظه می‌شود تعداد نقاط در هر بلوک را می‌توان به طور دلخواه تعیین کرد و محدودیت حالت قبلی وجود ندارد. شکل(۳-۳- د) همان میدان را به وسیله بلوک‌های هم‌پوشان شبکه‌بندی کرده است. در این شکل ملاحظه می‌شود که بلوک‌ها به دلخواه یکدیگر قرار گرفته‌اند و هیچ‌گونه محدودیتی ندارند. هر بلوک به طور دلخواه شبکه‌بندی شده و حتی می‌توان هر بلوک را در دستگاه مختصات مناسب شبکه‌بندی و حل نمود.

در حالت کلی، بلوک‌های هم‌پوشان از انعطاف‌پذیری بیشتری نسبت به بلوک‌های وصله‌ای برخوردارند. گرچه این بلوک‌ها نیز دارای معایبی نسبت به بلوک‌های وصله‌ای می‌باشند. بعضی از معایب بلوک‌های هم پوشان عبارتند از:

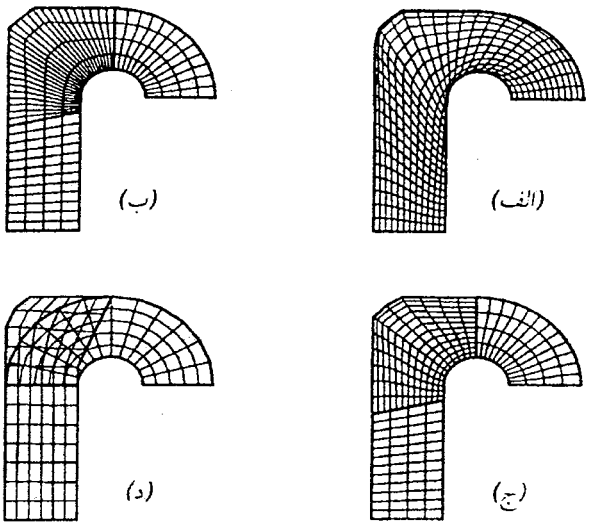
۱. در یک مساله n بعدی برای انتقال اطلاعات از یک بلوک به بلوک دیگر نیازمند درون‌یابی n بعدی هستیم در حالی که در حالت بلوک‌های وصله‌ای نیازمند درون‌یابی (n-1) بعدی خواهیم بود. این مساله هم در سرعت و هم در دقت شرایط مرزی بلوکی تاثیر می‌گذارد.

۲. ارضای معادلات بقایی در حالت بلوک‌های هم‌پوشان بسیار سخت می‌باشد.

۳. دقت و سرعت همگرایی محاسبات در این حالت بستگی به میزان هم‌پوشانی دارد.

۴. اندازه نسبی بلوک‌ها نیز در دقت و سرعت همگرایی موثر است.

اما به هر حال انعطاف‌پذیری روش هم‌پوشان در تولید شبکه بویژه در حالت سه بعدی باعث شده است تا این بلوک‌ها مورد توجه قرار گیرند. همچنین در مسائلی مانند جریان غیر دایم و یا جریانی که در آن حرکت نسبی بین هندسه‌های مختلف وجود دارد و مرز متحرک است، کارآیی بلوک‌های هم‌پوشان بیشتر است.

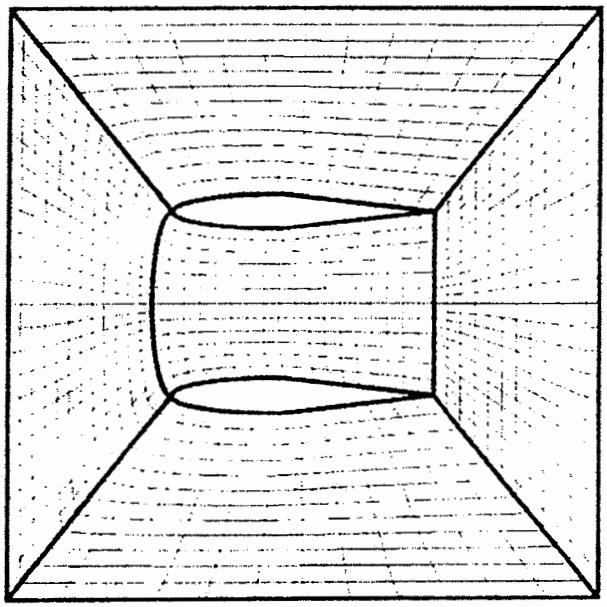


شکل۳-۳: تولید شبکه سازمان یافته به روش‌های مختلف، الف: شبکه باسازمان، ب: شبکه بلوکی منطبق، ج: شبکه بلوکی وصله‌ای، د: شبکه بلوکی هم‌پوشان .

۳-۵-۸- مرزهای بلوکی

اولین قدم در حل به روش چند بلوکی، شکستن میدان حل به بلوک‌های مختلف می‌باشد. این تقسیم‌بندی باعث می‌شود که در میدان حل مرزهایی به وجود بیاید که به نام مرزهای بلوکی معروفند. شکل(۳-۴) میدان حل بین دو ایرفویل را نشان می‌دهد که به بلوک‌های مختلف تقسیم

شده است. تولید شبکه تک بلوکی در این ناحیه که یک ناحیه همبند چند گانه است بسیار مشکل می‌باشد. به همین منظور، ناحیه مورد نظر را به پنج بلوک تقسیم شده است. ملاحظه می‌شود که هر کدام از این نواحی یک ناحیه همبند ساده با چهار مرز ساده می‌باشند و به راحتی می‌توان هر بلوک را شبکه‌بندی کرد. در این شکل، مرزهای بلوکی به طور مشخص می‌باشند که در فصل مشترک دو بلوک ایجاد شده‌اند. این مرزها باعث می‌شوند تا حل جریان به‌راحتیِ هنگامی که با یک بلوک سروکار داریم، نباشد.

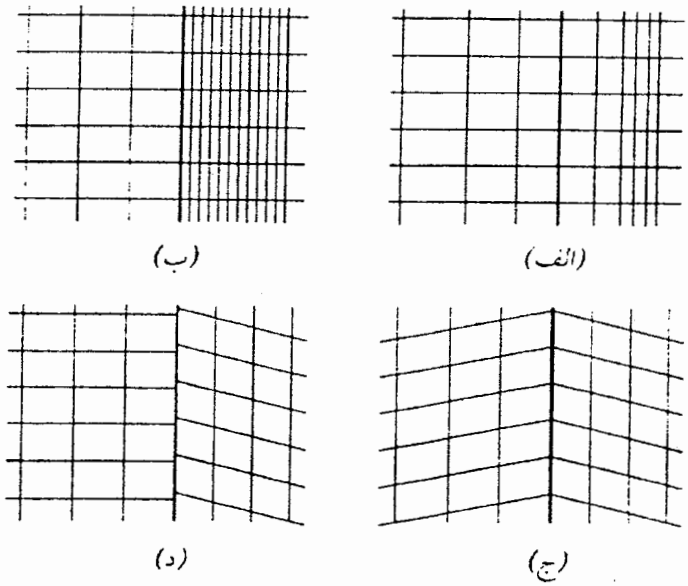


شکل۳-۴: تقسیم ناحیه ما بین دو ایرفویل به پنج بلوک.

محاسبه مقادیر روی مرزهای بلوکی باید طوری صورت گیرد که:

- حل پایدار باشد.
- روی مختصات منحنی‌الخط قابل اعمال باشد.

از آنجا که هر بلوک به طور جداگانه شبکه بندی می‌شود، خطوط شبکه در دو بلوک مجاور می‌توانند در یک راستا باشند (شبکه‌های منطبق) یا اینکه این خطوط در یک راستا نباشند(شبکه‌های غیر منطبق). حتی در شبکه‌های منطبق نیز تغییر ناگهانی در فواصل شبکه بین دو بلوک مجاور می‌تواند باعث پرش در متریک‌ها شود(شبکه‌های با متریک ناپیوسته). در شکل (۳-۵) انواع مختلف این شبکه‌ها دیده می‌شود. برای آن که اطلاعات از یک بلوک به بلوک دیگر به خوبی و با دقت منتقل شود، لازم است تا تبادل اطلاعات روی مرزها با دقت صورت گیرد. برای این عمل نیاز به روش‌های بادقت بالای درونیابی احساس می‌گردد.



شکل۳-۵: انواع مرزهای بلوکی. الف: مرز با پیوستگی شبکه، ب: مرز با پیوستگی متریک،ج: مرز با ناپیوستگی متریک، د: مرز با ناپیوستگی شبکه

فصل ۴-محاسبه مشتقات جزئی به روش عددی

٤-١-مقدمه

در این فصل نحوهٔ محاسبهٔ مشتقات جزئی به روش عددی بحث خواهد شد . روش محاسبهٔ این مشتقات در مختصات مکانی روش تفاضلات محدود فشرده است که در همین فصل نحوه محاسبهٔ آنها توضیح داده می‌شود. این روش از دقت بالایی برخوردار است و می توان برای محاسبه مشتقات از هر مرتبه ای از آن استفاده کرد. برای محاسبهٔ مشتق زمانی از روش رانگ – کوتا استفاده می شود . در انتهای فصل مشتق چند تابع ریاضی به روش تفاضلات محدود فشرده و به صورت عددی محاسبه شده است و نتایج آن با روش تحلیلی مقایسه شده است.

٤-٢-روش تفاضلات محدود فشرده برای محاسبهٔ مشتقات

مشتقات جزئی هم در جهت جریان و هم عمود بر جهت جریان از روش تفاضلات محدود فشرده محاسبه شده است این روش توسط Lele [3] ارائه شده است. در این روش مشتق اول تابعی دلخواه مانند f(x) به صورت زیر محاسبه می‌شود:

$$\alpha f'_{j-1} + f'_j + \alpha f'_{j+1} = \frac{\alpha+2}{3\Delta x}(f_{j+1} - f_{j-1}) + \frac{4\alpha-1}{12\Delta x}(f_{j+2} - f_{j-2}) \qquad (۱-۴)$$

که در رابطهٔ بالا علامت پریم نمایانگر مشتق مرتبهٔ اول می‌باشد و j نشان‌دهندهٔ شمارهٔ شبکهٔ مورد نظر می‌باشد ($1 \le j \le J$) و $\Delta x = L_x/(J-1)$. با جاگذاری $\alpha = 1/3$ و $\alpha = 1/4$ به‌ترتیب دقت

مرتبهٔ شش و چهاربه دست می‌آید. هنگامی که $\alpha = 1/4$ ویا $\alpha = 1/3$ باشد قطر اصلی در طرف راست معادلهٔ بالا چهار یا سه برابر کوچکتر از دو قسمت دیگر هستند، که این موضوع بیانگر این است که معادلهٔ بالا شرایط ناهنجاری دارد . یک راه‌حل برای رفع این مشکل این است که دو طرف رابطهٔ (۴-۱) را در $1/\alpha$ ضرب کنیم. این کار باعث می‌شود رابطهٔ(۴-۱) به صورت رابطهٔ (۴-۲) تغییر یابد:

$$f'_{j-1} + \frac{1}{\alpha} f'_j + f'_{j+1} = \frac{1+\frac{2}{\alpha}}{3\Delta x}(f_{j+1} - f_{j-1}) + \frac{4-\frac{1}{\alpha}}{12\Delta x}(f_{j+2} - f_{j-2}) \qquad (۴-۲)$$

در مرزهای جریان(یعنی در جائیکه j=1 و j=J) باشد،روش ضمنی یک طرفهٔ درجهٔ سه برای تقریب مشتق اول به کار برده می‌شود که به صورت زیر است:

$$f'_1 + 2f'_2 = \frac{1}{2\Delta x}(-5f_0 + 4f_1 + f_2) \qquad (۴-۳)$$

و

$$f'_j + 2f'_{j-1} = \frac{1}{2\Delta x}(5f_j - 4f_{j-1} - f_{j-2}) \qquad (۴-۴)$$

در نزدیکی مرزها (یعنی j=2 و j=J-1) شکل کلی معادله به کار برده می‌شود با این تفاوت که $\alpha = 1/4$. Lele [3]پیشنهادی مطرح می‌کند مبنی بر اینکه در j=3 و j=J-3 تغییراتی را انجام دهیم به طوریکه α را با α' جاگذاری کنیم که α' به صورت زیر تعریف می‌شود.

$$\alpha' = (16\alpha + 32)/(40\alpha - 1) \qquad (۴-۵)$$

این عمل به این دلیل است که این جاگذاری پایداری حل عددی را در مواردی که معادله به صورت $(\partial/\partial t)u = (\partial/\partial x)f(u)$ است ، بیشتر می کند.

معادلهٔ زیر نمایانگر مشتق مرتبهٔ دوم تابعی دلخواه مانند f(x) است که در آن از مرتبهٔ چهارم شکل تفاضلات محدود فشرده استفاده شده است:

$$f''_{j-1} + f''_j + \alpha f''_{j+1} = \frac{4(1-\alpha)}{3\Delta x^2}(f_{j-1} - 2f_j + f_{j+1}) + \frac{10\alpha - 1}{12\Delta x^2}(f_{j-2} - 2f_j + f_{j-2}) \qquad (۴-۶)$$

که در آن $\alpha = \frac{1}{4}$. در اینجا نیز برای غلبه بر شرایط ناهنجار هر دو طرف رابطهٔ (۴-۶) را در $1/\alpha$ ضرب می‌کنیم که در نتیجه رابطهٔ (۴-۷) حاصل خواهد شد.

$$f''_{j-1}+\frac{1}{\alpha}f''_j+f''_{j+1}=\frac{4(\frac{1}{\alpha}-1)}{3\Delta x^2}(f_{j-1}-2f_j+f_{j+1})+\frac{10-\frac{1}{\alpha}}{12\Delta x^2}(f_{j-2}-2f_j+f_{j-2}) \qquad (۷-۴)$$

در مرزهای جریان، حالت مرتبهٔ سوم یک طرفه به کار برده می‌شود که به صورت زیر است:

$$f''_1+11f''_2=\frac{1}{\Delta x^2}(13f_1-27f_2+15f_3-f_4) \qquad (۸-۴)$$

و

$$f''_j+11f''_{j-1}=\frac{1}{\Delta x^2}(13f_j-27f_{j-1}+15f_{j-2}-f_{j-3}) \qquad (۹-۴)$$

با مشتق گیری از طرفین رابطهٔ (۴-۳) داریم:

$$f''_1+2f''_2=\frac{1}{2\Delta x}(-5f'_1+4f'_2+f'_3) \qquad (۱۰-۴)$$

که به راحتی می‌توان نشان داد معادل رابطهٔ زیر است :

$$f''_1+2f''_2=\frac{-3}{\Delta x}f'_1+\frac{1}{2\Delta x}(f'_1+4f'_2+f'_3) \qquad (۱۱-۴)$$

با جاگذاری طرف چپ رابطهٔ (۴-۳) (با فرض $\alpha=1/4$) برای ترمهای داخل پارانتز در رابطهٔ (۴-۱۱) به رابطهٔ زیر می‌رسیم :

$$f''_1+2f''_2=\frac{-3}{\Delta x}\frac{df}{dx}|_{(x=0)}-\frac{3}{2\Delta x^2}(f_1-f_3) \qquad (۱۲-۴)$$

معادلهٔ(۴-۱۲) زمانی در داخل مرزها به کار برده می‌شود که مقادیر تابع و مشتق آنها موجود باشند.مشابه این روابط را می‌توان برای جریان خروجی ازمرز، هنگامی که تابع و مشتق آن در دسترس است به کار برد یعنی:

$$f''_j+2f''_{j-1}=\frac{3}{\Delta x}\frac{df}{dx}|_{(x=L_x)}-\frac{3}{2\Delta x^2}(f_j-f_{j-2}) \qquad (۱۳-۴)$$

در نزدیکی مرزهای جریان (در جهت جریان) (در j=2 و j=J-1) مرتبهٔ دوم از معادلات تفاضلات محدود فشرده استفاده می‌شود یعنی رابطهٔ(۴-۷) با $\alpha=1/10$. یک ارزیابی از مشتق مرتبهٔ چهار، مورد استفاده در روابط این است که مشتقات مرتبه دوم را دوبار بر روی معادلات اعمال کنیم.

۴-۳- نگاشت یک به یک جبری(Algebraic mapping)

در روش عددی برای حل مسائل لایه مرزی توربولانت ، باید این توانایی را داشته باشیم که یک ساختار در جریان به گونه‌ای شبیه‌سازی کنیم که یک مقیاس فاصله بندی شدهٔ خاصی نزدیک دیواره داشته باشیم. این شبیه‌سازی نیازمند این است که فاصلهٔ گره‌ها نزدیک دیواره خیلی ریز باشد. این عمل باعث صرفه جویی در هزینه محاسبات خواهد شد. این عمل در مورد جریان برشی نیز صاق است که در همین فصل بحث خواهد شد.

۱-تابع نگاشت برای لایه مرزی روی صفحهٔ تخت

تابع نگاشت زیر مفروض است:

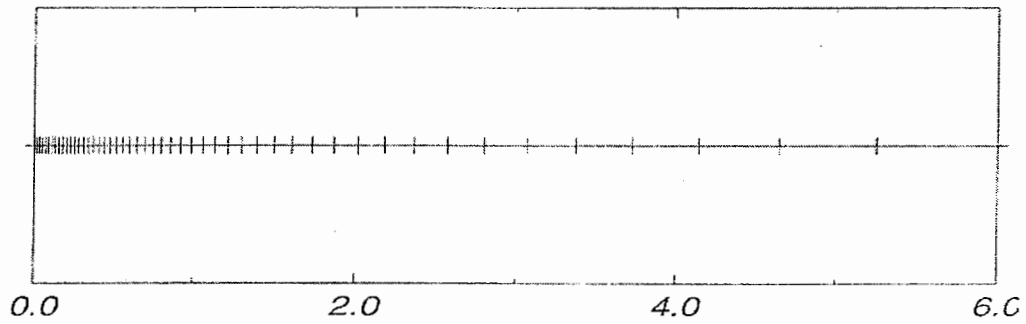
$$\eta(y)=\frac{y(L_y+y_0)}{L_y(y+y_0)} \qquad (۱۴-۴)$$

این نگاشت جبری ناحیه محاسباتی را به صورت زیر نگاشت می‌کند. $\eta\in[0,1]$ to $y\in[0,L_y]$

در رابطهٔ (۴-۱۴) y_0 پارامتر نگاشت است و میزان کشیدگی را تعیین می‌کند. در $y_0\to\infty$ ، $\eta(y)\to\frac{y}{L_y}$. که یک شبکه بندی منظم را ارائه می‌دهد. معکوس رابطهٔ بالا به صورت زیر است:

$$y(\eta)=\frac{L_y y_0\eta}{y_0+L_y(1-\eta)} \qquad (۱۵-۴)$$

یک شبکه بندی منظم در فواصل η ، $\Delta\eta=\frac{1}{N_y}$, $i=0,N_y$, $\eta_i=i\Delta\eta$ باعث ایجاد یک شبکهٔ کشیده شده در y می شود . شکل(۴-۱) توزیع گره‌ها را بر روی یک شبکه کشیده شده(Stretched) با $N_y=50$, $L_y=6$, $y_0=0$ را نشان می‌دهد.



شکل۴-۱: توزیع شبکه در یک شبکه بندی کشیده شدهٔ جبری با $N_y = 50$, $L_y = 6$, $y_0 = 0$ با استفاده از تابع نگاشت (۴-۱۴)

مشتقات جزیی به صورت زیر هستند:

$$\eta_y = \frac{\eta y_0}{y(y+y_0)}$$
$$\eta_{yy} = -2\frac{\eta y_0}{y(y+y_0)^2}$$
$$\eta_{yyy} = 6\frac{\eta y_0}{y(y+y_0)^3} \qquad (۱۶-۴)$$
$$\eta_{yyyy} = -24\frac{\eta y_0}{y(y+y_0)^4}$$

و مشتق تابع معکوس به صورت زیر است :

$$y_n = \frac{y_0 L_y (y_0 + L_y)}{(y_0 + L_y(1-\eta))^2} \qquad (۱۷-۴)$$

مشتقات در فواصل فیزیکی به وسیلهٔ قانون زنجیری[1] به هم مرتبط می‌شوند . عبارتهای مشتق اول ، دوم و چهارم در فواصل فیزیکی تابعی از مشتقات در دامنهٔ محاسباتی به صورت رابطهٔ (۴-۱۸)است :

$$\frac{d}{dy} = \eta_y \frac{d}{d\eta}$$
$$\frac{d^2}{dy^2} = \eta_{yy}\frac{d}{d\eta} + \eta_y^2 \frac{d^2}{d\eta^2} \qquad (۱۸-۴)$$
$$\frac{d^4}{dy^4} = \eta_{yyyy}\frac{d}{d\eta} + (4\eta_y\eta_{yyy} + 3\eta_{yy}^2)\frac{d^2}{dy^2} + 6\eta_y^2\eta_y y\frac{d^3}{d\eta^3}\eta_y^4\frac{d^4}{d\eta^4}$$

---

[1] Chain

۲-تابع نگاشت برای جریان برشی

برای تحلیل عددی جریان برشی از یک تابع نگاشت دیگری استفاده می کنیم که اندکی متفاوت با حالت لایه مرزی است در این قسمت مرز جامدی نداریم و می خواهیم شبکه از $(-\infty,+\infty)$ را به $(0,1)$ نگاشت کنیم تابع نگاشتی که در این قسمت مورد استفاده قرار گرفته است به صورت زیر است:

$$y = -\beta \cot(\pi\zeta) \qquad (۱۹-۴)$$

۴-۴-استخراج اپراتورهای مشتق در حالت استفاده از تابع نگاشت

مشتقات جزئی که در قسمت قبل برای محاسبهٔ مشتقات در جهت جریان به کار برده شد در این قسمت نیز می‌تواند برای محاسبهٔ مشتقات به کار رود اگر معادلات بسط یافته را در نظر بگیریم مشتقات به صورت زیر است:

$$f'_{j-1} + \frac{1}{\alpha} f'_j + f'_{j+1} = \frac{1+\frac{2}{\alpha}}{3\Delta x}(f_{j+1} - f_{j-1}) + \frac{4-\frac{1}{\alpha}}{12\Delta x}(f_{j+2} - f_{j-2}) \qquad (۲۰-۴)$$

و

$$\alpha f''_{j-1} + f''_j + \alpha f''_{j+1} = \frac{4(1-\alpha)}{3\Delta x^2}(f_{j-1} - 2f_j + f_{j+1}) + \frac{10\alpha - 1}{12\Delta x^2}(f_{j-2} - 2f_j + f_{j-2}) \qquad (۲۱-۴)$$

حال اگر این معادلات را به صورت ماتریسی بنویسیم داریم :

$$A_1 \frac{df}{d\eta} = B_1 f \qquad (۲۲-۴)$$

و

$$A_2 \frac{d^2 f}{d\eta^2} = B_2 f \qquad (۲۳-۴)$$

کـه در ایـن معـادلات مـاتریس‌هـای A_1 , B_1 , A_2 , B_2 از رابطـهٔ(۴-۲۰) و(۴-۲۱)بدسـت می‌آیند در جهت عمود بر جریان نیز می‌توانیم معادلات مشابه را ارائه کنیم که به صورت زیر است:

$$D = \Lambda_1 A_1^{-1} B_1$$
$$D^2 = \Lambda_2 A_2^{-1} B_2 + \Lambda_3 A_1^{-1} B_1 \qquad (۲۴-۴)$$

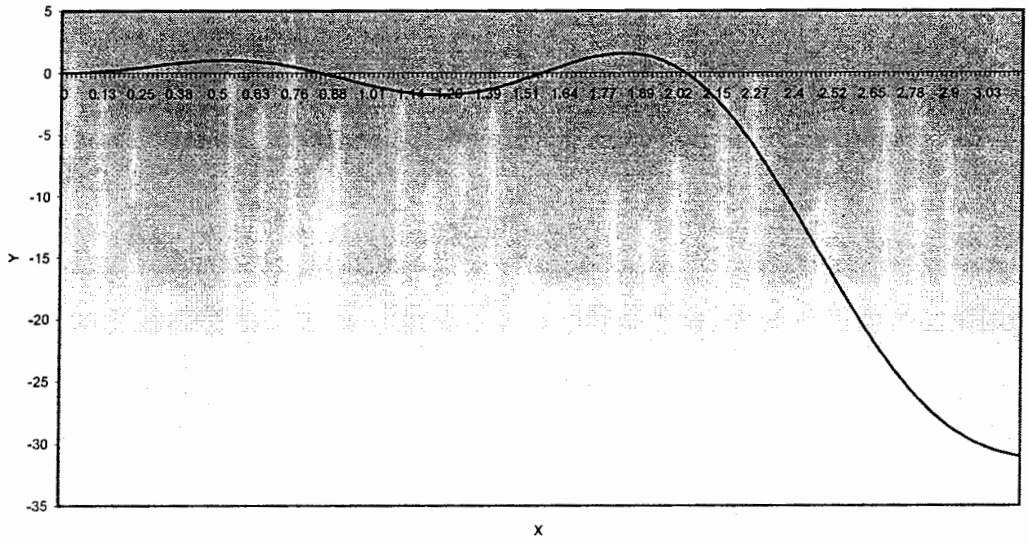
که در این معادلات Λ_1 , Λ_2 و Λ_3 ماتریس‌های قطری با ترتیب با مقـادیر $\frac{\eta y_0}{y(y+y_0)}$ ، $\left(\frac{\eta y_0}{y(y+y_0)}\right)^2$ و $-2\frac{\eta y_0}{y(y+y_0)^2}$ برای لایه مرزی روی صفحهٔ تخت و برای جریان برشی بـه ترتیب برابر $\frac{\sin^2(\pi\zeta)}{\pi\beta}$ ، $\left(\frac{1}{\pi\beta}\sin^2(\pi\zeta)\right)^2$ و $\frac{2}{\pi\beta^2}\sin^3(\pi\zeta)\cos(\pi\zeta)$است . باید دقت داشـت که شرایط مرزی به صورت پیش‌فرض ارضا می‌شوند چرا که اولین و آخرین عضو در ماتریس هـای قطری Λ_1 , Λ_2 و Λ_3 برابر صفر است.

۴-۵-ارزیابی و تست کدهای نوشته شده برای محاسبهٔ مشتقات

برای تست کدهای نوشته شده برای محاسبهٔ مشتقات ، مشتق توابع ریاضی زیـر را حـساب کرده‌ایم و با مشتق تحلیلی مقایسه کرده‌ایم که نتایج به صورت زیر است

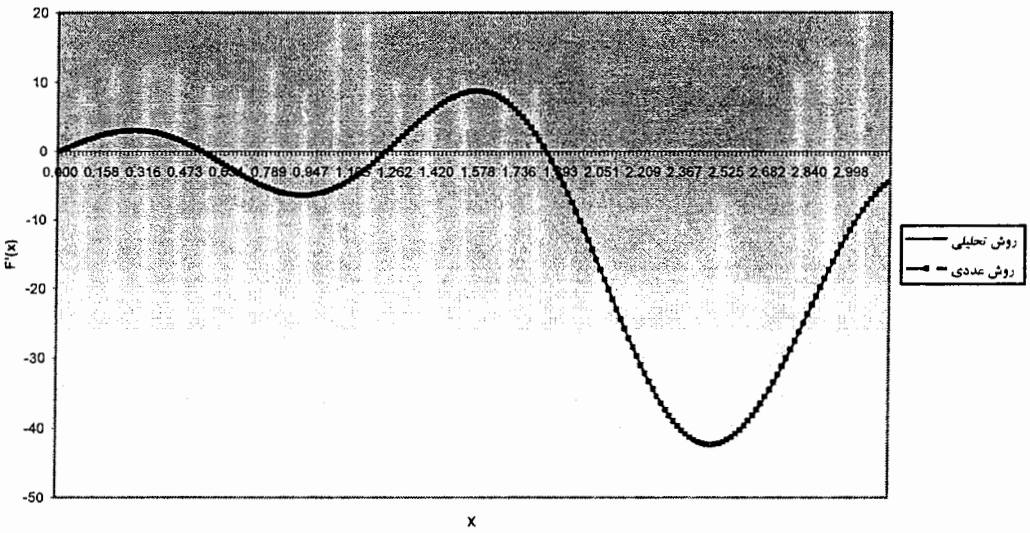
۱-محاسبه مشتق اول تابع

به عنوان اولین تست تابع $f(x) = x^3 \cos x + 2x\sin(4x)$ در نظـر گرفتـه شـده اسـت کـه نمودار تابع و رفتار آن در شکل(۴-۲) موجود است:



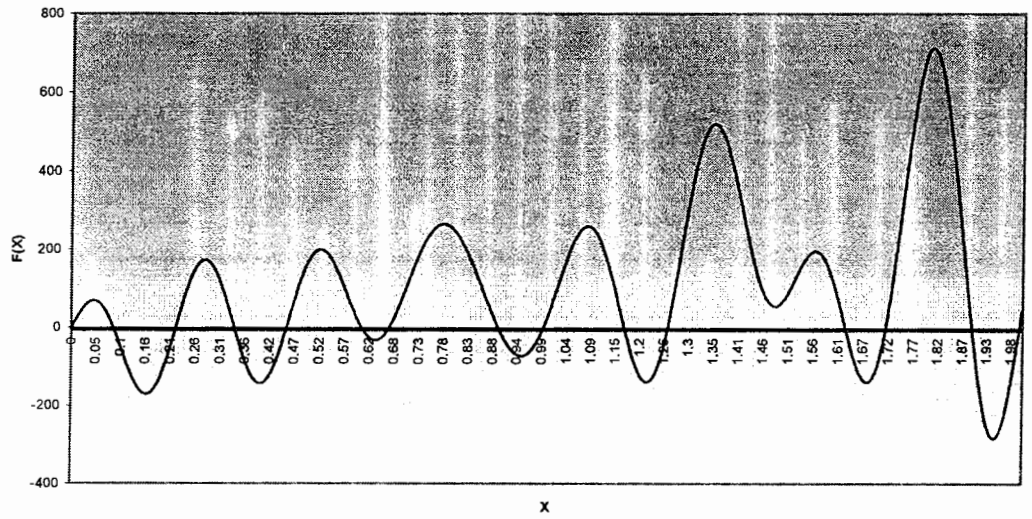
شکل۴-۲:نمودار تابع $f(x) = x^3 \cos x + 2x \sin(4x)$

مشتق اول این تابع که به روش تفاضلات محدود فشرده محاسبه شده است همراه با مشتق که بـه روش تحلیلی محاسبه شده است در شکل (۴-۳) رسم شده و قابل مقایسه است.

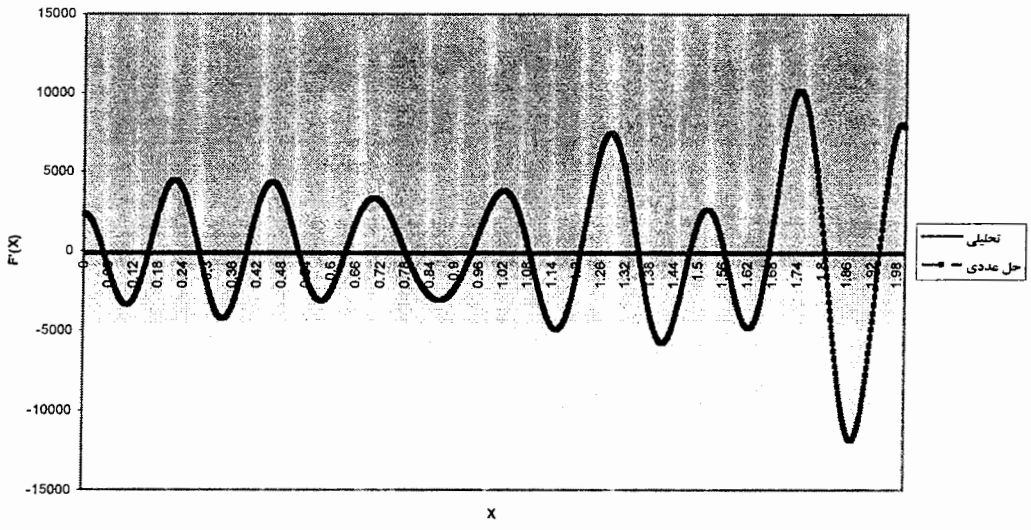


شکل۴-۳: نمودار مشتق اول تابع $f(x) = x^3 \cos x + 2x \sin(4x)$ (۲۰۰ گره)

برای اطمینان از صحت محاسبات ، سعی شده است تا مشتق مرتبهٔ اول یک تابع کاملاً موجی محاسبه شود و با حل تحلیلی مقایسه شود؛ این تابع به صورت $f(x) = \sum_{i=1}^{30} a_i (\sin x + ix \cos(ix))$ در نظر گرفته شده است که در ر ابطهٔ بالا a_i ها به صورت تصادفی ایجاد شده است یعنی در سری رابطهٔ بالا ابتدا مقادیر a_i توسط تابع Random که از توابع کتابخانه‌ای در فرترن است، ایجاد شده است و سپس مقدار تابع در هر ایستگاه مشخص با جاگذاری مقدار x و سپس محاسبهٔ جمع سری بدست می‌آید سپس مشتق تابع را به روش تحلیلی بدست آورده و نتایج با نتایج عددی مقایسه شده است. نکتهٔ قابل توجه در این است که اگر تعداد گره‌ها کم باشد دقت محاسبات خیلی کم است که این موضوع از طبیعت موجی بودن تابع نشعت می‌گیرد. همانطور که در نمودار توابع قبل ذکر شد، برای محاسبهٔ مشتق عددی این توابع تعداد ۲۰۰ گره خطای خیلی کمی دارد حال آنکه برای این تابع خطای محاسبه برای ۲۰۰گره قابل توجه بوده و در نتیجه در مورد این تابع ۱۰۰۰گره انتخاب شده است . نتایج در شکل(۴-۴) و (۴-۵) ارائه شده است که مشتق عددی و تحلیلی از مرتبهٔ اول را باهم مقایسه می‌کند.



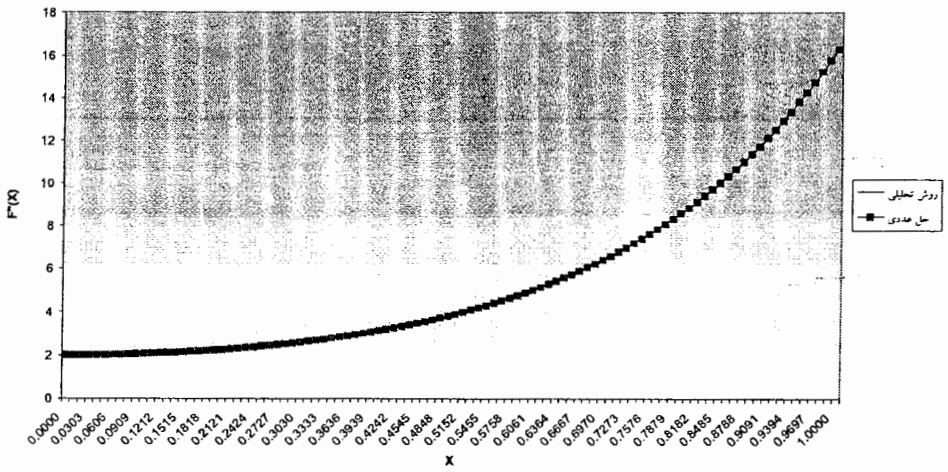
شکل۴-۴: نمودار تابع $f(x)=\sum_{i=1}^{30} a_i(\sin x + ix\cos(ix))$



شکل۴-۵: نمودار مشتق اول تابع $f(x)=\sum_{i=1}^{30} a_i(\sin x + ix\cos(ix))$ (با ۱۰۰۰ گره)

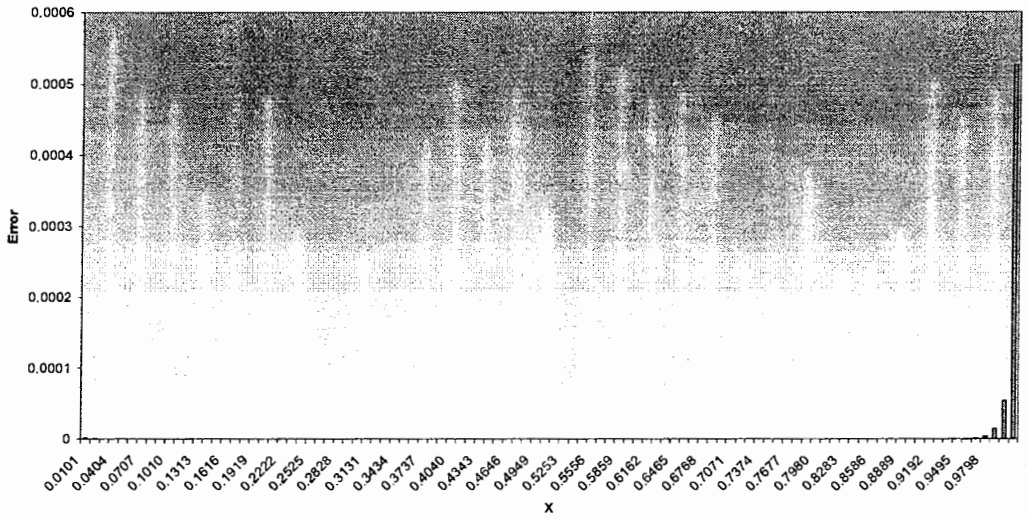
۲-محاسبه مشتق دوم تابع

مشتق دوم تابع قسمت قبل هم به روش عددی و هم روش تحلیلی محاسبه شده است کـه نتایج به صورت زیر است:



شکل۴-۶: نمودار مشتق دوم تابع $f(x)=e^{x^2}$ (با ۱۰۰گره)

همچنین نمودار خطای نسبی مشتق دوم در شکل (۴-۷) رسم شده است.



شکل۴-۷: نمودار خطای نسبی مشتق دوم تابع $f(x)=e^{x^2}$ (با ۱۰۰گره)

۳-مقایسه مشتق محاسبه شده در تابع نگاشت با حل تحلیلی

در این قسمت مانند قسمت قبل مشتق اول و دوم چند تابع ریاضی را که در محاسبه مشتق آنها از تابع نگاشت استفاده شده است ،بررسی کرده و با حل تحلیلی مقایسه کرده ایم.

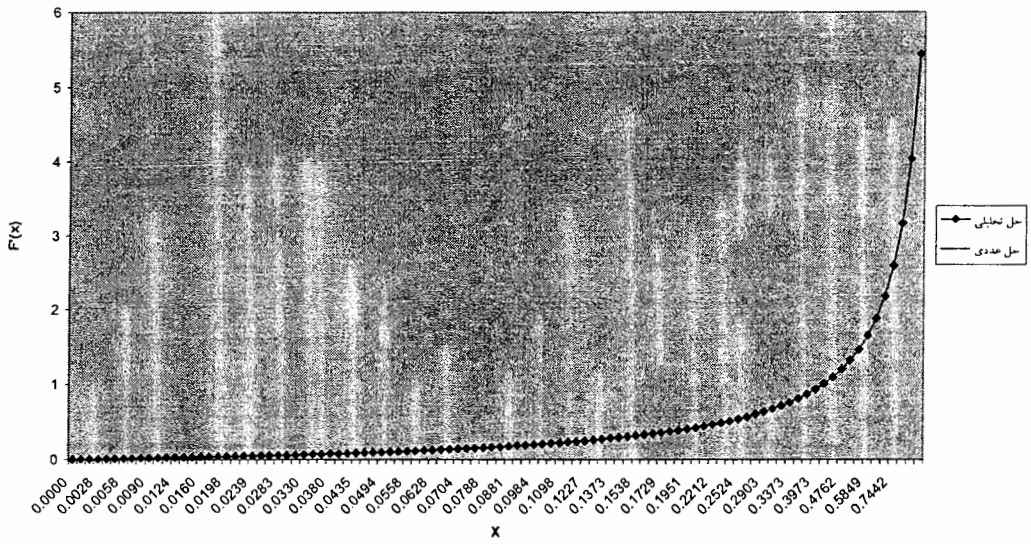
الف) استفاده از نگاشت لایه مرزی

همانطور که در قسمت های قبل اشاره شد نگاشت مورد استفاده برای لایه مرزی روی صفحهٔ تخت به صورت زیر در نظر گرفته شده است :

$$y(\eta)=\frac{L_y y_0 \eta}{y_0+L_y(1-\eta)} \qquad (۲۵-۴)$$

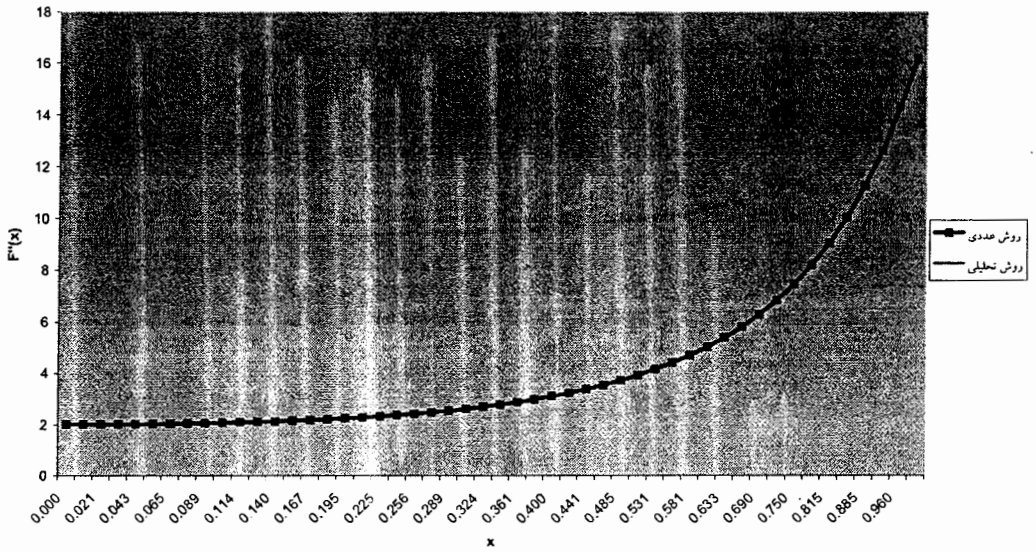
برای تست محاسبهٔ مشتقات به روش تفاضلات محدود فشرده همراه با تابع نگاشت لایه مزری، مشتق چند تابع محاسبه شده است که نتایج به صورت زیر است:

شکل (۴-۸) مشتق اول تابع $f(x)=e^{x^2}$ را با اعمال نگاشت نشان می دهد.



شکل۴-۸: نمودار مشتق اول تابع $f(x)=e^{-x^2}$ با اعمال نگاشت رابطهٔ۴-۲۵(با ۱۰۰گره)

و شکل (۴-۹)نمودار مشتق دوم همین تابع را نشان می دهد:



شکل۴-۹: نمودار مشتق دوم تابع $f(x)=e^{x^2}$ با اعمال نگاشت۴-۲۵(با ۱۰۰گره)

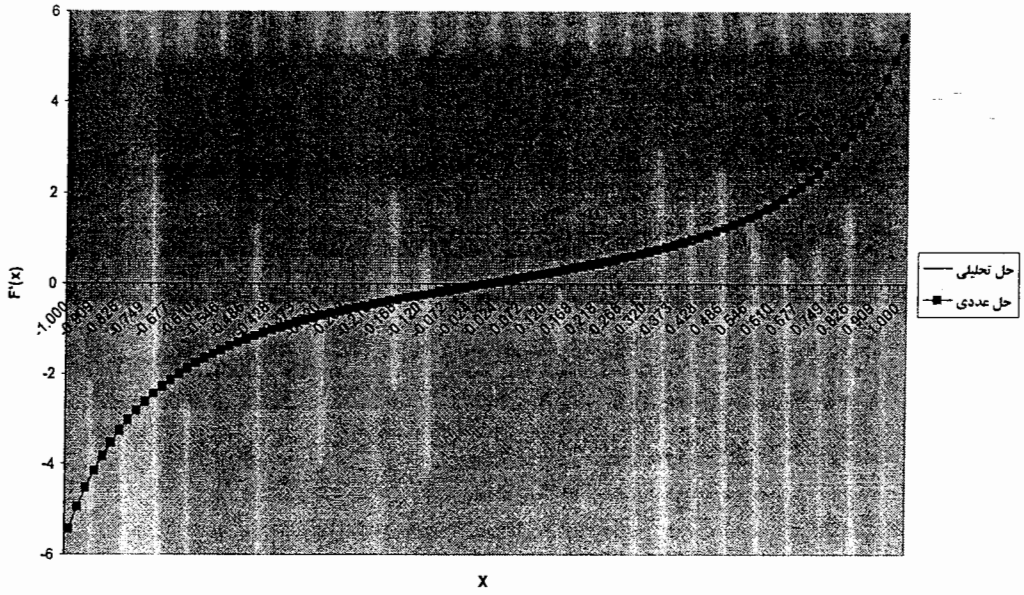
ب) استفاده از نگاشت کتانژانت برای جریان برشی

مطابق بحث قسمت های قبل تابع نگاشت استفاده شده برای جریان برشی به صورت معادلهٔ (۴-۲۶) است .

$$y = -\beta \cot(\pi\zeta) \qquad (۴-۲۶)$$

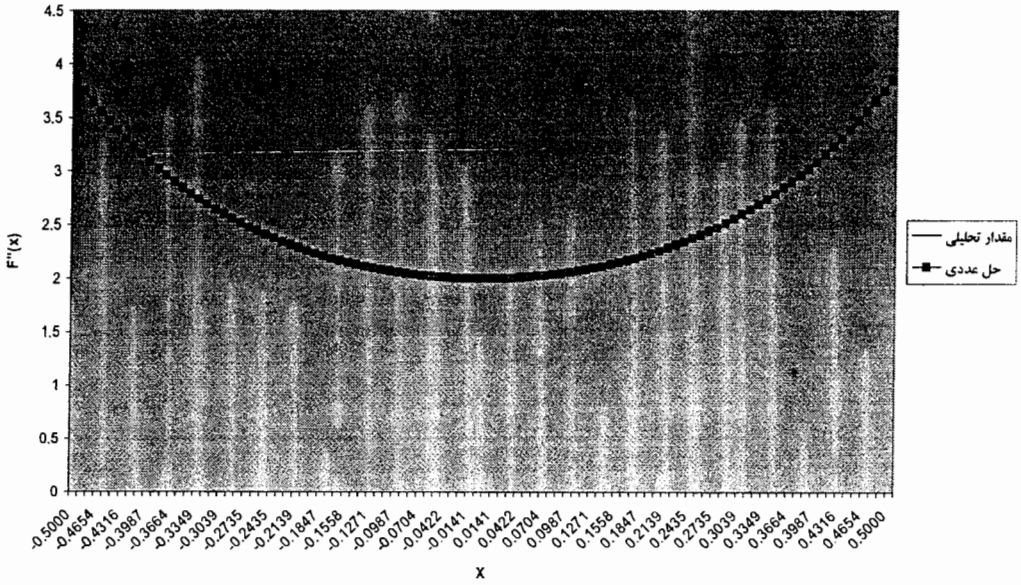
برای تست محاسبهٔ مشتقات به روش تفاضلات محدود فشرده همراه با تابع نگاشت جریان برشی، مشتق چند تابع محاسبه شده است که نتایج به صورت زیر است:

شکل ۴-۱۰ مشتق اول تابع $f(x) = e^{x^2}$ را با اعمال نگاشت جریان برشی نشان می دهد.



شکل۴-۱۰: نمودار مشتق اول تابع $f(x) = e^{x^2}$ با اعمال نگاشت رابطه ۴-۲۶(با ۱۰۰گره)

و شکل ۴-۱۱ نمودار مشتق دوم همین تابع را نشان می دهد:



شکل۴-۱۱: نمودار مشتق دوم تابع $f(x)=e^{x^2}$ با اعمال نگاشت رابطه ۴-۲۶(با ۱۰۰گره)

۴-۶-الگوی پیشروی در زمان

برای انجام محاسبات و پیشروی در زمان از روش رانج-کوتا مرتبهٔ سوم که توسط Wray ارائه شده است استفاده می‌کنیم. کاربرد این روش در معادله مدل زیر در جدول(۴-۱) و در سه زیر مرحلهٔ زمانی آورده شده است.

$$\frac{du}{dt}=R(u) \qquad (۴-۲۷)$$

جدول ۴-۱:الگوی پیشروی در زمان رانج – کوتا مرتبهٔ سوم

Time	First location	Second location
t^n	u^n	$R(u^n)$
$t'=t^n+c_1\Delta t$	$u'=u^n+c_1\Delta tR$	$R'=R(u')$
$t''=t'+(c_2+d_2)\Delta t$	$u''=u'+(c_2R'+d_2R)\Delta t$	$R''=R(u'')$
$t^{n+1}=t^n+\Delta t$	$u^{n+1}=u''+(c_3R''+d_3R')\Delta t$	

این جدول نشان می‌دهد که پیشرفت زمان در معادلهٔ(۴-۲۷) بوسیلهٔ پیشرفت زمانی (Δt) نیازمند محاسبهٔ طرف راست معادله (R)در سه زیربازهٔ زمانی است. در هر زیر بازهٔ زمانی، زمان با رابطهٔ $(c_i + d_i)\Delta t$ افزایش می‌یابد و u بوسیلهٔ ترکیبی خطی از R درزمان جاری و زیر بازهٔ زمانی قبلی بدست می‌آید.

ضرایب استفاده شده در روش پیشروی زمان $(c_i + d_i)$ با بسط سری تیلور نسبت به متغیر زمان و مساوی قرار دادن ضرایب هم‌مرتبه به دست می‌آید، پس داریم:

$$
\begin{aligned}
c_1 + c_2 + c_3 + d_1 + d_2 + d_3 &= 1 \\
c_1c_2 + c_3[\frac{d_2}{c_2}(1 + \frac{d_3}{c_3}) + c_2(1 + \frac{d_2}{c_2})] &= \frac{1}{2} \\
c_1^2c_2 + c_3[c_1 + c_2(1 + \frac{d_2}{c_2})]^2 + c_1^2d_3 &= \frac{1}{3} \\
c_1c_2c_3 &= \frac{1}{6}
\end{aligned}
$$

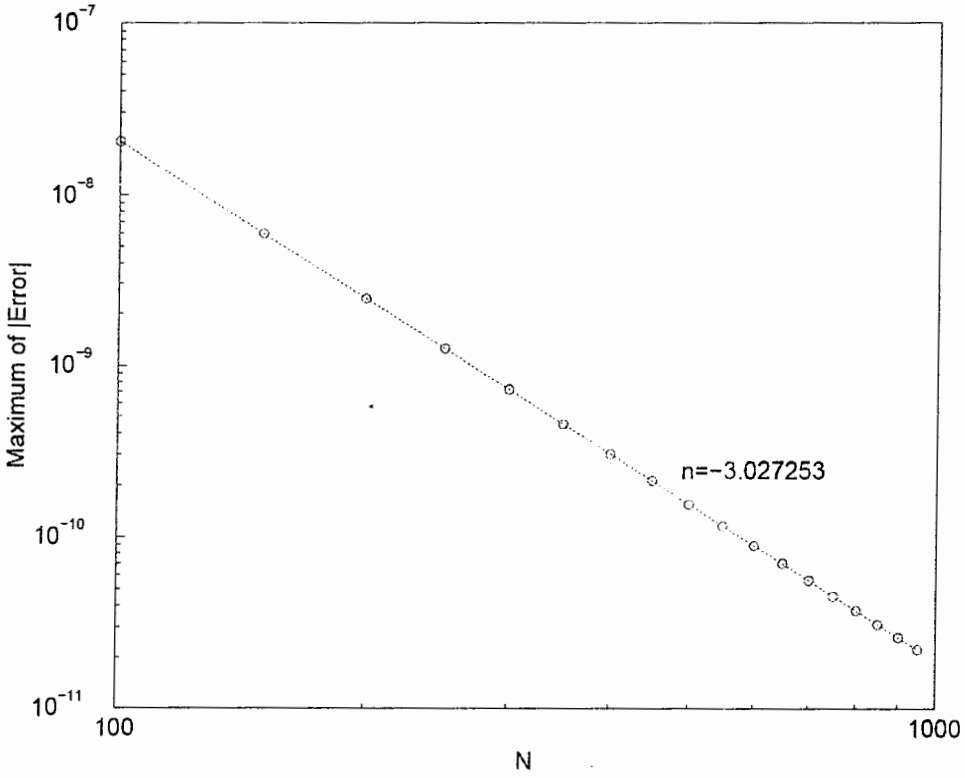
در روابط بالا مقدار دو پارامتر باید پیش‌بینی شود تا حل کامل شود. در شروع حل واضح است که d=0 . پس بقیهٔ پارامترها به صورت تابعی از یک پارامتر بدست می‌آید. یک جواب از دستهٔ جوابها می‌تواند به صورت زیر باشد:

$$
\begin{aligned}
c_1 &= 8/15, & d_1 &= 0 \\
c_2 &= 5/12, & d_2 &= -17/60 \\
c_3 &= 3/4, & d_3 &= -5/12
\end{aligned}
$$

برای تست روش ذکر شده در بالا و همچنین بررسی میزان دقت آن معادلهٔ زیر را در نظر گرفته و آن را به روش عددی حل می‌کنیم:

$$\frac{du}{dt} = -u(t) \quad , \quad \text{u(0) = -1}$$

می‌دانیم که جواب حل تحلیلی معادلهٔ بالا به صورت $u(t) = e^{-t}$ است. ماگزیمم خطای نتایج عددی و حل دقیق در شکل(۴-۱۲) رسم شده است. این شکل به طور واضح میزان دقت این روش را که از مرتبهٔ سه است نشان می‌دهد.



شکل۴-۱۲: مرتبه دقت الگوی پیشروی زمان **[4]**

فصل ۵-معادلات دیفرانسیل حاکم و الگوریتم حل عددی آنها

۵-۱-مقدمه

در این فصل معادلات حاکم بر حرکت سیال یعنی معادلات پیوستگی و ناویر- استوکس و همچنین روشهای حل مستقیم عددی این معادلات بررسی خواهد شد. در ابتدای این فصل با اعمال چند اپراتور برداری معادلات ناویر- استوکس را به صورتی دیگر تبدیل می‌کنیم تا بوسیلهٔ آن بتوانیم مجهولات را در این معادله کم کرده و آن را به روش عددی حل کنیم. مشتقات جزئی به همان روشی که در فصل ۴ بررسی شد، محاسبه می‌شود.

۵-۲-شکل چرخشی معادلهٔ ناویر- استوکس

با در نظر گرفتن رابطه برداری مشخص زیر:

$$\nabla(A.B) = (B.\nabla)B + (A.\nabla)B + B\times(\nabla\times A) + A\times(\nabla\times B) \qquad (۱-۵)$$

وبا فرض اینکه بردارهای **A** و **B** با هم برابر و مساوی بردار **U** باشد، داریم:

$$A = B = U = (U, V, W)$$

$$(U.\nabla)U = \omega\times U + \frac{1}{2}\nabla(U.U) \qquad (۲-۵)$$

که در آن:

$$\omega = (\omega_1, \omega_2, \omega_3) = \nabla \times U$$

ازطرفی می دانیم که فرم بی بعد شده معادله ناویر استوکس به صورت رابطه (۵-۳) می باشد:

$$\frac{\partial U}{\partial t} + (U.\nabla)U = -\nabla p + \frac{1}{\mathrm{Re}}(\nabla^2 U) \qquad (۵-۳)$$

با درنظر گرفتن رابطه (۵-۲) و (۵-۳) به رابطه (۵-۴) خواهیم رسید:

$$\frac{\partial U}{\partial t} = H - \nabla\left(p + \frac{U.U}{2}\right) + \frac{1}{\mathrm{Re}}(\nabla^2 U) \qquad (۵-۴)$$

که در آن:

$$H = (H_1, H_2, H_3) = U \times \omega$$

با ضرب طرفین معادله (۵-۴) در بردار کرل ($\nabla\times$) داریم:

$$\frac{\partial(\nabla \times U)}{\partial t} = \nabla \times H - \nabla \times \nabla\left(P + \frac{U.U}{2}\right) + \frac{1}{\mathrm{Re}}\nabla^2(\nabla \times U) \qquad (۵-۵)$$

از طرفی می دانیم که $\nabla(scalar) = 0$ می باشد. درنتیجه معادله (۵-۵) به فرم معادله (۵-۶) تبدل خواهد شد:

$$\frac{\partial \omega}{\partial t} = \nabla \times H + \frac{1}{\mathrm{Re}}\nabla^2 \omega \qquad (۵-۶)$$

با تکرار عمل ضرب بالا در معادله (۵-۶) به معادله زیر خواهیم رسید:

$$\frac{\partial \nabla \times (\nabla \times U)}{\partial t} = \nabla \times (\nabla \times U) + \frac{1}{\mathrm{Re}}\nabla^2\big(\nabla \times (\nabla \times U)\big) \qquad (۵-۷)$$

با بکار بردن معادله پیوستگی $(\nabla.U) = 0$ و رابطه ریاضی مشخص زیر

$$\nabla \times (\nabla \times U) = \nabla(\nabla.U) + \nabla^2 U \qquad (۵-۸)$$

به رابطه محاسباتی (۵-۹) خواهیم رسید:

$$\frac{\partial \nabla^2 U}{\partial t} = \nabla \times (\nabla \times H) + \frac{1}{\mathrm{Re}}\nabla^4 U \qquad (۵-۹)$$

مزیت رابطهٔ(۵-۹) در این است که ترم فشار از معادلهٔ ناویر- استوکس حذف شده و می توان سرعت در لحظهٔ n+1 را بعد از گسسته‌سازی معادله بدست آورد . اما از طرفی دیگر چون مرتبهٔ مشتقات بزرگتر شده است ، هزینه محاسبات بیشتر شده و اعمال شرایط مرزی نیز سخت‌تر شده است.

۵-۳- روش حل عددی معادلات

برای حل عددی معادلهٔ (۵-۹) ابتدا سرعت را به صورت اجزای زیر تفکیک می‌کنیم:

$$U(x,y,z,t) = u(x,y,z,t) + U_0(y) \qquad (۱۰-۵)$$

که در رابطهٔ بالا $U_0(y)$ همان سرعت مبنا است و فقط تابع مختصات y است. ترم بعد همان ترم محاسباتی است که باید به روش عددی محاسبه شود. مزیت تفکیک انجام شده در معادلهٔ (۵-۱۰) در این است که مقدار سرعت در مرزهای جریان آزاد برابر صفر خواهد شد. با این تفکیک می‌توان معادلهٔ(۵-۹) را به صورت زیر نوشت:

$$\frac{\partial \nabla^2 u}{\partial t} = \nabla \times (\nabla \times H) + \frac{1}{\mathrm{Re}} \nabla^4 U \qquad (۱۱-۵)$$

برای اینکه یک روش عددی مناسب برای حل معادلهٔ (۵-۱۱) ارائه دهیم، ابتدا طرف راست معادله را که در حقیقت یک عبارت شامل مشتقات جزئی است را به روش تفاضلات محدود فشرده محاسبه می‌کنیم که نتیجه یک ماتریس $n \times m$ خواهد بود این ماتریس را با RHS نمایش می دهیم . با این عمل و گسسته سازی طرف چپ معادلهٔ(۵-۱۱) به معادلهٔ زیر می‌رسیم:

$$\frac{\nabla^2 u^{n+1} - \nabla^2 u^n}{\Delta t} = RHS^n \qquad (۱۲-۵)$$

و یا به عبارت دیگر سرعت در لحظهٔ n+1 از رابطهٔ زیر محاسبه می‌شود:

$$\nabla^2 u^{n+1} = \Delta t (RHS^n + \nabla^2 u^n) \qquad (۱۳-۵)$$

معادلهٔ(۵-۱۳) در حقیقت همان معادلهٔ پواسون دو بعدی است در این قسمت می‌خواهیم اثبات کنیم که معادلهٔ پواسون بعد از گسسته‌سازی مشتقات جزئی به روش تفاضلات محدود فشرده ، به فرم ماتریسی AX+XB=C تبدیل می‌شود.

۵-۴-روش گسسته‌سازی معادلهٔ پواسون دو بعدی

در حالت کلی معادله پواسون دو بعدی به صورت زیر است:

$$\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}=R(x,y) \qquad (۱۴-۵)$$

حال فرض کنیم که می‌خواهیم معادله پواسون را برای یک دامنه حل کنیم که یک شبکه $I\times J$ را به صورت شکل زیر تشکیل می‌دهد:

$$\begin{matrix} u_{I\times 1} & u_{I\times 2} & \cdots & u_{I\times J} \\ \vdots & \vdots & \ddots & \vdots \\ u_{2\times 1} & u_{2\times 2} & \cdots & u_{2\times J} \\ u_{1\times 1} & u_{1\times 2} & \cdots & u_{1\times J} \end{matrix}$$

پس u به صورت $[u_{ij}]$ در نظر گرفته می‌شود حال از معادلات تفاضل محدود فشرده داریم

الف) مشتق دوم در جهت X

$$Au_{xx}=Bu \qquad (۱۵-۵)$$

که در آن u_{xx} مشتق دوم در جهت X است معادلهٔ بالا به شکل ماتریس به صورت زیر است:

$$A_{I\times J}u^{T}_{xxI\times J}=B_{I\times J}u^{T}_{I\times J} \quad \Rightarrow \quad u^{T}_{xxI\times J}=(A^{-1}_{I\times J}B_{I\times J})u^{T}_{I\times J} \qquad (۱۶-۵)$$

حال یک بار دیگر ترانهادهٔ طرفین رابطهٔ بالا را محاسبه می‌کنیم تا به رابطهٔ(۵-۱۷)برسیم:

$$u_{xxI\times J}=u_{I\times J}(A^{-1}_{I\times J}B_{I\times J})^{T}=UB \qquad (۱۷-۵)$$

ب) مشتق دوم در جهت Y

$$Au_{yy}=Bu$$

که در آن u_{yy} مشتق دوم در جهت Y است معادلهٔ بالا به شکل ماتریس به صورت (۵-۱۸) است:

$$A_{I\times J}u_{yyI\times J}=B_{I\times J}u_{I\times J} \quad \Rightarrow \quad u_{yyI\times J}=(A^{-1}_{I\times J}B_{I\times J})u_{I\times J}=AU \qquad (۱۸-۵)$$

با جاگذاری این روابط و تبدیل R به فرم ماتریسی یعنی $R_{I\times J}$ در معادلهٔ پواسون داریم:

$$AU + UB = R \qquad (۱۹-۵)$$

۵-۵- حل معادلهٔ ماتریسی AX+XB=C

در این قسمت می‌خواهیم معادلهٔ ماتریسی زیر را حل کرده و ماتریس مجهول X را محاسبه کنیم.[2]

$$AX + XB = C \qquad (۲۰-۵)$$

در رابطهٔ بالا A ، B، C ماتریسهای حقیقی با ابعاد $m \times m$ و $n \times n$ و $m \times n$ هستند.که m تعداد گره در جهت y و n تعداد گره در جهت x است. از سابروتین‌های اضافی موجود می‌توان برای حل معادله(۵-۲۱) نیز استفاده کرد:

$$A^T X + XA = C \qquad (۲۱-۵)$$

که در اینجا ماتریسC متقارن می باشد . معادله (۵-۲۰) در حل مستقیم معادله پواسون کاربرد دارد .

کاملا آشکار است که معادله (۵-۲۰) فقط یک راه حل دارد ؛اگر و فقط اگر ، مقادیر ویژه $\alpha_1, \alpha_2, ..., \alpha_m$ ماتریس A و $\beta_1, \beta_2, ..., \beta_n$ ماتریس B شرط زیررا ارضا کند :

$$\alpha_i + \beta_j \neq 0 \qquad (i = 1,2,...., m; j = 1,2,...., n) \qquad (۲۲-۵)$$

روش حل بر مبنای یک روش محاسباتی می باشد اما اگر شرایط سیستم ، مقدار ویژه ناهنجار باشد روش حل ناپایدار خواهد بود . روش پیشنهاد شده در اینجا استفاده از روش کاهش مثلثی Schur با استفاده از تبدیل شبه متعامد می باشد .

معادله (۵-۲۰) به این صورت حل می شود .

ماتریس A با استفاده از تبدیل شبه متعامدU به ماتریس حقیقی Lower Schur مانند A' کاهش می یابد.

$$A' = U^T AU = \begin{bmatrix} A'_{11} & & & O \\ A'_{21} & A'_{22} & & \\ \vdots & \vdots & & \\ A'_{p1} & A'_{p2} & \cdots & A'_{pp} \end{bmatrix} \qquad (۲۳-۵)$$

که هر ماتریس A'_{ii} حداکثر از مرتبه ۲ می باشد. ماتریس B نیز با استفاده از روش فـوق و با استفاده از ماتریس متعامد V به ماتریس Upper Schur مانند B' کاهش می‌یابد.

$$B' = V^T BV = \begin{bmatrix} B'_{11} & B'_{12} \ldots\ldots\ldots\ldots & B'_{1q} \\ & B'_{22} \ldots\ldots\ldots\ldots & B'_{2q} \\ O & & B'_{qq} \end{bmatrix} \qquad (۲۴-۵)$$

که ماتریس B'_{ii} نیز حداکثر از مرتبه ۲ می باشد. اگر

$$C' = U^T CV = \begin{bmatrix} C'_{11} & \cdots & C'_{111q} \\ \vdots & & \\ C'_{11} & \cdots & C'_{11} \end{bmatrix} \qquad (۲۵-۵)$$

و

$$X' = U^T XV = \begin{bmatrix} X'_{11} & \cdots & X'_{1q} \\ \vdots & & \\ X'_{p1} & \cdots & X'_{pq} \end{bmatrix} \qquad (۲۶-۵)$$

آنگاه معادله (۵-۲۰) با معادلهٔ زیر هم ارز می شود :

$$A'X' + X'B' = C'$$

اگر قسمت های X', C', B', A' همانند (Conformal)باشند آنگاه :

$$A'_{KK}X'_{KL} + X'_{KL}B'_{LL} = C'_{KL} - \sum_{j=1}^{k-1} A'_{Kj}X'_{jL} - \sum_{i=1}^{l-1} X'_{ki}B'_{iL} \qquad (۲۷-۵)$$

$$(k = 1,2,\ldots,p \;\; ; \;\; l = 1,2,\ldots,q$$

این معادله برای مقادیر $X'_{11}, X'_{21}, \ldots, X'_{p1}, X'_{12}, X'_{22}, \ldots$ حل می شود آنگاه حل معادله (۵-۲۰) بویسله $X = UX'V^T$ داده می شود.

کاهش ماتریس A و B به شکل حقیقی Schur با استفاده از روش های استاندارد انجام شده است . ماتریس B با استفاده از روش هاوس هولدر (Householders) به ماتریس بالا هسنبرگی(Hsesenberg) کاهش پیدا می کند. آنگاه ماتریس بالا هسنبرگی با استفاده از الگوریتم QR به فرم حقیقی Schur تبدیل می شود. نتیجه این تبدیل در کاهش به شکل ماتریس V مورد استفاده قرار می‌گیرد . تبدیل ماتریس A به فرم پایین مثلثی حقیقی Schur با استفاده از کاهش ماتریس ترانهاده A به بالا مثلثیSchur و سپس عمل عکس ترانسپوز مسیر می شود.

از آنجائیکه الگوریتم QR استفاده شده در اینجا یک روش تکراری می باشد ، کاهش عناصر زیر قطری ماتریس بالا مثلثی هسنبرگ به صفر ، نیاز به استفاده از فرضیاتی برای محاسبه ، وقتی که مقدار یک عنصر ناچیز می شود، دارد.

در این برنامه هر المان از H اگر دارای مقداری کمتر از $\varepsilon_H \|H\|_\infty$ باشد ،قابل صرفنظر کردن است. که در اینجا ε_H مقدار ثابتی است که توسط کاربر اعمال می شود. استفاده از این فرض اگر عناصر H دارای اندازه های یکسانی باشند ، فرض مناسبی می باشد اما اگر این عناصر در یک طیف وسیعی باشند و المان های کوچک مهم باشند. احتیاج به فرضیات دیگری می باشد.

حل X'_{KL} در (۵-۲۷) هنوز احتیاج به حل معادلهٔ ماتریسی به شکل (۵-۲۰) دارد. در این حالت ماتریس A'_{KK} و B'_{LL} از مرتبه حداکثر دو هستند. از اینرو حل (۵-۲۷) را می توان با حل سیستم خطی از مرتبه حداکثر چهار بدست آورد. برای مثال اگر A'_{KK} , B'_{LL} هر دو از مرتبه دو باشند.

آنگاه :

$$\begin{bmatrix} a'_{11}+b'_{11} & a'_{12} & b'_{12} & 0 \\ a'_{21} & a'_{22}+a'_{11} & 0 & b'_{21} \\ b'_{12} & 0 & a'_{11}+b'_{22} & a'_{12} \\ 0 & b'_{12} & a'_{21} & a'_{22}+b'_{22} \end{bmatrix} \begin{bmatrix} X'_{11} \\ X'_{21} \\ X'_{12} \\ X'_{22} \end{bmatrix} = \begin{bmatrix} d_{11} \\ d_{21} \\ d_{12} \\ d_{22} \end{bmatrix} \qquad (۵-۲۸)$$

که a'_{ij} , b'_{ij} , x'_{ij} المان های عناصر $X'_{LK}, B'_{LL}, A'_{KK}$ را مشخص می کنند و d_{ij} المان های سمت راست معادله (۵-۲۰) هستند.

دستگاهی که از معادله (۵-۲۷) بدست می آید با استفاده روش کاهش کروت (Crout) حل می‌شود.

این برنامه تمهیداتی برای کاربر در نظر گرفته است تا در آن مرحلهٔ تبدیل ماتریس A به فرم حقیقی Schur حذف شود در نتیجه چنانچه ماتریس تبدیل شدهA توسط کاربر وارد شود محاسبهٔ ماتریس های A' و U براحتی امکان‌پذیر بوده و از آنها برای می‌توان برای حل یک دستگاه جدید با ماتریس های B و C مختلف استفاده نمود.

عملاً می‌توان الگوریتم توصیف شده در بالا را برای حل سیستم های متقارن (۵-۲۱) بکاربردو در نتیجه می‌توان مزیت تقارن را مشاهده کرد. اگر U متعامد و $A' = U^T A U$ به شکل فرم بالا مثلثی حقیقی Schur باشد ، آنگاه مقادیر A' و $C' = U^T C U$ و $X' = U^T X U$ به شکل زیر خواهند بود :

$$A' = \begin{bmatrix} A'_{11} & A'_{12} \\ 0 & A'_{22} \end{bmatrix}$$

$$X' = \begin{bmatrix} X'_{11} & X'^{T}_{21} \\ X'_{21} & X'_{22} \end{bmatrix}$$

$$C' = \begin{bmatrix} C'_{11} & C'^{T}_{21} \\ C'_{21} & C'_{22} \end{bmatrix}$$

که A'_{11} ، X'_{11} ، C'_{11} حداکثر از مرتبه دو خواهند بود. آنگاه از معادله $A'^{T} X' + X' A' = C'$ داریم :

$$A'^{T}_{22} X'_{22} + X'_{22} A'_{22} = C'_{22} - X'_{21} A'_{12} - A'^{T}_{12} X'_{21}$$

بنابراین هنگامیکه X'_{11} ، X'_{21} محاسبه شدند، اندازه مساله می تواند کاهش یابد.

ماتریس X'_{21} در حالت کلی ، مانند روش توصیف شده دربالا می‌تواند محاسبه شود .

ماتریس X'_{11} معادله متقارن را ارضا می کند.

$$A'^{T}_{11} X'_{11} + X'_{11} A'_{11} = C'_{11} \qquad (۵-۲۹)$$

که اگر A'_{11} از مرتبه یک باشد ، حل این معادله آسان می باشد. اما هنگامیکه A'_{11} از مرتبه دوباشد، معادله فوق یک سیستم جدید مرتبه سه را برای سه المان متفاوت X'_{11} معرفی می کند.

در اینجا برای پرهیز از توضیح جزئیات خلاصهٔ برنامه را ذکر مـی‌کنـیم و خواننـده را بـرای مطالعهٔ بیشتربه مراجع راهنمایی می‌کنیم.

زیر برنامهAXPXB برنامه را برای حل معادله (۵-۲۰) هدایت می کند. با استفاده از A و B و C داده شده، ماتریس C با ماتریس جواب X بازنویسی می‌شود. شکل حقیقی پایینی مـاتریسA ،خود ماتریس A و شکل Schur بـالایی مـاتریس B ، مـاتریس B را بازنویـسی مـی کنـد. کـاربر می‌تواند فرمSchur را خود آماده کرده و مرحلهٔ تبدیل را حذف کند. این زیر برنامه سابروتین‌های HSHLDRو BCKMLT ،SCHUR ،SHRSLV وSYSSLV را احضار می‌کند.

ATXPXA: برنامه را برای حل معادله (۵-۲۱) هدایت می کند. بـا اسـتفاده از A و C داده شده،، ماتریس C با ماتریس جواب X بازنویسی می‌شود. . شکل حقیقـی بـالایی مـاتریسA ،خـود ماتریس A را بازنویسی می کند. کاربر می‌تواند فرمSchur را خود آماده کـرده و مرحلـهٔ تبـدیل را حذف کند. این زیـر برنامـه سـابروتین‌هـای HSHLDR و BCKMLT ،SCHUR ،SYMSLV و SYSSLV را احضار می‌کند.

HSHLDR: ماتریس A را به شکل بالا هسنبرگی کاهش می دهد. این فرم بالا هسنبرگی و تاریخچهٔ تبدیل ، A را بازنویسی می کند.

BCKMLT: ماتریس A را که به فرم بالا هسنبرگی است ،گرفتـه و مـاتریس متعامـد U را محاسبه می کند.

SCHUR: شکل حقیقی بالای Schur یک ماتریس بالا مثلثی هسنبرگ را محاسبه می‌کند.

SHRSLV: معادله (۵-۲۰) را حل می کند، در حالیکه A به شکل Schur حقیقی پایینی و B به شکل Schur حقیقی بالایی می باشد.

SYMSLV: معادله (۵-۲۱) را حل می کند در حالیکه A به شـکل Schur حقیقـی بـالایی است.

SYSSLV: دستگاه معادله خطی را حل می کند.

۵-۵-۱- تست حل عددی معادلهٔ پواسون:

همانطور که در قسمتهای قبل توضیح داده شد، معادلهٔ پواسون پس ازگسسته سازی به فرم معادلهٔ(۵-۲۰) تبدیل می‌شود در این قسمت برای تست این مراحل و اعمال شرایط مرزی مناسب معادلهٔ پواسون(۵-۳۰) با شرایط مرزی(۵-۳۱) تا (۵-۳۴) به روش عددی حل شده و نتایج با روش تحلیلی مقایسه شده است:

$$\frac{\partial^2 f}{\partial x^2}+\frac{\partial^2 f}{\partial y^2}=\sin x(2-y^2)+e^x \qquad (۳۰-۵)$$

$$f(y)=6 \qquad x=0 \qquad (۳۱-۵)$$

$$f(y)=y^2\sin 2+e^2+5 \qquad x=2 \qquad (۳۲-۵)$$

$$f(x)=e^x+5 \qquad y=0 \qquad (۳۳-۵)$$

$$f(y)=9\sin x+e^x+5 \qquad x=2 \qquad (۳۴-۵)$$

همانطور که می دانیم این معادلهٔ دیفرانسیل دارای حل تحلیلی است که جواب آن به صورت $f(x,y)=y^2\sin x+e^x+5$ است. روش حل ارائه شده در قسمت۵-۵ از دقت مناسبی برای این حل برخوردار است. سابروتین PoissonSolver (ضمیمهٔ پایان‌نامه) وظیفهٔ حل معادلهٔ پواسون را دارد.

۵-۶- نحوهٔ اعمال شرایط مرزی

به دلیل اینکه معادلات ناویر استوکس و پیوستگی برای متغیر های محاسباتی حل می شوند شرایط مرزی بر روی متغییر های محاسباتی اعمال می شوند. معادله(۵-۹) یک معادله دیفرانسیل درجه چهار می باشد ، در نتیجه نیاز به اعمال چهار شرط مرزی داریم . مقادیر u در مرز ورودی و مرز خروجی مجموعه محاسباتی مشخص می باشد. همچنین با توجه به معادله پیوستگی $\frac{\partial u}{\partial x}$

هم در مرزهای ورودی و خروجی مجموعه محاسباتی معلوم می باشند . این شرایط مرزی به شرایط دریچلت و نیومن معروف می باشند.

در مرز خروجی هم از یک شرط مرزی جابه جایی استفاده شده است . در مرز خروجی نباید هیچگونه برگشت جریان و یا وجود تاثیرات خروجی به داخل شبکه محاسباتی باشیم . در این مرز از معادله جابه جایی برای تولید شرط مرزی دریچلت برای هر دو مولفه سرعت استفاده می کنیم که معادله آن به صورت زیر می باشد :

$$\frac{\partial \Psi}{\partial t} = -c \frac{\partial \Psi}{\partial x} \qquad (۳۵-۵)$$

در این معادله ، مولفه های سرعت u و vجایگزین Ψ می گردند. ضریب c برابر با سرعت کلی انتقال موج و یا سرعت متوسط جریان در جهت اصلی در مرز خروجی است . مقدار c بین صفر و یک می باشد.

البته مقدار دقیق این پارامتر مشخص نیست و باید با تخمین مناسبی جریان را حل کرد. استفاده از سرعت انتقال موج که از تحلیلی پایداری خطی بدست می آید نیز جهت تعیین c مناسب می باشد.

۵-۷- شرط اولیه

در حالت پایدار زمانی یک پروفیل سرعت یکنواخت برای تمام مقاطع مختلف x به عنوان شرط اولیه برای جریان انتخاب شده است .

فصل ۶- معرفی نرم افزار و ارائه نتایج

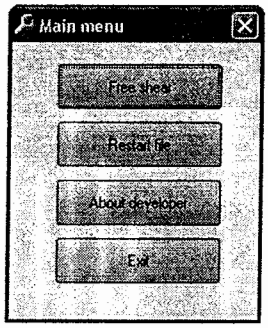
۶-۱- مقدمه

در این فصل ابتدا نرم افزار نوشته شده برای حل عددی معرفی شده و قسمتهای مختلف این نرم افزار مورد بررسی قرار می گیرد. سپس نتایج حاصل از جریان جت ،دنباله و لایهٔ اختلاطی مورد بررسی قرار خواهد گرفت.

۶-۲- معرفی نرم افزار و قسمتهای مختلف آن

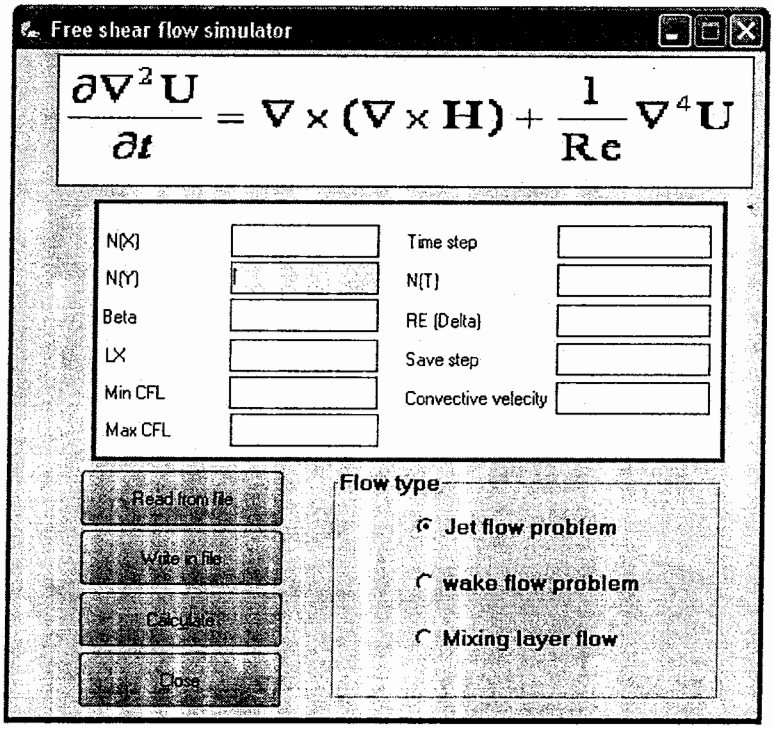
نرم افزار طراحی شده از دو قسمت عمده تشکیل شده است قسمت اول شامل حل لایهٔ مرزی روی صفحهٔ تخت و قسمت دوم شامل حل جریان برشی است این دو قسمت نرم افزار شبیه همدیگر بوده و در این قسمت نرم‌افزار جریان برشی توضیح داده می‌شود.

شکل(۶-۱) منوی اصلی این نرم افزار را نشان می‌دهد:



شکل۶-۱: منوی اصلی نرم افزار

در این منو از کاربر خواسته می‌شود تا انتخاب خود را مشخص سازد انتخاب حالت اول به این معنی است که کاربر می‌خواهد یک حل جدید را آغاز کند. با انتخاب این گزینه فرم زیر ظاهر می‌شود:



شکل۶-۲ : وارد کردن پارامترهای جدید برای حل عددی

در این فرم از کاربر خواسته می شود تا پارامترهای لازم برای حل جدید را مشخص کند این پارامترها عبارتند از :

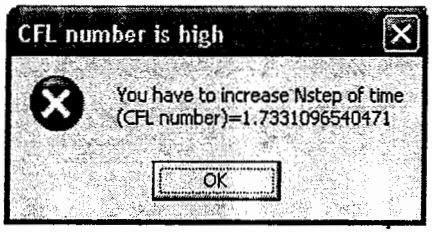
N(x): تعداد گره‌ها در جهت x

N(y): تعداد گره‌ها در جهت y

Beta: ضریب کشیدگی در جهت y است هر چه قدر مقدار این پارامتر به سمت صفر نزدیک شود تجمع گره‌ها در مرکز دامنه بیشتر می‌شود و اگر Beta بیشتر شود یک شبکهٔ تقریباً منظم خواهیم داشت.

LX: طول بی‌بعد دامنه در جهت xاست که در آن طول بوسیلهٔ پارامتر مشخص طولی بی‌بعد شده است.

Max CFL: این عدد ، معیاری است که کاربر برای حل تعیین می‌کند چنانچه عدد CFL محاسبه شده توسط نرم افزار از این معیار بزرگتر باشد نرم افزار پیغامی را به صورت شکل (۶-۳) نشان می‌دهد و از کاربر می‌خواهد که یکی از پارامترهای مربوطه را تغییر دهد.



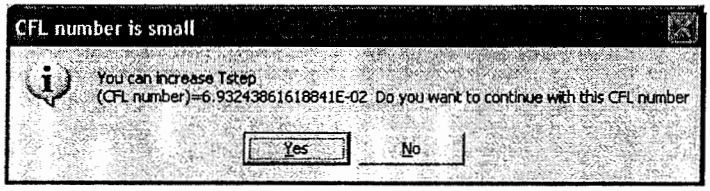
شکل۶-۳ : بزرگ بودن عدد **CFL** از معیار تعیین شده توسط کاربر

البته لازم به ذکر است که محاسبهٔ عددی CFL نیازمند رعایت نکته‌ای است و آن هم اینکه باید دقت کنیم فاصلهٔ گره‌ها در جهت y یکسان نیست. همانطور که می‌دانیم عدد CFL به صورت زیر تعریف می‌شود:

$$CFL = C\frac{\Delta t}{\min(\Delta x, (\Delta y)_{\min})} \qquad (۱-۶)$$

کوچکترین مقدار Δy در جریان برشی در مرکز جریان و برای لایهٔ مرزی نزدیک صفحه است در نتیجه باید این فاصله محاسبه شود. چنانچه مقدار CFL خیلی کوچک انتخاب شود مقدار گامهای زمانی نیز کاهش می‌یابد که این امر به نوبهٔ خود باعث طولانی شدن مدت زمان حل

خواهدشد، به این دلیل نرم افزار این قابلیت را دارد تا چنانچه عدد CFL از Min CFL کمتر باشد پیغامی را به صورت شکل (۶-۴) عنوان کند.



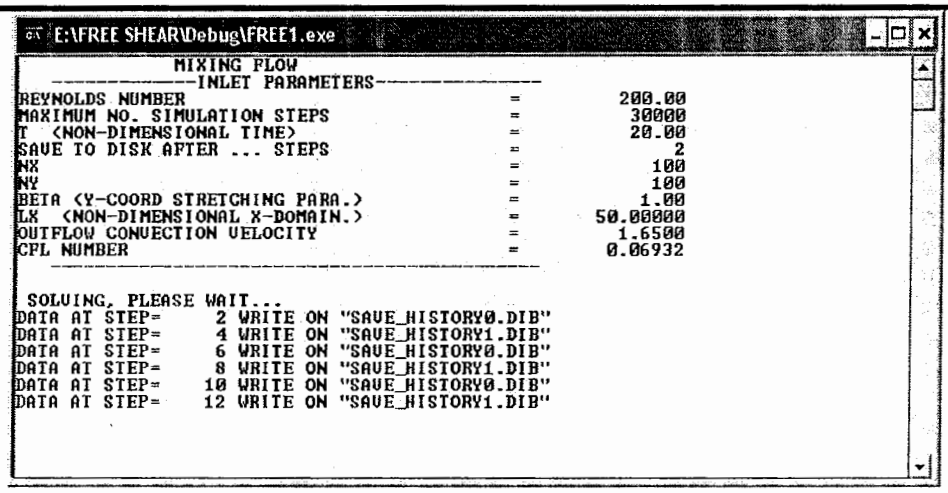
شکل۶-۴: کوچک بودن عدد **CFL** از مقدار **Min CFL**

Time step: مدت زمان بی‌بعد مسئله است که کاربر انتظار دارد تا حل عددی پیش رود.

N(T): تعداد بازه‌ها در مقیاس زمانی

Re: که همان عدد رینولدز است که در آن طول مشخصه برابر دلتا در ایستگاه اول و سرعت مشخصه برای آن مقدار سرعت بی‌بعد جریان آزاد برای لایهٔ مرزی و سرعت گره مرکزی در ایستگاه اول برای جریان جت، اختلاف سرعت جریان آزاد و سرعت گره مرکزی در ایستگاه اول برای دنباله و اختلاف سرعت ماگزیمم و مینیمم برای لایهٔ اختلاطی می‌باشد.

Save step: تعداد بازهٔ زمانی است که کاربر مایل است تا پس از سپری شدن آن نتایج حاصل از حل عددی ذخیره شود. چنانچه به هر دلیلی اعم از قطع برق یا خرابی سیستم محاسبات متوقف شود، کاربر می‌تواند از نتایج ذخیره شده در این فایل به عنوان شرط اولیه استفاده کرده و حل را ادامه دهد. نرم‌افزار مطابق شکل (۶-۵) بعد از اتمام هر مرحلهٔ مشخص شده توسط این پارامتر گزارش خود را بر روی صفحهٔ کنسول چاپ می‌کند.



شکل۶-۵: گزارش ذخیره اطلاعات حاصل از حل عددی در بازه‌های مشخص شده

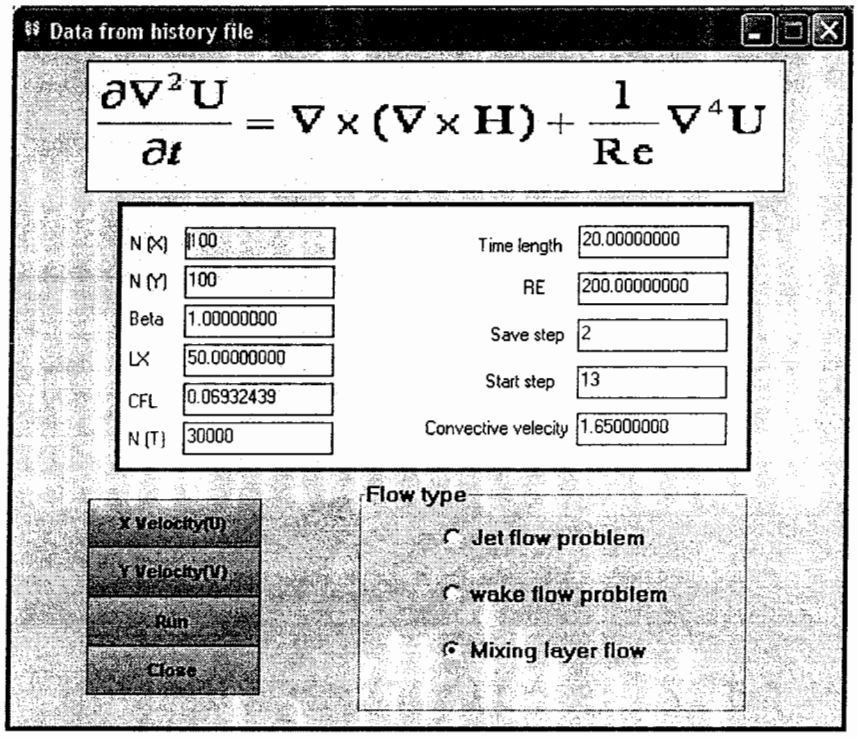
Convective velocity: سرعت انتقال دهندهٔ سیال به بیرون از دامنه است که برای مسائل مختلف یک مقدار تقریبی دارد این مقدار برای جریانهای مختلف با تقریب خوب به صورت انتقال تودهٔ بزرگ (Large structure)معرفی می‌شود.

چنانچه کاربر بخواهد اطلاعات ورودی را در یک فایل متن ذخیره کندمی‌تواند از دکمهٔ ذخیره (Write in file) استفاده کند. فرمت ذخیره اطلاعات به صورت شکل(۶-۶) است.

```
Reynolds Number                          =100
Maximum No. Simulation Steps             =1000
T  (non-dimensional time)                =50
CFL Number                               =0.7
Save to Disk after ... steps             =2
Nx                                       =100
Ny                                       = 100
Y0 (y-coord stretching para.)            =18
LX  (non-dimensional x-domain.)          =50
Outflow Convection Velocity              =1
```

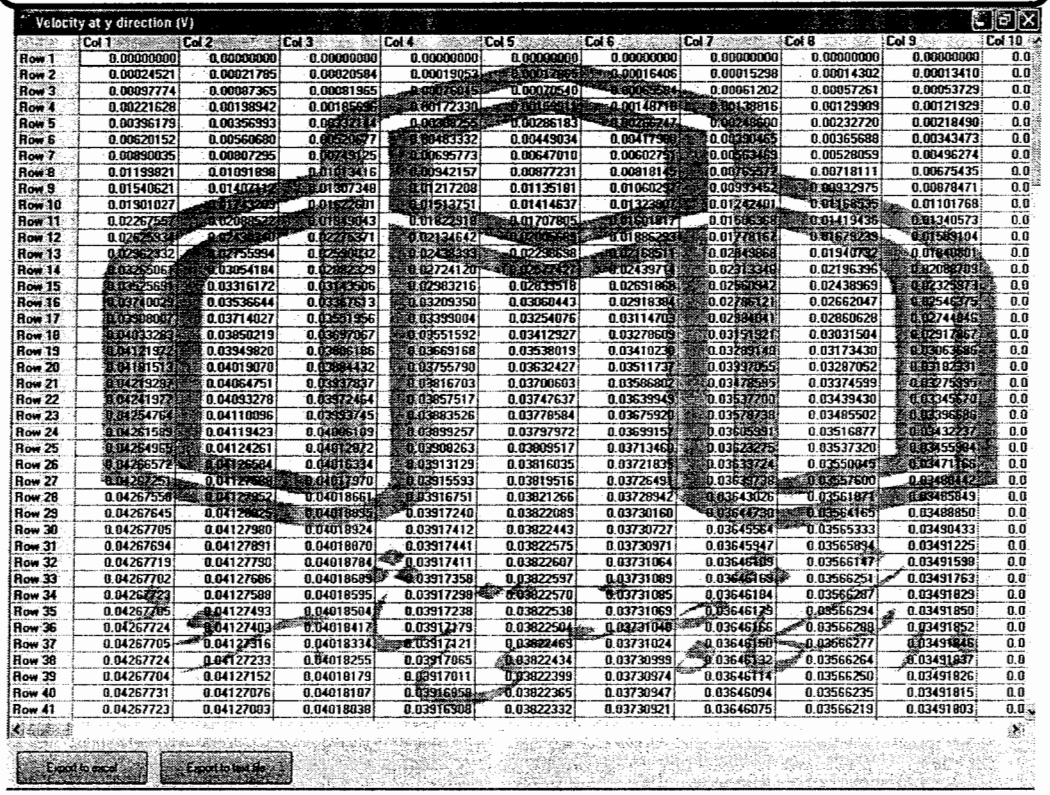
شکل۶-۶: فرمت فایل ذخیره شده توسط نرم افزار

چنانچه کاربر بر روی دکمهٔ Restart file کلیک کند بدین معنی است که تمایل دارد که حل را از یک فایل تاریخچه که با پسوند dib.* بر روی سیستم ذخیره شده است ادامه دهد. شکل (۶-۷) یک مثال از انتخاب این منو را نشان می‌دهد.



شکل۶-۷: ادامهٔ حل از فایل تاریخچه با پسوند **dib.***

در این فرم اطلاعات قابل تغییر نیست و کاربر نمی‌تواند اطلاعات موجود در جعبهٔ متن‌ها را عوض کند. با فشار دادن دکمهٔ Run ادامهٔ حل شروع می‌شود. دکمه‌های X Velocity(U) و Y Velocity(v) مولفه‌های سرعت در جهت x و y را تا لحظهٔ حل شده نمایش می‌دهد شکل (۶-۸) یک نمونه را نشان می‌دهد.



	Col 1	Col 2	Col 3	Col 4	Col 5	Col 6	Col 7	Col 8	Col 9	Col 10
Row 1	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.0
Row 2	0.00024521	0.00021785	0.00020584	0.00019053	0.00017951	0.00016406	0.00015298	0.00014302	0.00013410	0.0
Row 3	0.00097774	0.00087365	0.00081965	0.00076015	0.00070540	0.00063584	0.00061202	0.00057261	0.00053729	0.0
Row 4	0.00221628	0.00198942	0.00185696	0.00172330	0.00160113	0.00148718	0.00138816	0.00129909	0.00121929	0.0
Row 5	0.00396179	0.00356993	0.00332184	0.00308255	0.00286183	0.00266747	0.00248600	0.00232720	0.00218490	0.0
Row 6	0.00620152	0.00560680	0.00520677	0.00483332	0.00449034	0.00417796	0.00390465	0.00365688	0.00343473	0.0
Row 7	0.00890035	0.00807295	0.00749125	0.00695773	0.00647010	0.00602751	0.00563469	0.00528059	0.00496274	0.0
Row 8	0.01199821	0.01091898	0.01013416	0.00942157	0.00877231	0.00818145	0.00765577	0.00718111	0.00675435	0.0
Row 9	0.01540621	0.01407412	0.01307348	0.01217208	0.01135181	0.01060257	0.00993452	0.00932975	0.00878471	0.0
Row 10	0.01901027	0.01732095	0.01622501	0.01513751	0.01414637	0.01323907	0.01242401	0.01168335	0.01101768	0.0
Row 11	0.02267557	0.02085522	0.01949043	0.01822918	0.01707805	0.01601817	0.01506368	0.01419436	0.01340573	0.0
Row 12	0.02629344	0.02430501	0.02275371	0.02134642	0.02005459	0.01885283	0.01776164	0.01678239	0.01589104	0.0
Row 13	0.02982332	0.02755994	0.02580802	0.02423333	0.02280858	0.02148551	0.02028868	0.01919782	0.01819801	0.0
Row 14	0.03325067	0.03054184	0.02882229	0.02724120	0.02577421	0.02439774	0.02313349	0.02196396	0.02086709	0.0
Row 15	0.03652689	0.03316172	0.03143506	0.02983216	0.02833518	0.02691868	0.02560942	0.02438969	0.02325873	0.0
Row 16	0.03740029	0.03536644	0.03387513	0.03209350	0.03060443	0.02918384	0.02786121	0.02662047	0.02545279	0.0
Row 17	0.03908087	0.03714027	0.03561956	0.03399004	0.03254076	0.03114709	0.02984041	0.02850628	0.02744045	0.0
Row 18	0.04033283	0.03850219	0.03697067	0.03551592	0.03412927	0.03278609	0.03151921	0.03031504	0.02917867	0.0
Row 19	0.04121972	0.03949820	0.03806186	0.03669168	0.03538019	0.03410239	0.03288148	0.03173430	0.03063589	0.0
Row 20	0.04181513	0.04019070	0.03884432	0.03755790	0.03632427	0.03511737	0.03397055	0.03287052	0.03182231	0.0
Row 21	0.04219282	0.04064751	0.03937837	0.03816703	0.03700603	0.03586802	0.03478585	0.03374599	0.03275397	0.0
Row 22	0.04241972	0.04093278	0.03972464	0.03857517	0.03747637	0.03639943	0.03537700	0.03439430	0.03345570	0.0
Row 23	0.04254764	0.04110096	0.03993745	0.03883526	0.03776584	0.03675920	0.03578738	0.03485502	0.03396586	0.0
Row 24	0.04261589	0.04119423	0.04006109	0.03899257	0.03797972	0.03699157	0.03605391	0.03516877	0.03432237	0.0
Row 25	0.04264959	0.04124261	0.04012872	0.03908263	0.03809517	0.03713460	0.03623275	0.03537320	0.03455984	0.0
Row 26	0.04266572	0.04126504	0.04016334	0.03913129	0.03816035	0.03721836	0.03633724	0.03550045	0.03471160	0.0
Row 27	0.04267251	0.04127504	0.04017970	0.03915593	0.03819516	0.03726491	0.03637738	0.03557600	0.03480442	0.0
Row 28	0.04267556	0.04127952	0.04018661	0.03916751	0.03821266	0.03728942	0.03643026	0.03561871	0.03485849	0.0
Row 29	0.04267645	0.04128025	0.04018895	0.03917240	0.03822089	0.03730160	0.03644730	0.03564165	0.03488850	0.0
Row 30	0.04267705	0.04127980	0.04018924	0.03917412	0.03822443	0.03730727	0.03645584	0.03565333	0.03490433	0.0
Row 31	0.04267694	0.04127891	0.04018870	0.03917441	0.03822575	0.03730971	0.03645947	0.03565894	0.03491225	0.0
Row 32	0.04267719	0.04127790	0.04018784	0.03917411	0.03822607	0.03731064	0.03646108	0.03566147	0.03491590	0.0
Row 33	0.04267702	0.04127686	0.04018689	0.03917358	0.03822597	0.03731089	0.03646164	0.03566251	0.03491763	0.0
Row 34	0.04267723	0.04127588	0.04018595	0.03917298	0.03822570	0.03731085	0.03646184	0.03566287	0.03491829	0.0
Row 35	0.04267705	0.04127493	0.04018504	0.03917238	0.03822538	0.03731069	0.03646179	0.03566294	0.03491850	0.0
Row 36	0.04267724	0.04127403	0.04018417	0.03917179	0.03822504	0.03731048	0.03646166	0.03566288	0.03491852	0.0
Row 37	0.04267705	0.04127316	0.04018334	0.03917121	0.03822469	0.03731024	0.03646150	0.03566277	0.03491845	0.0
Row 38	0.04267724	0.04127233	0.04018255	0.03917065	0.03822434	0.03730999	0.03646132	0.03566264	0.03491837	0.0
Row 39	0.04267704	0.04127152	0.04018179	0.03917011	0.03822399	0.03730974	0.03646114	0.03566250	0.03491826	0.0
Row 40	0.04267731	0.04127076	0.04018107	0.03916958	0.03822365	0.03730947	0.03646094	0.03566235	0.03491815	0.0
Row 41	0.04267723	0.04127003	0.04018038	0.03916908	0.03822332	0.03730921	0.03646075	0.03566219	0.03491803	0.0

شکل۶-۸: مقادیر سرعت در جهت y در گره‌های مختلف

کاربر می‌تواند نتایج را برای بررسی بیشتر به نرم افزار Excel ارسال کند این عمل با فشار دادن دکمهٔ Export to excel امکان‌پذیر است.

بعد از اتمام حل نرم افزار در پوشهٔ محل نصب خود شش فایل با فرمت Tecplot ایجاد می‌کند که به صورت زیر است:

۱-self_selmilarity.dat که در آن اطلاعات تشابهی هر ایستگاه موجود است. یعنی در هر ایستگاه y به دلتای آن ایستگاه و u به سرعت مشخصهٔ آن ایستگاه تقسیم شده است.

۲-delta.dat که در این فایل موقعیت دلتای هر ایستگاه ذخیره شده است.

۳-uvmesh.dat که در این فایل مختصات گره‌ها و مولفه‌های سرعت و ورتیسیتی در هر گره ذخیره شده است.

۴-Transit_u.dat که در این فایل تاریخچهٔ زمانی مولفهٔ u سرعت ذخیره شده است.

۵-Transit_v.dat که در این فایل تاریخچهٔ زمانی مولفهٔ v سرعت ذخیره شده است.

۶-Transit_w.dat که در این فایل تاریخچهٔ زمانی گردابه ذخیره شده است.

۶-۳- گردابه های استوارت

استوارت[1] [18]حل دو بعدی وابسته به زمان معادله ناویر استوکس غیر لزج[2] را برای گروه لایه‌های اختلاطی معرفی کرد. حل به صورت تابع جریان ψ است که مولفه‌های سرعت $u = \frac{\partial \psi}{\partial y}$ و $v = -\frac{\partial \psi}{\partial x}$ است.

$$\psi(x, y, t) = cy + \ln(a\cosh(y - y_0) + b\cos(x - ct)) \qquad (۱-۶)$$

در اینجا $b = \sqrt{a^2 - 1}$. به راحتی می‌توان نشان داد که معادله بالا، معادله جابجایی را ارضا می‌کند که c برابر سرعت جابجایی موج است. در نتیجه حل استوارت می‌تواند برای ارزیابی صحت شرط مرزی خروجی جابجایی مورد استفاده قرار گیرد. در این تست تشکیل ترم لزج تست نمی‌شود ولی برای ارزیابی تشکیل ترم‌های غیرخطی و پیشروی محاسبات در زمان مناسب است.

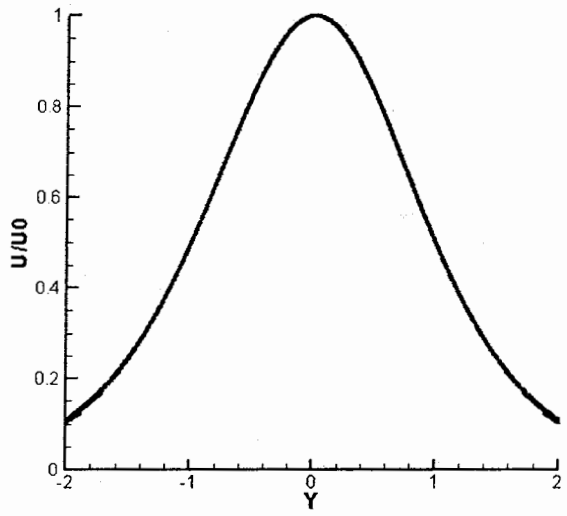
به این نکته توجه کنید که مولفه سرعت در جهت جریان نمی تواند حل پایداری داشته باشد. برای رفع این مشکل $\tanh(y - y_0)$ را به آن اضافه و کسر می‌کنیم. قسمت پایدار آن یعنی $a\sinh(y - y_0)/(a\cosh(y - y_0) + b\cos(x - ct)) - \tanh(y - y_0)$ را به صورت شرط اولیه و شرط مرزی ورودی در نظر می‌گیریم. باقیمانده عبارت به عنوان جریان اولیه محسوب می‌شود.

پارامتر ها به صورت $L_x = 2\pi$ و $\beta = 3$ و $\mathrm{Re} = 10^9$ در نظر گرفته می شود تا جریان بصورت موثر ایده ال باقی بماند. به عبارت دیگر با در نظر گرفتن $\mathrm{Re} = 10^9$ جمله غیر خطی در مقایسه با جمله لزجتی کاملا غالب خواهد بود.

---

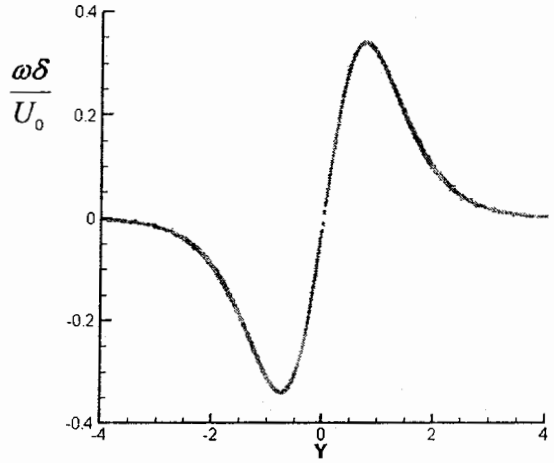
[1] Stewart
[2] Inviscid

همانطور که قبلاً توضیح داده شد طبق نتایج تئوری، جت خود تشابه است. شکل (۶-۱۱) این موضوع را نشان می‌دهد که سرعت بوسیلهٔ سرعت خط مرکزی و y با نیم عرض جت نرمال شده است.



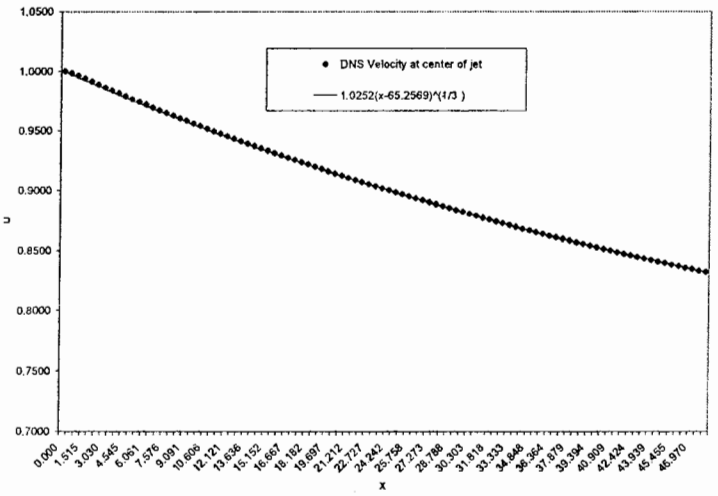
شکل ۶-۱۱: پروفیل سرعت **u** در مختصات خود تشابه برای شبیه سازی جت بدون اغتشاش ورودی

شکل (۶-۱۲) خود تشابهی گردابه‌های جت را نشان می‌دهد در این شکل گردابه بوسیلهٔ $\frac{\delta}{U_0}$بی‌بعد شده است که در آن δنیم عرض جت و U_0سرعت خط مرکزی در هر ایستگاه می‌باشد.



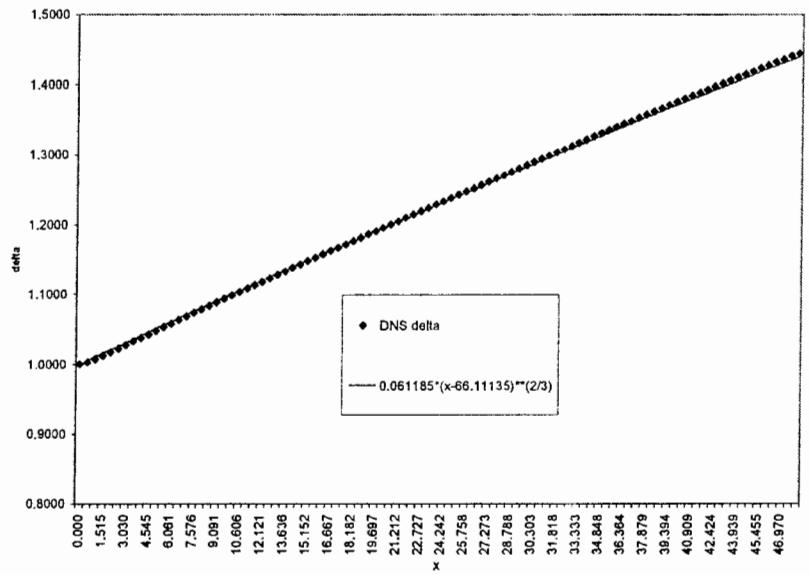
شکل ۶-۱۲: پروفیل ωدر مختصات خود تشابه برای شبیه سازی جت بدون اغتشاش ورودی

طبق نتایج تئوری جت سرعت خط مرکزی در جریان جت دو بعدی آرام متناسب بـا $x^{-1/3}$ است. در نمودار ۶-۱۳ می‌توان این وابستگی را بخوبی مشاهده کرد.



شکل ۶-۱۳: سرعت خط مرکزی برای شبیه سازی جت بدون اغتشاش ورودی

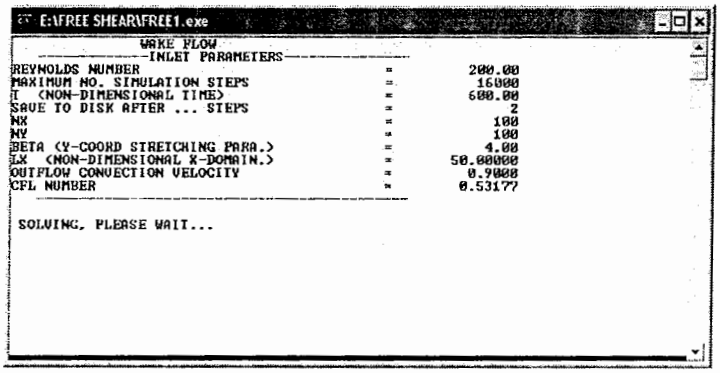
همچنین نیم عرض جت یعنیb/2 هم متناسب با $x^{2/3}$است که شـکل(۶-۱۴) تـصدیقی بـر این موضوع است.



شکل ۶-۱۴: ضخامت نیم عرض جت و مقایسه با روش تحلیلی برای شبیه سازی جت بدون اغتشاش ورودی

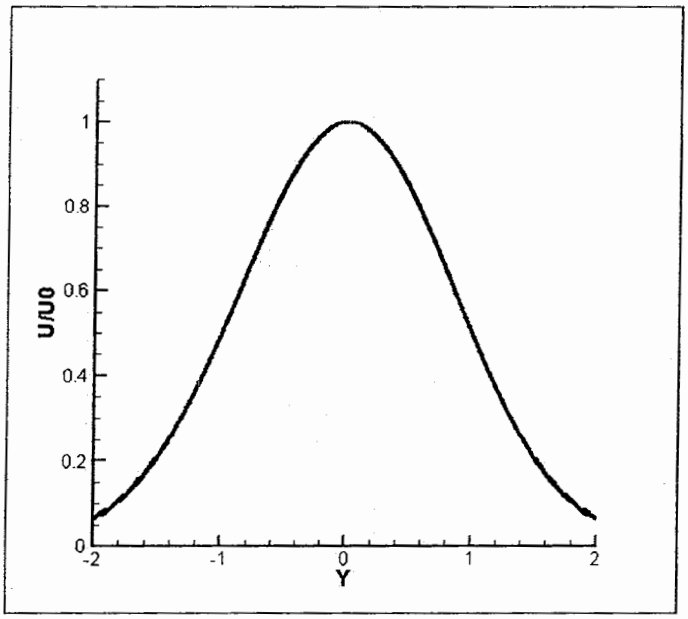
۶-۵-نتایج حل عددی برای دنباله

برای شبیه سازی دنباله اطلاعات به صورت زیر به نرم افزار معرفی شده است:



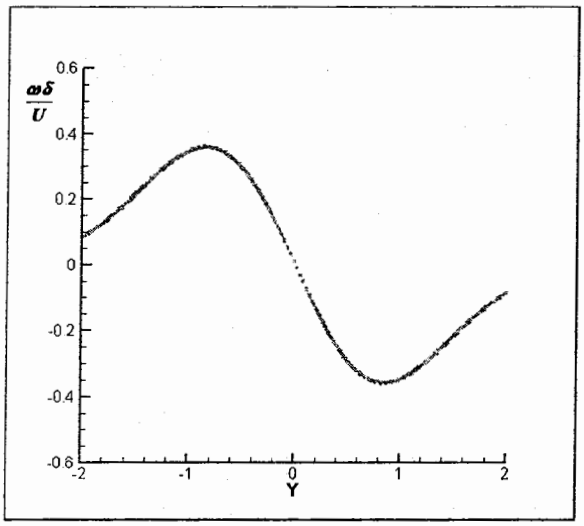
شکل ۶-۱۵: پارامترهای ورودی برای شبیه سازی دنباله بدون اغتشاش ورودی

همانطور که قبلاً توضیح داده شد طبق نتایج تئوری دنباله خود تشابه است. شکل (۶-۱۶) این موضوع را نشان می‌دهد که سرعت بوسیلهٔ سرعت خط مرکزی جت مکمل و y با نیم عرض جت مکمل نرمال شده است.



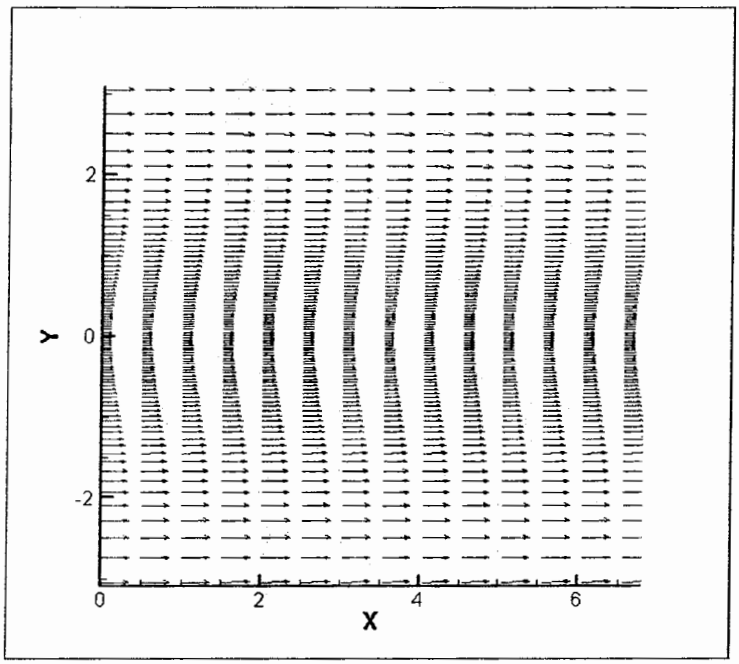
شکل ۶-۱۶: پروفیل سرعت **u** در مختصات خود تشابه برای شبیه سازی دنباله بدون اغتشاش ورودی

شکل (۶-۱۷) خود تشابهی گردابه‌های دنباله را نشان می‌دهد.



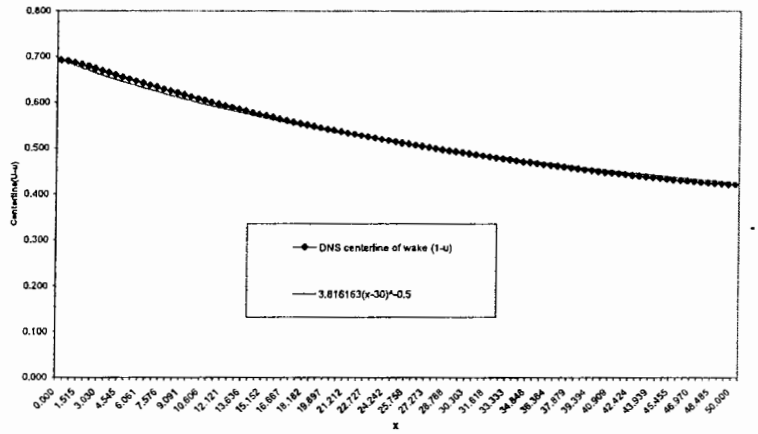
شکل ۶-۱۷: پروفیل ω در مختصات خود تشابه برای شبیه سازی دنباله بدون اغتشاش ورودی

شکل ۶-۱۸ بردارهای سرعت را برای شبیه سازی دنباله در ایستگاههای مختلف نشان می‌دهد



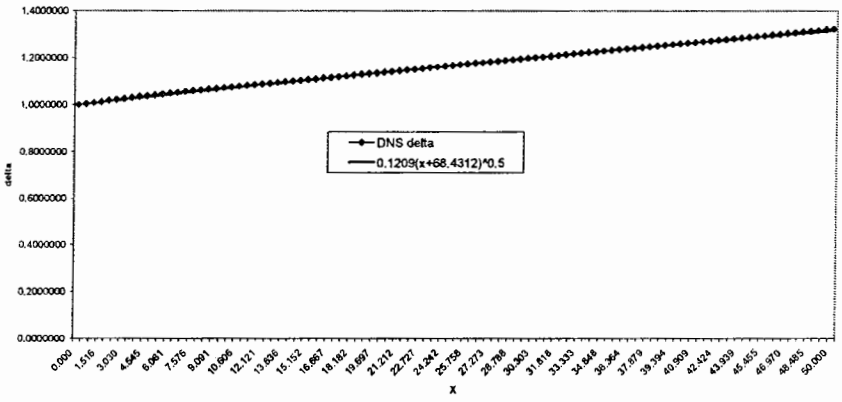
شکل ۶-۱۸: بردار سرعت برای شبیه سازی دنباله بدون اغتشاش ورودی

طبق نتایج تئوری دنباله سرعت خط مرکزی در جریان دنباله دو بعدی آرام متناسب با $x^{-1/2}$ است. در نمودار ۶-۱۹ می‌توان این وابستگی را بخوبی مشاهده کرد.



شکل ۶-۱۹: سرعت خط مرکزی برای شبیه سازی دنباله بدون اغتشاش ورودی

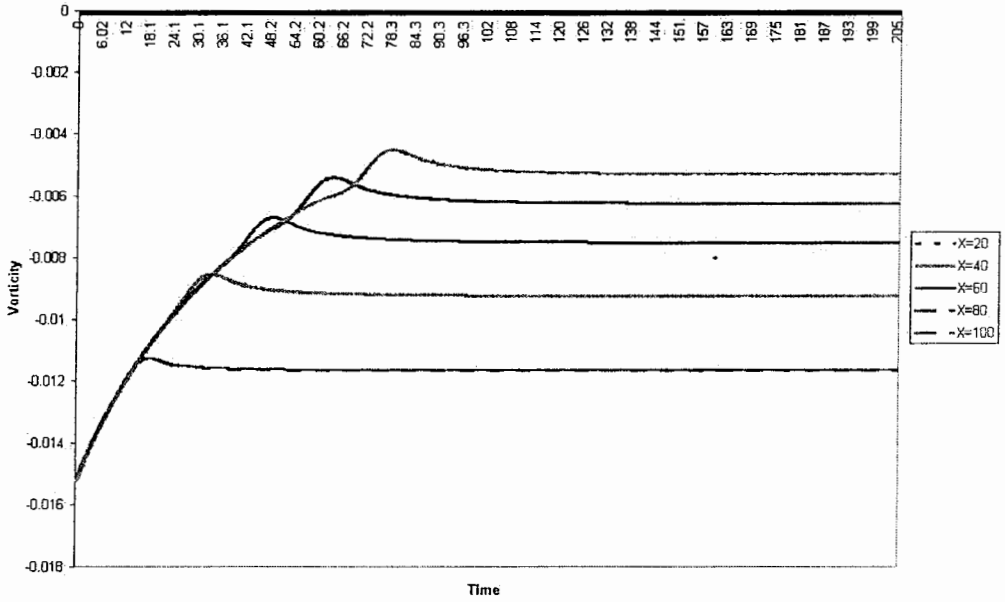
همچنین نیم عرض دنباله یعنی$b/2$ هم متناسب با $x^{1/2}$است که شکل(۶-۲۰) تصدیقی بر این موضوع است.



شکل ۶-۲۰: ضخامت نیم عرض دنباله و مقایسه با روش تحلیلی برای شبیه سازی دنباله بدون اغتشاش ورودی

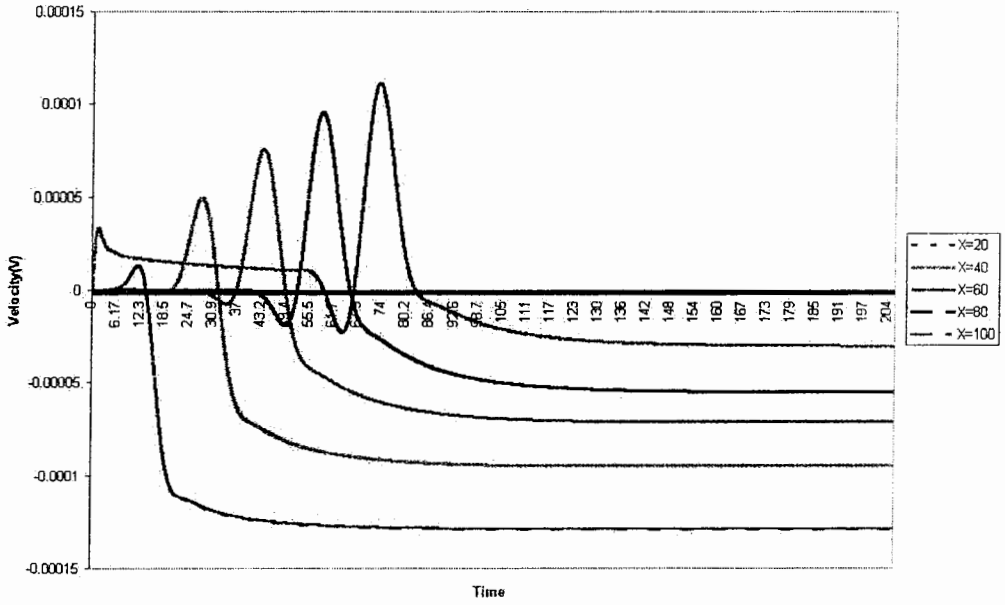
شکل (۶-۲۱) تاریخچهٔ زمانی سرعتU در ایستگاههای مختلف را نشان می دهد. همانطور که انتظار داریم نمودارهای تاریخچهٔ زمانی برای شبیه سازی بدون اغتشاش ورودی بعد از سپری شدن

مدت زمانی به یک حالت پایدار می‌رسد شکل(۶-۲۱) تا (۶-۲۳) این موضوع را به خوبی نشان می‌دهد.



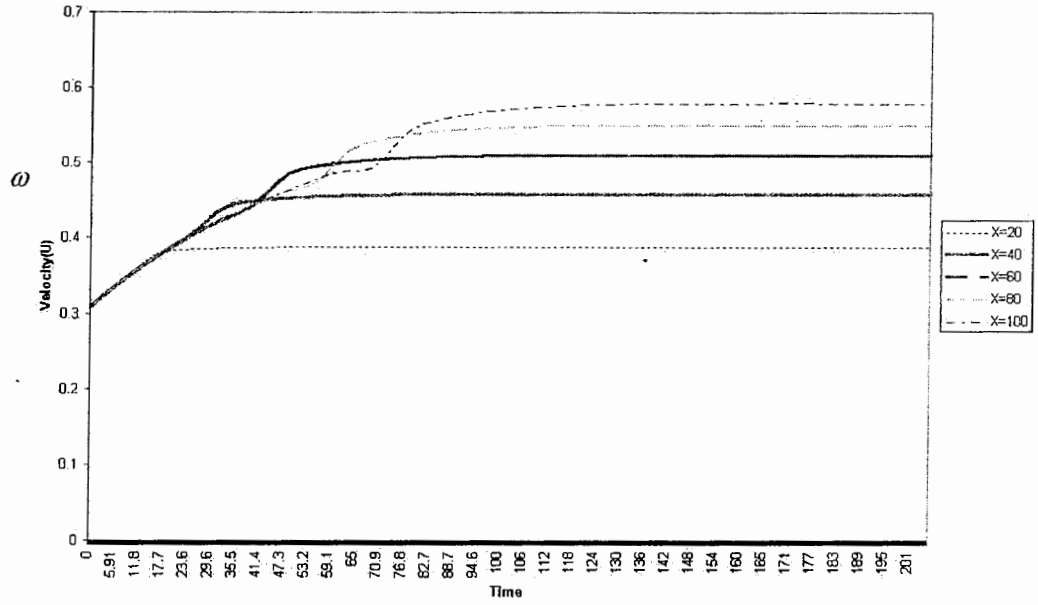
شکل ۶-۲۱: تاریخچهٔ زمانی سرعت **u** در ایستگاههای مختلف برای شبیه سازی دنباله بدون اغتشاش ورودی

شکل (۶-۲۲) تاریخچهٔ زمانی سرعت V در ایستگاههای مختلف را نشان می دهد.



شکل ۶-۲۲: تاریخچهٔ زمانی سرعت**v** در ایستگاههای مختلف برای شبیه سازی دنباله بدون اغتشاش ورودی

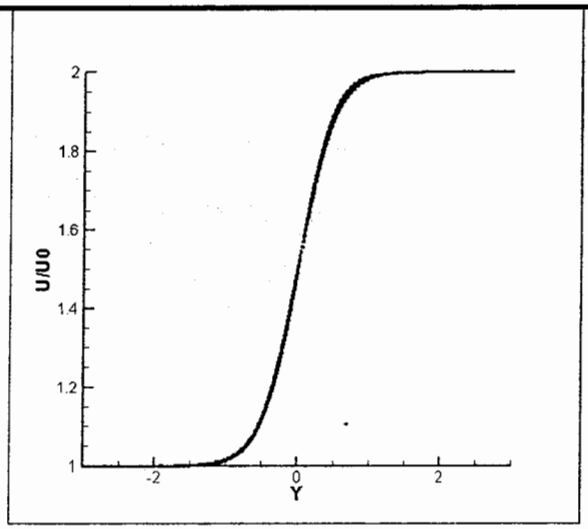
شکل (۶-۲۳) تاریخچهٔ زمانی گرادبه ها در ایستگاههای مختلف را نشان می دهد.



شکل ۶-۲۳: تاریخچهٔ زمانی گردابه در ایستگاههای مختلف برای شبیه سازی دنباله بدون اغتشاش ورودی

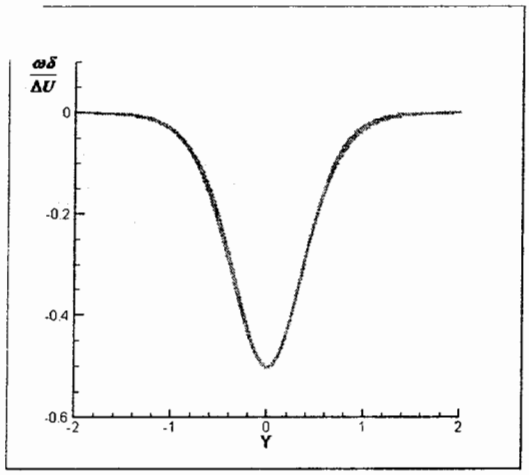
۶-۶-نتایج حل عددی برای لایهٔ اختلاطی

همانطور که قبلاً توضیح داده شد طبق نتایج تئوری، جریان اختلاطی خود تشابه است. شکل (۶-۲۴) این موضوع را نشان می‌دهد که سرعت بوسیلهٔ اختلاف سرعت و y با فرمول ارائه شده در فصل ۱ (۱-۶۲) نرمال شده است.



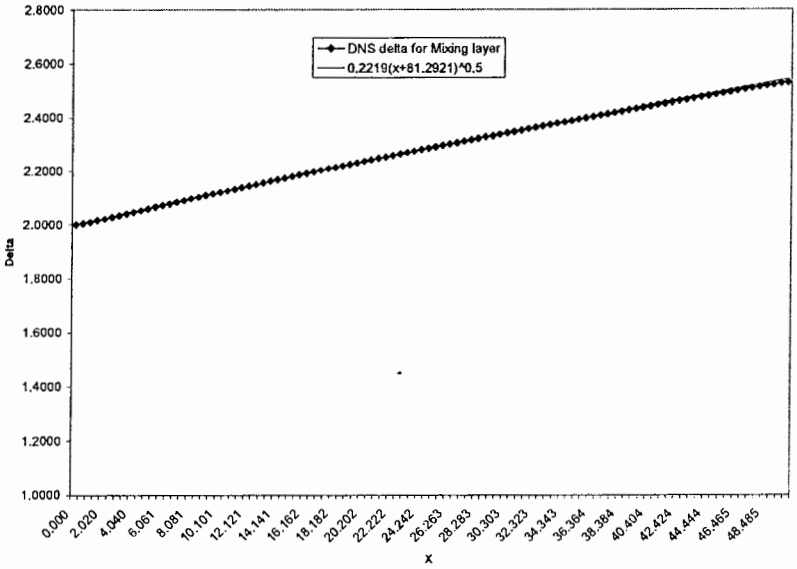
شکل ۶-۲۴: پروفیل سرعت **u** در مختصات خود تشابه برای شبیه سازی لایهٔ اختلاطی بدون اغتشاش ورودی

شکل (۶-۲۵) خود تشابهی گردابه‌های لایهٔ اختلاطی را نشان می‌دهد.



شکل ۶-۲۵: پروفیل ω در مختصات خود تشابه برای شبیه سازی لایهٔ اختلاطی بدون اغتشاش ورودی

طبق نتایج تئوری اختلاطی سرعت خط مرکزی در جریان لایه اختلاطی دو بعدی آرام متناسب با $x^{1/2}$ است. در نمودار ۶-۲۶ می‌توان این وابستگی را بخوبی مشاهده کرد.



شکل ۶-۲۶: سرعت خط مرکزی برای شبیه سازی لایهٔ اختلاطی بدون اغتشاش ورودی

ضمیمهٔ الف) تحلیل خطا

به منظور تعیین مرتبهٔ دقت یک روش عددی برای ارزیابی یک تابع ، بسط سری تیلور مورد استفاده قرار می‌گیرد . اجازه بدهید خطای عددی در یک نقطهٔ خاص از یک مجموعه را به صورت اختلاف بین مقدار حقیقی و مقدار تقریبی تابع در آن نقطه تعریف کنیم . در تقریب‌های عددی با استفاده از سری تعدای از جملات بسط سری تیلور مورد استفاده قرار می گیرد. اولین ترم بریده شده از بسط در تقریب عددی می‌توان تخمین خیلی خوبی برای خطا باشد که می‌تواند مرتبهٔ دقت را نشان دهد. اگر

$$E(x_i) \cong \frac{(L/N)^n}{n!} f^{(n)}(x_i) \qquad \text{(الف-۱)}$$

که $E(x_i)$، خطا در ارزیابی $f(x_i)$ در x_i است و L و N به ترتیب برابر طول و تعداد تقسیمات در دامنه می باشند.

با گرفتن لگاریتم از هر دو طرف معادلهٔ (الف-۱) به معادلهٔ (الف-۲) می رسیم:

$$\log(E_{\max}) \cong C - n\log(N) \qquad \text{(الف-۲)}$$

که C یک ثابت است. معادلهٔ (الف-۲) به وضوح نشان می دهد که شیب خط در نمودار $\log(E_{\max})$ بر حسب $\log N$ برابر n– میباشد که برابر مرتبهٔ خطای تقریب می باشد.

ضمیمهٔ ب) ماتریس هسنبرگ

تعریف ماتریس هسنبرگ(**Hessenberg Matrix**)

-ماتریس بالا هسنبرگی

به یک ماتریس مربعی $n\times n$ ماتریس بالا هسنبرگی گوییم هرگاه بجز اولین قطر زیر قطر اصلی، بقیه داریه های زیر قطر اصلی صفر باشد به عبارت دیگر اگر $i>j+1$ آنگاه داشته باشیم $a_{i,j}=0$:

$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \cdots & a_{1,n-3} & a_{1,n-2} & a_{1,n-1} & a_{1,n} \\ a_{2,1} & a_{2,2} & a_{2,3} & \cdots & a_{2,n-3} & a_{2,n-2} & a_{2,n-1} & a_{2,n} \\ 0 & a_{3,2} & a_{3,3} & \cdots & a_{3,n-3} & a_{3,n-2} & a_{3,n-1} & a_{3,n} \\ 0 & 0 & a_{4,3} & \cdots & a_{4,n-3} & a_{4,n-2} & a_{4,n-1} & a_{4,n} \\ 0 & 0 & 0 & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & a_{n-2,n-3} & a_{n-2,n-2} & a_{n-2,n-1} & a_{n-2,n} \\ 0 & 0 & 0 & 0 & 0 & a_{n-1,n-2} & a_{n-1,n-1} & a_{n-1,n} \\ 0 & 0 & 0 & 0 & 0 & 0 & a_{n,n-1} & a_{n,n} \end{bmatrix}$$

-ماتریس پایین هسنبرگی

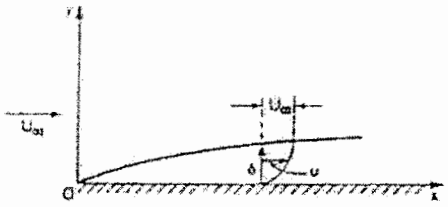
به یک ماتریس مربعی $n\times n$ ماتریس پایین هسنبرگی گوییم هرگاه بجز اولین قطر زیر قطر اصلی، بقیه داریه های زیر قطر اصلی صفر باشد به عبارت دیگر اگر $i<j-1$ آنگاه داشته باشیم: $a_{i,j}=0$

$$\begin{bmatrix} a_{1,1} & a_{1,2} & 0 & 0 & 0 & 0 & 0 & 0 \\ a_{2,1} & a_{2,2} & a_{2,3} & 0 & 0 & 0 & 0 & 0 \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & 0 & 0 & 0 \\ a_{n-3,1} & a_{n-3,2} & a_{n-3,3} & a_{n-3,4} & \cdots & a_{n-3,n-2} & 0 & 0 \\ a_{n-2,1} & a_{n-2,2} & a_{n-2,3} & a_{n-2,4} & \cdots & a_{n-2,n-2} & a_{n-2,n-1} & 0 \\ a_{n-1,1} & a_{n-1,2} & a_{n-1,3} & a_{n-1,4} & \cdots & a_{n-1,n-2} & a_{n-1,n-1} & a_{n-1,n} \\ a_{n,1} & a_{n,2} & a_{n,3} & a_{n,4} & \cdots & a_{n,n-2} & a_{n,n-1} & a_{n,n} \end{bmatrix}$$

ضمیمهٔج) لایهٔ مرزی روی صفحهٔ تخت

ج-۱- حل تحلیلی معادلهٔ بلازیوس

جریان عبوری سیال لزج وتراکم ناپذیر را که از روی صفحه نازک با سرعت یکنواخت U_∞ می گذرد را مطابق شکل (ج-۱) در نظر می گیریم .طول صفحه بی نهایت می باشد .از آنجایی که مسئله بصورت دو بعدی می باشد می توان محاسبات را با آنالیز کردن معادلات لایه مرزی پرانتل پیگیری کرد .



شکل ج-۱: لایه مرزی روی صفحهٔ تخت

همانطور که می دانیم معادلهٔ لایهٔ مرزی پرانتل بصورت رابطهٔ(ج-۱) بیان می شود :

$$u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}=\upsilon\frac{\partial^2 u}{\partial y^2}$$
$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}=0 \qquad (ج-۱)$$

شرایط مرزی معادلات بالا به صورت زیر می باشد :

$$y=0 \qquad u=v=0$$
$$y=\infty \qquad u=U_\infty \qquad (ج-۲)$$

پارامترهای مشخصه در این مسئله x,y,υ,U_∞ می باشند که توسط این پارامتر ها مسئله بطور کامل تعریف می شود . مطابق با قانون تشابه سازی می توان پروفیل سرعت را بصورت معادله (ج-۳) فرض نمود .

$$\frac{u}{U_\infty} = F(y, x, \upsilon, U_\infty) = F(\eta) \qquad (ج-۳)$$

مطابق با حرکت برروی یک صفحهٔ تخت داریم:

$$\delta \sim \sqrt{vt} \sim \sqrt{\frac{vx}{U_\infty}} \qquad (ج-۴)$$

زمان t در این رابطه بدین معنی است کـه ذرات سـیال ، یـک مـسافت x ای را بـا سـرعت U_∞ طی می‌کند .

پارامتر مسافت را می توان بصورت بدون بعد بصورت معادله (ج-۵) نمایش داد که با معادلهٔ (ج-۳) مطابقت دارد .

$$\eta = \frac{y}{\delta} = \frac{y}{\sqrt{vx/U_\infty}} \qquad (ج-۵)$$

حال می توان تابع جریان را از مولفه‌های سرعت در معادله (ج-۳) بدست آورد .

$$\psi = \int u dy = \frac{U_\infty}{\sqrt{U_\infty / vx}} \int f(\eta) d\eta = \sqrt{U_\infty vx} f(\eta) \qquad (ج-۶)$$

اجزا سرعت ومشتق های آن بصورت زیر تعریف می شود .(دیفرانسیل های مربوط بـه η بـا پرایم نمایش داده شده‌اند .)

$$u = \frac{\partial \psi}{\partial y} = \frac{\partial \psi}{\partial \eta}\frac{\partial \eta}{\partial y} = U_\infty f'(\eta) \qquad (ج-۷)$$

$$v = -\frac{\partial \psi}{\partial x} = -\frac{1}{2}\sqrt{\frac{vU_\infty}{x}} f(\eta) - \sqrt{U_\infty vx}\left(-\frac{\eta}{2x}\right) f'(\eta)$$
$$= \frac{1}{2}\sqrt{\frac{vU_\infty}{x}}\left[\eta f'(\eta) - f(\eta)\right] \qquad (ج-۸)$$

$$\frac{\partial u}{\partial x} = \frac{\partial^2 \psi}{\partial x \partial y} = -\frac{U_\infty}{2}\frac{\eta}{x} f''(\eta) \qquad (ج-۹)$$

$$\frac{\partial u}{\partial y} = U_\infty \sqrt{\frac{U_\infty}{vx}} f''(\eta) \qquad (ج-۱۰)$$

$$\frac{\partial^2 u}{\partial y^2} = \frac{U_\infty^2}{vx} f'''(\eta) \qquad (ج-۱۱)$$

که در روابط بالا:

$$\frac{\partial \eta}{\partial y} = \sqrt{\frac{U_\infty}{vx}} \quad , \quad \frac{\partial \eta}{\partial x} = -\frac{1}{2}\frac{y}{x}\sqrt{\frac{U_\infty}{vx}} = -\frac{1}{2}\frac{\eta}{x} \qquad (ج-۱۲)$$

با جای گذاری معادله (ج-۱۱) در معادله (ج-۱)

$$-\frac{U_\infty^2}{2}\frac{\eta}{x} f' f'' + \frac{U_\infty^2}{2x}\left[\eta f' - f\right] f'' = \frac{U_\infty^2}{x} f''' \qquad (ج-۱۳)$$

بعداز ساده سازی معادله (ج-۱۴) به رابطه زیر خواهیم رسید.

$$2\frac{d^3 f}{d\eta^3} + f\frac{d^2 f}{d\eta^2} = 0 \qquad (ج-۱۴)$$

شرایط مرزی موجود در رابطهٔ (ج-۲) به صورت جملات f و η بصورت

$$\eta = 0: \qquad f = 0 \qquad \frac{df}{d\eta} = 0$$
$$\eta = \infty: \qquad \frac{df}{d\eta} = 1 \qquad (ج-۱۵)$$

تغییر می یابد.

معادله (ج-۱۵) یک معادله غیر خطی ازمرتبه سوم می باشد ، که یک راه حل معینی ندارد .بلازیوس در سال ۱۹۰۸ توانست جواب معادله (ج-۱۴) را بصورت سری توانی ، بوسیله بسط حول $\eta = 0$ بدست آورد که به صورت زیر است .

$$f = A_0 + A_1\eta + \frac{A_2}{2!}\eta^2 + \frac{A_3}{3!}\eta^3 + \ldots\ldots \qquad (ج-۱۶)$$

$$f' = A_1 + A_2\eta + \frac{A_3}{2!}\eta^2 + \frac{A_4}{3!}\eta^3 + \ldots \qquad (ج-۱۷)$$

$$f'' = A_2 + A_3\eta + \frac{A_4}{2!}\eta^2 + \frac{A_5}{3!}\eta^3 + \ldots \qquad (ج-۱۸)$$

$$f'''' = A_3 + A_4\eta + \frac{A_5}{2!}\eta^2 + \frac{A_6}{3!}\eta^3 + \ldots \qquad (ج-۱۹)$$

با اعمال شرایط مرزی موجود در رابطهٔ(ج-۱۵) داریم:

$$A_0 = 0 \quad , \quad A_1 = 0 \qquad (ج-۲۰)$$

با جاگذاری f و f'' و f''' در معادلهٔ (ج-۱۴) رابطهٔ زیر بدست می‌آید:

$$2A_3 + (2A_4)\eta + (A_2^2 + 2A_5)\frac{\eta^2}{2!} + (4A_2A_4 + 2A_6)\frac{\eta^3}{3!} + \ldots = 0 \qquad (ج-۲۱)$$

برای اینکه عبارت بالا یک عبارت صحیح باشد باید ضرایب مختلف از η برابر با صفر باشد بنابراین:

$$\begin{aligned} &A_3 = A_4 = A_6 = A_7 = 0 \\ &A_5 = 0 \\ &A_8 = \frac{11}{4}A_2^3 \end{aligned} \qquad (ج-۲۲)$$

با جاگذاری (ج-۲۰) و(ج-۲۲) در رابطهٔ (ج-۱۶) مقدار f بر حسب A_2 و η بدست می‌آید :

$$f = \frac{A_2}{2!}\eta^2 - \frac{1}{2}\frac{A_2^2}{5!}\eta^5 + \frac{1}{4}\frac{11A_2^3}{8!}\eta^8 - \frac{1}{8}\frac{375A_2^4}{11!}\eta^{11} + \frac{1}{16}\frac{27{,}897A_2^5}{141}\eta^{14} + \ldots \qquad (ج-۲۳)$$

رابطهٔ(ج-۲۳) دو شرط مرزی را در $\eta = 0$ ارضا می‌کند ، در حالیکه مقدار A_2 را می‌توان از شرط مرزی دیگر در $\eta = \infty$ بدست آورد.

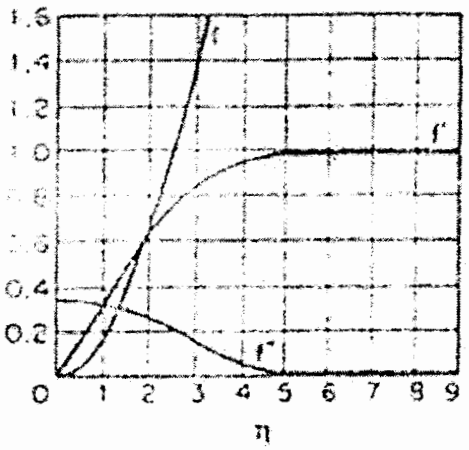
معادلهٔ (ج-۲۳) را به صورت زیر بازنویسی می‌کنیم:

$$\begin{aligned} f &= A_2^{1/3}\left[\frac{\left(A_2^{1/3}\eta\right)^2}{2!} - \frac{1}{2}\frac{\left(A_2^{1/3}\eta\right)^5}{5!} + \frac{1}{4}\frac{11\left(A_2^{1/3}\eta\right)^8}{8!} - \frac{1}{8}\frac{375\left(A_2^{1/3}\eta\right)^{11}}{11!} + \ldots\right] \\ &= A_2^{1/3}F(A_2^{1/3}\eta) \end{aligned} \qquad (ج-۲۴)$$

با اعمال شرط مرزی باقیمانده در $\eta = \infty$ برای معادلهٔ بالا داریم:

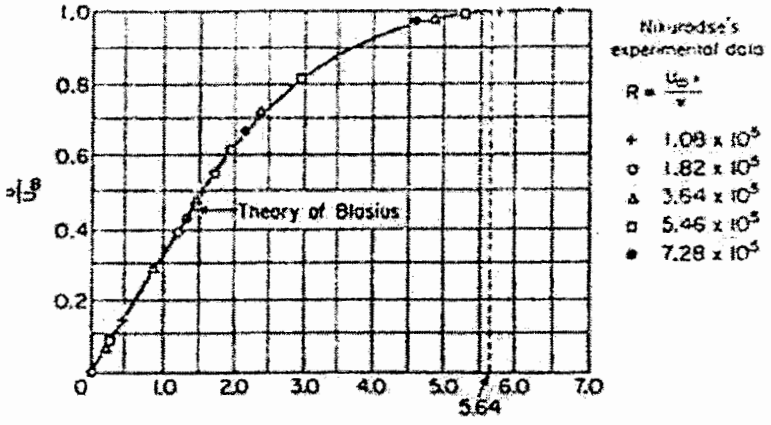
$$\begin{aligned} &\lim_{\eta\to\infty}\left[A_2^{1/3}F(A_2^{1/3}\eta)\right] = f'(\infty) = 1 \\ &or \\ &A_2 = \left[\frac{1}{\lim_{\eta\to\infty}F'(\eta)}\right]^{3/2} \end{aligned} \qquad (ج-۲۵)$$

مقدار A_2 را می‌توان به روش عددی تا هر دقت دلخواه محاسبه کـرد. Howearth مقـدار $A_2 = 33026$ را بدست آورد. با این مقدار برای A_2 نمودارهای توابـع f, f', f'' در نمـودار شـکل (ج-۲) رسم شده است.



شکل ج-۲: نمودار توابع f, f', f''

مقادیر مولفه‌های سرعت را می‌توان از روابط (ج-۷) و(ج-۸) بدست آورد نمـودار $\frac{u}{U_\infty}$ در شـکل (ج-۳) رسم شده است و با نتایج نیکورادزه که نتایج آزمایشگاهی است مقایسه شده است.



شکل ج-۳: نمودار مولفهٔ $\frac{u}{U_\infty}$ سرعت و مقایسهٔ ان با نتایج نیکورادزه [1]

ج-۲-۲- حل عددی معادلهٔ بلازیوس:

همانطور که می دانیم جریان آرام روی یک صفحه تحت از معادله خود مشابهی زیر تبعیت می کند.

$$f''' + \frac{1}{2} f f'' = 0 \qquad (ج-۲۶)$$

$$f(0)=0 \quad , \quad f'(0)=0 \quad , \quad f'(\eta_\infty)=1$$

که این معادله از معادلات لایه مرزی بدست می آید که در معادله بالا $\frac{u}{u_\infty} = f'(\eta)$, $\eta = \frac{y}{\sqrt{(\frac{vx}{u_\infty})}}$, η_∞ یک مقدار ثا بت و به اندازه کافی بزرگ است. تجربه نشان داده است که η_∞از مرتبه ۱۰ می تواند مقدار مناسبی باشد.

ما می خواهیم یک روش عددی برای حل معادله (ج-۲۶) به دست آوریم. تا از این حل برای به دست آوردن $f''(0)$ که بوسیله آن مقدار اصطکاک سطحی محاسبه می شود، استفاده کنیم.

همانطور که می دانیم حل تحلیلی بلازنویس مقدار زیر را برای C_f پیشنهاد می کند.

$$C_f = \frac{0.664}{\sqrt{\mathrm{Re}_x}} \qquad (ج-۲۷)$$

معادله (ج-۲۶) معادله‌ای است که در آن دو نقطه مختلف مرزی دارای مقادیر معلوم هستند، (Two- point boundary value) که می تواند بوسیله روش تفاضل محدود حل شود یا اینکه مسأله تبدیل به مسئله ای با شرایط اوّلیه شده و حل شود. در روش اوّل به علّت خطی سازی مجبور به انجام تکرار هستیم. در روش دوّم یک شرط اوّلیه دیگر در $\eta = 0$ (همراه با دو روش اولیه داده شده) باید حدس زده شود وسپس با انجام تکرار تا زمانی که شرط مرزی در η_∞ ارضا شود ، محاسبات باید ادامه یابد . ما در این قسمت هردو روش را توضیح داده وآنرا حل می کنیم .

در روش اول ما باید معادله (ج-۲۶) را گسسته کرده وخطی کنیم و سپس به صورت ضمنی معادلات خطی جبری حاصل را حل کنیم ، لازم به ذکر است که در حل دستگاه معـادلات خطـی باید شرایط مرزی لحاظ شوند ، سه نوع خطی سازی می تواند در نظر گرفته شود .

۱) خطی سازی خام[1]

حل شناخته شده از f را با علامت $\bar{f}$ نشان مـی دهـیم (کـه در مرحلـه اول و بـرای شـروع مقداری برای آن حدس زده می شود)رابطه (ج-۲۶) به صورت زیر در می آید.

$$f''' + \frac{1}{2}\bar{f}f'' = 0 \qquad \text{(ج-۲۸)}$$

استفاده از این روش بسار ساده است ولی نمی توان آن را بـرای توانهـای متغیـر وابـسته یـا مشتقات آن اعمال کرد .

۲) روش شیب[2]

با معرفی کمیت فرضی f به صورت $\tilde{f} = f - \bar{f}$ و $\tilde{f} = f' - \bar{f}'$ و... کـه علامـت « Over bar» نمایانگر حل معلوم است ، آنگاه ترم غیر خطی $N(f, f', ...)$ با استفاده از فرمول زیر خطـی می شود:

$$N(f, f', ...) \cong \overline{N}(\bar{f}, \bar{f}', ...) + \frac{\overline{\partial N}}{\partial f}(f - \bar{f}') + \frac{\overline{\partial N}}{\partial f'}(f' - \bar{f}') + ... \qquad \text{(ج-۲۹)}$$

با اعمال این روش رابطه (۲۶-۲-ج) به صورت زیر تغییر می کند .

$$f''' + \frac{1}{2}\bar{f}f'' + \frac{1}{2}\bar{f}\bar{f}'' = \frac{1}{2}\overline{ff}'' \qquad \text{(ج-۳۰)}$$

جوابی از f در رابطه (ج-۳۰) مورد نظر است که بهتـر از $\bar{f}$ باشـددر مرحلـه بعـد $\bar{f}$ بـا f جاگذاری می شود . این روش تا زمانی ادامه پیدا می کند که اختلاف $|f - f'|$ در تمام گروه هـا از یک مقدار کوچک ، کوچکتر باشد . یاد آوری کنیم که $|f - f'|$ الزاماً خطـای مطلـق را نـشان

---

[1] Crude linearization
[2] Gradient method

نمی‌دهند ، اما معیاری برای تکرار است .همچنین گسسته سازی f'' منجر به ایجاد یک ماتریس ضرایب چهار قطری می شود که نیازمند اعمال روش گاوس پیشرو است تا تمام عناصر زیر قطر اصلی صفر شود .

۳) روش نیوتن

روش معروف دیگری که معمولا استفاده می شود استفاده از روش چند بعدی نیوتن است . دوباره یک راه حل تقریبی دیگر را با $\bar{f}$ نشان می دهیم . در این قسمت ما یک کمیت جدید اغشاش $\tilde{f}$را به صورت زیر معرفی می کنیم .

$$f = \bar{f} + \tilde{f} \qquad (ج-۳۱)$$

اگر رابطه (ج-۳۱) را در رابطه (ج-۳۲) قرار دهیم واز ترمهای $\tilde{f}^2$ صرف نظر کنیم وترمهای غیر خطی را با استفاده از فرمول (ج-۳۲) خطی سازی کنیم :

$$N(f, f, ...) \cong \overline{N}(\bar{f}, \bar{f}', ...) + \frac{\overline{\partial N}}{\partial f}\tilde{f} + \frac{\overline{\partial N}}{\partial f'}\tilde{f}' + ... \qquad (ج-۳۲)$$

آنگاه ما به یک رابطه خطی شده حاکم به صورت تابعی از ترم اغشاش به صورت زیر می‌رسیم .

$$\tilde{f}''' + \frac{1}{2}\bar{f}\,\tilde{f}'' + \frac{1}{2}\bar{f}''\,\tilde{f} = -\bar{f}''' - \frac{1}{2}\bar{f}\,\bar{f}'' ,$$
$$\tilde{f}(0) = 0 \quad , \quad \tilde{f}'(0) = 0 \quad , \quad \tilde{f}'(\delta_\infty) = 0 \qquad (ج-۳۳)$$

که در رابطه (ج-۳۳) ما یک معادله دیفرانسیل با شرایط مرزی همگن داریم .بعد از گسسته سازی وحل دستگاه معادلات خطی برای گره های مختلف مقدار $\tilde{f}$ محاسبه می شود با جاگذاری $\bar{f}, \tilde{f}$ در رابطه (ج-۳۱) مقدار جدید f محاسبه شده و جایگزین $\bar{f}$ می شود ، یاد آوری میشودچنانچه f به جواب واقعی خیلی نزدیک شود طرف راست معادله (ج-۳۳) صفر شده ودر نتیجه خود $\tilde{f}$ هم صفر خواهد شد ، بنابراین $\tilde{f}$ مقداری خطای مطلق است .

دقت شود که فرمول تقاضل محدود برای مسائل با مقادیر مرزی دونقطه ای وبوسیله یکی از روشهای خطی سازی گفته شده نیازمند مقادیر قابل توجه هزینهٔ محاسباتی وهمچنین تعداد گره قابل قبول برای رسیدن به جواب با دقت مناسب است . بنابراین بیشتر متداول است تا روش دوم ذکر شده را به کار ببریم ومسئله را به مسئله از نوع شرط اولیه تبدیل کنیم .

رابطه (ج-۲۶) به صورت سه معادله دیفرانسیل مرتبه اول که به صورت دوتایی به هم مربوط می‌شوند ، نوشته می شود .

$$\begin{aligned} Z_1 &= Z_2 \quad , \quad Z_1(0) = 0 \\ Z_2 &= Z_3 \quad , \quad Z_2(0) = 0 \\ Z_3 &= -\frac{1}{2} Z_1 Z_3 \quad , \quad Z_3(0) = f''(0) \end{aligned} \qquad (ج-۳۴)$$

که در معادلات بالا $Z_1 = f$ و $Z_2 = f'$ و $Z_3 = f''$ فرض شده است. سه معادله بالا همزمان بوسیله روش Runge – Kutta حل می شود که برنامه آن به زبان فرترن ضمیمه می باشد در تکرار اول ما یک مقدار اولیه نامعلوم را برای $f''(0)$ حدس می زنیم مانند $g = f''(0) = 0.5 (= Z30)$ تجربه نشان می دهد که حل تا $\eta_\infty = 10$ یا EMAX=10 حل لایه مرزی و ضخامت مورد نظر را پوشش می دهد در این نقطه ماشرایط مرزی باقیمانده را چک می کنیم $f'(\delta_\infty) = 1.0 \; (= Z2)$.این کار با دقت دلخواه بوسیله متغیر eps انجام می‌شود . اگر این شرط مرزی ارضا نشود ما نیازمند یک معیار برای متغیر Z30 هستیم تا یک تکرار موفقیت آمیز داشته باشیم .

یک روش ساده استفاده از روش نصف کردن است .فرض می شود که $f'(\delta_\infty)$ یک تابع پیوسته و یکنواخت از $f''(0)$ باشد در این روش $f''(0)$ با مقداراولیه $\delta f''(0) \; (= \text{DZ30})$ افزایش می یابد این افزایش تا زمانی ادامه پیدا می کند که مقدار خطا $ERR \; (= Z2 - 1.0 = e = f'(\delta_\infty) - 1.0$ یک عدد منفی باشد. هنگامی که مقدار خطا مثبت باشد گام پیشروی نصف شده وعلامت آن تغییر می کند، یعنی در حقیقت قبل از شروع پیشروی باید از قطعه کد زیر استفاده کنیم .

```
IF ((DZ30 .GT. 0. .AND. ERR .GT.0.) .OR.
```

```
(DZ30 .LT. 0. AND. ERR .LT.0.)) DZ30=-0.5*DZ30
Z30=Z30+DZ30
```

قبل از شروع روش دیگرلازم است یادآوری کنیم که این روش خوب است ولی سرعت آن کم است

.

یک معیار دیگر برای چک کردن این موضوع استفاده از روش نیوتن است .در این روش فرض بر این است که e یعنی خطا تابع یکنواختی از حدس یعنی g باشد. سری بسط یافتهٔ e(g) به صورت زیر می باشد.

$$e \simeq e_1 + (\frac{\partial e}{\partial g})_1 \Delta g \qquad (ج-۳۵)$$

بنابراین برای اینکه مقدار خطا در مرحله بعدی صفر شود، آنگاه g باید با گام زیر افزایش یابد .

$$\Delta g = -\frac{e_1}{(\partial e / \partial g)_1} \qquad (ج-۳۶)$$

در نتیجه مقدار پیشروی حدس به صورت زیر است .

$$g = g_1 + \Delta g \qquad (ج-۳۷)$$

در برنامه ای که در ادامه این بحث آمده است . ما مقادیر متوالی از حدس اولیه g را با Z301 و Z30 نشان داده ایم و ERRوERR1 مطابق با مقادیر متوالی از حدس اولیه g در شرط مرزی η_∞ است الگوریتم روابط (ج-۳۶) و(ج-۳۷) به صورت زیر در می آید .

```
DZ30=-ERR/(ERR-ERR1)*(Z30-Z301)
Z30=Z30 + DZ30
```

جزئیات به روز درآوردن وذخیره مقادیر متوالی در کد(۱-۲-۵) قابل مشاهده است.

کد ج-۱ : حل عددی معادلهٔ بلازیوس به روش **Shooting**

```
 INTEGER IT,I
REAL*8 IPP,K1,K2,K3,K4,L1,L2,L3,L4,M1,M2,M3,M4
REAL*8 DE,EMAX,Z30,Z301,DZ30,EPS,IP,E,Z1,Z2,Z3

OPEN(1,FILE='BELASIUS.TXT',STATUS='UNKNOWN')
EMAX=12
EPS=0.000001
DE=0.01
Z30=0.5
DZ30=0.2
```

```
      IP=20
      IT=0
      IPP=0
            DO
                  I=0
                  IT=IT+1
                  Z1=0.
                  Z2=0.
                  Z3=Z30
                  E=0.

                  IF (IPP.NE.0.) THEN
                        WRITE(1,10) E,Z1,Z2,Z3
10                      FORMAT(4(2X,F12.6))
                  END IF
                        DO
                         I=I+1
                         K1 = Z2
                         L1 = Z3
                         M1 = -0.5 * Z1 * Z3

                         K2 = Z2 + L1 * DE / 2
                         L2 = Z3 + M1 * DE / 2
                         M2 = -0.5 * (Z1 + K1 * DE / 2) * (Z3 + M1 * DE / 2)

                         K3 = Z2 + L2 * DE / 2
                         L3 = Z3 + M2 * DE / 2
                         M3 = -0.5 * (Z1 + K2 * DE / 2) * (Z3 + M2 * DE / 2)

                         K4 = Z2 + L3 * DE
                         L4 = Z3 + M3 * DE
                         M4 = -0.5 * (Z1 + K3 * DE) * (Z3 + M3 * DE)

                         Z1 = Z1 + (K1 + 2 * K2 + 2 * K3 + K4) / 6 * DE
                         Z2 = Z2 + (L1 + 2 * L2 + 2 * L3 + L4) / 6 * DE
                         Z3 = Z3 + (M1 + 2 * M2 + 2 * M3 + M4) / 6 * DE
                         E = E + DE
                              IF(IPP.NE.0.) THEN
                              IF(INT(I/IP)*IP.EQ.I) THEN
                                          WRITE(1,10) E,Z1,Z2,Z3
                                    END IF
                              END IF
                              IF (E.GT.EMAX) THEN
                                    EXIT
                              END IF
                        END DO
                  ERR0=Z2-1
                  IF (ABS(ERR0).LE.EPS) THEN
                        IF (IPP.LE.0.) THEN
                              IPP=1.
                        ELSE
                              WRITE(1,20)
20                            FORMAT('------------------------------')
                              WRITE(1,21) IT
21                            FORMAT('ITERATION COUNT='I2)
                              WRITE(1,20)
```

```
                    EXIT
                END IF
            ELSE
                IF(IT.GT.1.) THEN
                    DZ30=-ERR0/(ERR0-ERR1)*(Z30-Z301)
                    ERR1=ERR0
                    Z301=Z30
                    Z30=Z30+DZ30
                ELSE
                    ERR1=ERR0
                    Z301=Z30
                    Z30=Z30+DZ30
                END IF
            END IF
            END DO
110     CONTINUE
        CLOSE(1)
        END
```

نتایج حل کد(ج-۱) در جدول (ج-۱) ارائه شده است. همانطور که در ردیف اول که حالت انتخاب شده دارد، مشاهده می‌شود که مقدار ضریب اصطکاک با حل تحلیلی بلازیوس مطابقت دارد.

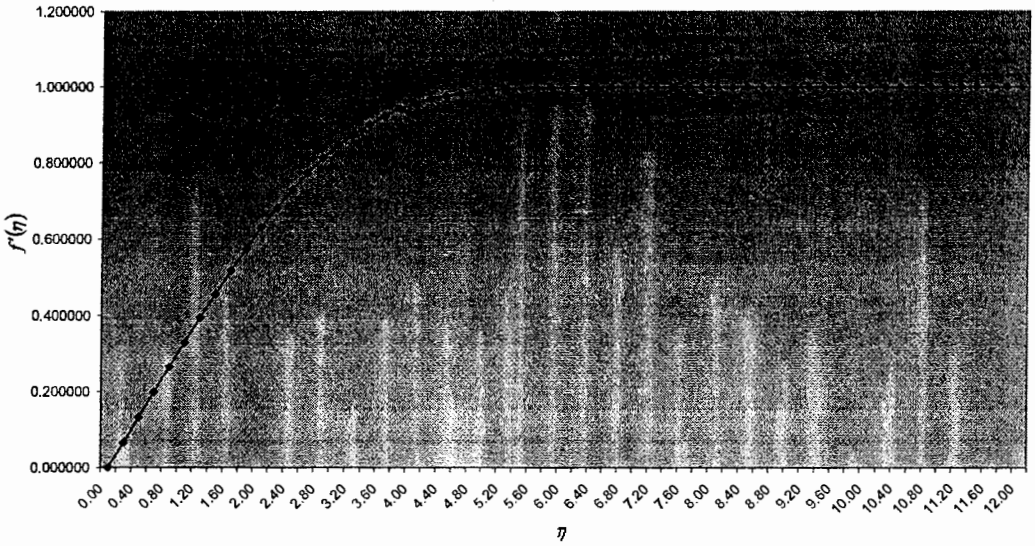
$$\tau_0 = \mu\left(\frac{\partial u}{\partial y}\right)_{y=0} = \frac{\mu U_\infty f''(0)}{\sqrt{\upsilon x/U_\infty}} = \frac{0.332}{\sqrt{\mathrm{Re}_x}}\rho U_\infty^2$$

$$C_f = \frac{\tau_0}{1/2\rho U_\infty^2} = \frac{0.332}{\sqrt{\mathrm{Re}_x}} \qquad (ج-۳۸)$$

جدول ج-۱: نتایج حل عددی بلازیوس به روشShooting

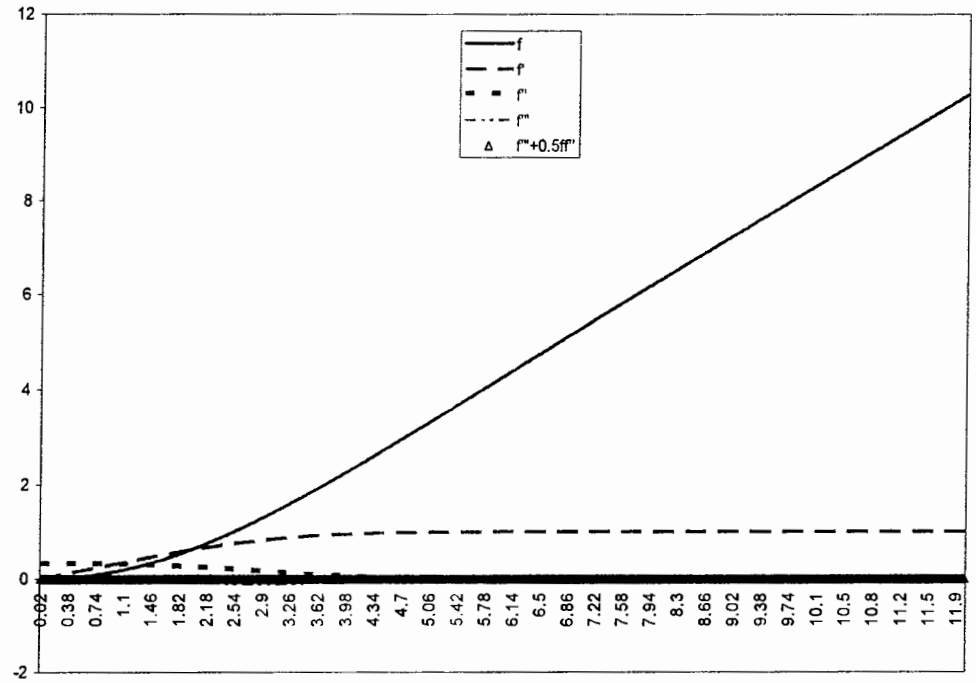
$\eta = y\sqrt{\frac{U_\infty}{\upsilon x}}$	f	$f' = \frac{u}{U_\infty}$	f''
0.00	0.000000	0.000000	0.332057
0.20	0.006641	0.066408	0.331984
0.40	0.026560	0.132764	0.331470
0.60	0.059735	0.198937	0.330079
0.80	0.106108	0.264709	0.327389
1.00	0.165572	0.329780	0.323007
1.20	0.237949	0.393776	0.316589
1.40	0.322982	0.456262	0.307865
1.60	0.420321	0.516757	0.296663
1.80	0.529518	0.574758	0.282931
2.00	0.650024	0.629766	0.266752
2.20	0.781193	0.681310	0.248351
2.40	0.922290	0.728982	0.228092

2.60	1.072506	0.772455	0.206455
2.80	1.230977	0.811509	0.184007
3.00	1.396808	0.846044	0.161360
3.20	1.569095	0.876081	0.139128
3.40	1.746950	0.901761	0.117876
3.60	1.929525	0.923330	0.098086
3.80	2.116029	0.941118	0.080126
4.00	2.305746	0.955518	0.064234
4.20	2.498039	0.966957	0.050520
4.40	2.692360	0.975871	0.038973
4.60	2.888247	0.982683	0.029484
4.80	3.085320	0.987789	0.021871
5.00	3.283273	0.991542	0.015907
5.20	3.481867	0.994245	0.011342
5.40	3.680918	0.996155	0.007928
5.60	3.880290	0.997478	0.005432
5.80	4.079881	0.998375	0.003648
6.00	4.279620	0.998973	0.002402
6.20	4.479457	0.999362	0.001550
6.40	4.679356	0.999612	0.000981
6.60	4.879295	0.999768	0.000608
6.80	5.079259	0.999864	0.000370
7.00	5.279238	0.999921	0.000220
7.20	5.479226	0.999956	0.000129
7.40	5.679219	0.999975	0.000074
7.60	5.879215	0.999987	0.000041
7.80	6.079213	0.999993	0.000023
8.00	6.279212	0.999996	0.000012
8.20	6.479212	0.999998	0.000006
8.40	6.679212	0.999999	0.000003
8.60	6.879211	0.999999	0.000002
8.80	7.079211	1.000000	0.000001
9.00	7.279211	1.000000	0.000000
9.20	7.479211	1.000000	0.000000
9.40	7.679211	1.000000	0.000000
9.60	7.879211	1.000000	0.000000
9.80	8.079211	1.000000	0.000000
10.00	8.279211	1.000000	0.000000
10.20	8.479211	1.000000	0.000000
10.40	8.679211	1.000000	0.000000
10.60	8.879211	1.000000	0.000000
10.80	9.079211	1.000000	0.000000
11.00	9.279211	1.000000	0.000000
11.20	9.479211	1.000000	0.000000
11.40	9.679211	1.000000	0.000000
11.60	9.879211	1.000000	0.000000
11.80	10.079211	1.000000	0.000000
12.00	10.279211	1.000000	0.000000



شکل ج-۴ : نمودار نتایج حل عددی معادلهٔ بلازیوس به روش **Shooting**

شکل ج-۵ نموداری را نشان می‌دهد که در آن f''', f'', f', f و همچنین $f''' + \frac{1}{2} ff''$ در یک نمودار رسم شده است همانطور که از نمودار مشخص است مقدار $f''' + \frac{1}{2} ff''$ با دقت قابل قبولی برابر صفر است.



شکل ج-۵ : حل عددی معادلهٔ بلازیوس ، نمودارهای f''', f'', f', f و $f''' + \frac{1}{2} ff''$

ضمیمهٔ د) کد مستقیم عددی جریان برشی

```
      IMPLICIT NONE
      CALL FREE_SHEAR()
      END
C     ////////////////////////////////////////////////////////////
C     ////////////////////////////////////////////////////////////
C
C     MAIN ENTRY FOR THIS APPLOCATION WHICH CALCULATE THE مستقیم عددی METHOD FOR
C     FREE_SHEAR FLOW
C
C     ////////////////////////////////////////////////////////////
      SUBROUTINE FREE_SHEAR()

      IMPLICIT NONE
C
C     REQUIRED VARIABLES
C

      INTEGER NX,NY,NXP,NYP,STATE,NSTEP,ERROR,ML,MU,LDA,SUB,STEP,
     1PERIOD,START,J,I,NO,NSUB,HISTORY,K
      REAL*8 DT,LX,TIME,T,BETA,ETA,DX,AL,RE,CRATIO,DETA,ACFL,PI,LETA
      real*8,ALLOCATABLE :: U(:,:),V(:,:),UP(:,:),VP(:,:),DX1(:,:),
     1DX2(:,:),DY1(:,:),DY2(:,:),B_FLOW(:,:),A1(:,:),DUP(:,:),
     1R(:,:),B2(:,:),OLD_R(:,:),C1(:,:),C2(:,:),DX2T(:,:),UBX2(:),
     1DY2T(:,:),RT(:,:),DVB1(:),DVB2(:),UBX1(:),UBY1(:),UBY2(:),OLD_V(:)
     1,OLD_U(:),V_T1(:),V_T2(:),XM(:,:),YM(:,:),VORTICITY(:,:),
     1INFLOWDU(:),OUTFLOWDU(:),ARKW(:),BRKW(:),DX1B(:,:),DX2B(:,:)
     1,DY2T_INV(:,:)
C
C     READ INPUT FILE GENERATED FROM VISUAL BASIC
C
      OPEN(1,FILE='INPUT.MHD',STATUS='UNKNOWN')
      READ(1,*) PERIOD,NX,NY,NSTEP,START,STATE,LX,T,ACFL,BETA,RE
     1,CRATIO,HISTORY
      CLOSE(1)
C
C     PRINT DATA ON MONITOR
C
      IF(STATE.EQ.0) THEN
      WRITE(*,*)'          JET FLOW  '
      ELSE IF (STATE.EQ.1) THEN
      WRITE(*,*)'          گردابه FLOW '
      ELSE IF (STATE.EQ.2) THEN
      WRITE(*,*)'          MIXING FLOW    '
      END IF
C
C
      WRITE(*,699)
699   FORMAT('  ------------INLET PARAMETERS---------------')
      WRITE(*,700)RE
700   FORMAT('REYNOLDS NUMBER                    ='F15.2)
```

```
      WRITE(*,701)NSTEP
701   FORMAT('MAXIMUM NO. SIMULATION STEPS          ='I15)
      WRITE(*,702)T
702   FORMAT('T  (NON-DIMENSIONAL TIME)             ='F15.2)
      WRITE(*,704)PERIOD
704   FORMAT('SAVE TO DISK AFTER ... STEPS          ='I15)
      WRITE(*,705)NX
705   FORMAT('NX                          ='I15)
      WRITE(*,706)NY
706   FORMAT('NY                          ='I15)
      WRITE(*,707)BETA
707   FORMAT('BETA (Y-COORD STRETCHING PARA.)       ='F15.2)
      WRITE(*,708)LX
708   FORMAT('LX  (NON-DIMENSIONAL X-DOMAIN.)       ='F15.5)
      WRITE(*,800)CRATIO
800   FORMAT('OUTFLOW CONVECTION VELOCITY           ='F15.4)
      WRITE(*,801)ACFL
801   FORMAT('CFL NUMBER                  ='F15.5)
      WRITE(*,802)
802   FORMAT('  ----------------------------------------------')
      WRITE(*,803)
803   FORMAT('')
C
C     SET PARAMETERS
C
      NSUB= 3
      ML=1
      MU=ML
      LDA=2*ML+MU+1
      DX=LX/(NX-1)
      DT=T/NSTEP
      TIME=(START-1)*DT
      NXP=NX
      NYP=NY
      PI=4.*DATAN(1.0_8)
      DETA=1./(NY-1)
C
C     GET DYNAMIC ARRAY (POINTER)
C
      ALLOCATE (U(NYP,NXP),V(NYP,NXP),UP(NYP,NXP),B_FLOW(NYP,NXP),
     1VP(NYP,NXP),DX1(NXP,NXP),DX2(NXP,NXP),DY1(NYP,NYP),DY2(NYP,NYP),
     1A1(NXP,NXP),DUP(NYP,NXP),ARKW(3),BRKW(3),R(NYP,NXP),C1(NXP,NXP),
     1C2(NXP,NXP),B2(NXP,NXP),DY2T(NYP,NYP),DX2T(NXP,NXP),OLD_R(NYP,NXP)
     1,RT(NYP,NXP),DVB1(NYP),DVB2(NYP),UBX1(NYP),UBX2(NYP),UBY1(NXP)
     1,UBY2(NXP),OLD_U(NYP),OLD_V(NYP),V_T1(NYP),V_T2(NYP),INFLOWDU(NYP)
     1,OUTFLOWDU(NYP),DX1B(NXP,NXP),DX2B(NXP,NXP),DY2T_INV(NXP,NXP),
     1XM(NYP,NXP),YM(NYP,NXP),VORTICITY(NYP,NXP),STAT = ERROR)
      IF (ERROR.NE.0) STOP 'ERROR ALLOCATE AT MAIN'
C
C     SET ARRAYES ALEMENTS EQALL ZERO
C
      U=0._8;V=0._8;UP=0._8;VP=0._8;DX1=0._8;DX2=0._8;DY1=0._8;DY2=0._8;
      B_FLOW=0._8;A1=0._8;DUP=0._8;R=0._8;B2=0._8;OLD_R=0._8;C1=0._8;
      C2=0._8;DX2T=0._8;UBX2=0._8;DY2T=0._8;RT=0._8;DVB1=0._8;DVB2=0._8;
      UBX1=0._8;UBY1=0._8;UBY2=0._8;OLD_V=0._8;OLD_U=0._8;V_T1=0._8;
      XM=0._8;YM=0._8;VORTICITY=0._8;INFLOWDU=0._8;OUTFLOWDU=0._8;
```

```
      DX1B=0._8;DX2B=0._8;V_T2=0._8;
C
C     TIME ADVANCMENT PARAMETERS
C
      ARKW(1) = 8./15.
      ARKW(2) = 5./12.
      ARKW(3) = .75
      BRKW(1) = 0.
      BRKW(2) = -17./60.
      BRKW(3) = -5./12.
C
C     CALCULATE DRIVATE OPERATION
C
      CALL DRIVED_Y(4,DY1,DY2,NY,NYP,BETA)
      CALL DRIVED_X(4,DX1,DX2,NX,NXP,LX)
      CALL DRIVED_X(2,DX1B,DX2B,NX,NXP,LX)
      AL=1./4.
      CALL PADEBUELL2(2,AL,DX,NX,NXP,A1,B2)
      CALL INVERSE(A1,C2,NX,NXP,ML,MU,LDA)
      CALL TRANSPOSE(C2,NX,NXP)
      CALL TRANSPOSE(B2,NX,NXP)
      CALL TRANSPOSE(DX1,NX,NXP)
      CALL TRANSPOSE(DX2,NX,NXP)
      CALL TRANSPOSE(DX1B,NX,NXP)
      CALL TRANSPOSE(DX2B,NX,NXP)

C     SET VALUES OF DY2T AND DX2T WHICH NOT CONTAIN FIRST AND LAST ROW AF DY2 AND DX2

      DO 11 I=2,NY-I
            DO 12 J=2,NY-1
                  DY2T(I-1,J-1)=DY2(I,J)
12          CONTINUE
11    CONTINUE
      ML=NY-2
      MU=ML
      LDA=2*ML+MU+1
      CALL INVERSE(DY2T,DY2T_INV,NY-2,NYP,ML,MU,LDA)
      DO 30 I=2,NX-1
            DO 40 J=2,NX-1
                  DX2T(I-1,J-1)=DX2(I,J)
40          CONTINUE
30    CONTINUE
C
C
C     INITIAL CONDITION (I.C.)
C
      IF (HISTORY.NE.1) THEN
      CALL INITIALVALUES(STATE,NY,NX,NYP,B_FLOW,LX,BETA)
      PRINT *,"SOLVING, PLEASE WAIT..."
      DO 100 I=1,NY
            DO 101 J=1,NX
                  U(I,J)=B_FLOW(I,J)
                  V(I,J)=0._8
101         CONTINUE
100   CONTINUE
```

```
      ELSE
            PRINT *,"SOLVING FROM HISTORY PLEASE WAIT..."
            OPEN(1,FILE='U_OUT.TXT',STATUS='UNKNOWN')
            OPEN(2,FILE='V_OUT.TXT',STATUS='UNKNOWN')
            DO 104 I=1,NY
                  READ (1,*) (U(I,J),J=1,NX)
                  READ (2,*) (V(I,J),J=1,NX)
104         CONTINUE
            CLOSE(1)
            CLOSE(2)
      END IF


C
C
C     BEGIN SOLUTION
C
C
      K=NX/5
      OPEN(100,FILE='Transit_u.DAT',STATUS='UNKNOWN')
      WRITE(100,*) 'VARIABLES = "Time"'
      OPEN(101,FILE='Transit_v.DAT',STATUS='UNKNOWN')
      WRITE(101,*) 'VARIABLES = "Time"'
      OPEN(102,FILE='Transit_W.DAT',STATUS='UNKNOWN')
      WRITE(102,*) 'VARIABLES = "Time"'
333   FORMAT(',"X=',I0,'"')
      DO I=K,NX,K
      WRITE (100,333) I
      WRITE (101,333) I
      WRITE (102,333) I
      END DO
      DO 20 STEP=START,NSTEP
            DO 10 SUB=1,NSUB
                  CALL ZARB(DY1,V,VP,NY,NY,NX,NYP,NYP)
                  INFLOWDU(1:NY)=-VP(1:NY,1)
                  OUTFLOWDU(1:NY)=-VP(1:NY,NX)
                  DO 231 I=1,NY
                        V_T1(I)=V(I,1)
                        V_T2(I)=V(I,NX)
231               CONTINUE
                  CALL V_FROM_UILL(U,V,INFLOWDU,OUTFLOWDU,NX,NY
     1            ,NXP,NYP,DX1,DY1,DY2T_INV)
                  V(1:NY,1)=V_T1(1:NY)
                  V(1:NY,NX)=V_T2(1:NY)
                  DO 36 I=1,NY
                        DO 37 J=1,NX
                              RT(I,J)=0._8
37                      CONTINUE
                        UBX2(I)=0._8
                        UBX1(I)=0._8
                        DVB2(I)=0._8
36                CONTINUE
                  CALL RHS(U,V,INFLOWDU,OUTFLOWDU,R,DX1,DX2,DY1,DY2
     1            ,C2,B2,RE,NX,NY,NXP,NYP,DX,STEP,SUB,DT)
                  CALL TIMEADVANCMENT(RT,R,OLD_R,DT,ARKW(SUB),BRKW(SUB),
     1      NX,NY,NYP)
```

```
                        CALL SET_BC_CONDITION(V,U,ARKW(SUB),BRKW(SUB),DX1,DY1
     1                  ,OLD_V,OLD_U,DT,DVB1,DVB2,UBX1,UBX2,UBY1,UBY2,NY,NX,NYP,NXP,
   1              CRATIO)
                        CALL POISSONSOLVER(RT,DUP,DY2,DX2,DY2T,DX2T,UBX1,UBX2
     1                  ,UBY1,UBY2,NY,NX,NYP,NXP)
                        DO 21 I=1,NY
                                DO 22 J=1,NX
                                        U(I,J)=U(I,J)+DUP(I,J)
22                              CONTINUE
21                      CONTINUE
                        DO 23 I=1,NY
                                V(I,1)=V(I,1)+DVB1(I)
                                V(I,NX)=V(I,NX)+DVB2(I)
23                      CONTINUE
                TIME=TIME+DT*(ARKW(SUB)+BRKW(SUB))
10              CONTINUE
                NO=STEP/PERIOD
                IF (NO*PERIOD.EQ.STEP) THEN
                CALL SAVE_HISTORY(U,V,DUP,NX,NY,NSTEP,LX,T,ACFL,BETA,RE
     1          ,STEP+1,STATE,NO,PERIOD,CRATIO)
                END IF
20      CONTINUE
        CLOSE(100)
        CLOSE(101)
        CLOSE(102)
C
C       END OF SOLUTION
C


        OPEN(1,FILE='MAIN_TEST.TXT',STATUS='UNKNOWN')
        DO 102 I=1,NY
                        WRITE(1,999)(V(I,J),J=1,NX)
999                     FORMAT(500(2X,F20.6))
102     CONTINUE
        CLOSE(1)
        PRINT *,"SOLUTION COMPLATE."
        PRINT *,"Converting data to Tecplot format..."
        CALL ZEROS(UP,NYP,NXP,NYP)
        CALL ZEROS(VP,NYP,NXP,NYP)
        CALL ZARB(V,DX1,VP,NY,NX,NX,NYP,NXP)
        CALL ZARB(DY1,U,UP,NY,NY,NX,NYP,NYP)
        CALL MINUS(VP,UP,VORTICITY,NY,NX,NYP)
        DO 81 I=1,NY
                ETA=(I-1)*DETA
                DO 81 J=1,NX
                        VORTICITY(I,J)=0.5*VORTICITY(I,J)
                        XM(I,J)=DX*(J-1)
81                      YM(I,J)=-BETA/DTAN(PI*ETA)
        OPEN(1,FILE='UVMESH.DAT',STATUS='UNKNOWN')
        WRITE (1,*) 'VARIABLES = "X", "Y", "U","V","VORTICITY"'
        WRITE (1,*) 'ZONE I=', NX, ', J=', NY-2, ', DATAPACKING=POINT'
        DO 1 I=2,NY-1
                DO 1 J=1, NX
1                       WRITE (1,*) XM(I,J),YM(I,J),U(I,J),V(I,J),VORTICITY(I,J)
        CLOSE(1)
```

```
      CALL SELF_SELMILARITY(STATE,XM,YM,U,V,NY,NX,NYP,RE,VORTICITY,UP)
      PAUSE 'Press any key to continue...'
      RETURN
      END SUBROUTINE
C     ////////////////////////////////////////////////////////
      SUBROUTINE SELF_SELMILARITY(STATE,XM,YM,U,V,NY,NX,NYP,RE,
     1      VORTICITY,UP)
            INTEGER I,J,K,NX,NY,NYP,M,STATE
            REAL*8 XM(NYP,1),YM(NYP,1),U(NYP,1),V(NYP,1),RE,UP(NYP,1)
            REAL*8 X1,X2,Y1,Y2,مستقیم عددی_DELTA,UNORM,VORTICITY(NYP,1)
            OPEN(1,FILE='SELF_SELMILARITY.DAT',STATUS='UNKNOWN')
            OPEN(2,FILE='DELTA.DAT',STATUS='UNKNOWN')
            OPEN(3,FILE='U.DAT',STATUS='UNKNOWN')
            WRITE (1,*) 'VARIABLES = "Y", "U/U0" ,"WB/U0"'
            WRITE (2,*) 'VARIABLES = "X","مستقیم عددی DELTA"'
            M=2
C           IF JET FLOW
            IF (STATE.EQ.0) THEN
            NIM=NY/2+1
            DO 1 K=1,NX
                  UNORM=U(NIM,K)
                  WRITE (3,*) XM(NIM,K),U(NIM,K)
                  DO 3 I=2,NY-1
                  IF (U(I,K)>=0.5*UNORM) THEN
                        Y1=YM(I,K)
                        Y2=YM(I-1,K)
                        X1=U(I,K)
                        X2=U(I-1,K)
                        مستقیم عددی_DELTA=DABS((Y1+(Y2-Y1)/(X2-X1)*(0.5*UNORM-X1)))
                        WRITE (2,*) XM(I,K),مستقیم عددی_DELTA
                        EXIT
                  END IF
3                 CONTINUE
                  DO 2 J=M,NY-M
                        WRITE (1,*) YM(J,K)/مستقیم عددی_DELTA,U(J,K)/UNORM,
     1                  VORTICITY(J,K)*مستقیم عددی_DELTA/UNORM
2                 CONTINUE
1           CONTINUE
C           IF گردابه FLOW
            ELSE IF(STATE.EQ.1) THEN
            NIM=NY/2+1
            DO K=1,NX

                  UNORM=(1.-U(NIM,K))
                  WRITE (3,*) XM(NIM,K),(1-U(NIM,K))

                  DO I=2,NY-1
                  IF ((1.-U(I,K))>=0.5*UNORM) THEN
                        Y1=YM(I,K)
                        Y2=YM(I-1,K)
                        X1=(1.-U(I,K))
                        X2=(1.-U(I-1,K))
                        مستقیم عددی_DELTA=DABS((Y1+(Y2-Y1)/(X2-X1)*(0.5*UNORM-X1)))
```

```
                        WRITE (2,*) XM(I,K),مستقیم عددی_DELTA
                        EXIT
                    END IF
                    END DO
                    DO J=M,NY-M
                        WRITE (1,*) YM(J,K)/مستقیم عددی_DELTA,(1.-U(J,K))/UNORM,
     1                  VORTICITY(J,K)*مستقیم عددی_DELTA/UNORM
                    END DO
                END DO
C               IF MIXING FLOW
                ELSE
                DO  K=1,NX
                    UNORM=1.0
                    مستقیم عددی_DELTA=UNORM/MAXVAL(DABS(UP(:,K)))
                    WRITE (2,*) XM(1,K),مستقیم عددی_DELTA
                    DO J=M,NY-M
                        WRITE (1,*) YM(J,K)/مستقیم عددی_DELTA,U(J,K)/UNORM,
     1                  VORTICITY(J,K)*مستقیم عددی_DELTA/UNORM
                    END DO
                END DO
                END IF
                CLOSE(1)
                CLOSE(2)
                CLOSE(3)
                RETURN
            END SUBROUTINE
C           ////////////////////////////////////////////////////////////
            SUBROUTINE SAVE_HISTORY(U,V,DUP,NX,NY,NSTEP,LX,T,CFL,BETA,RE
            1,START,STATE,NO,PERIOD,CRATIO)
            INTEGER NX,NY,NSTEP,START,STATE,NO,PERIOD,BW
            REAL*8 LX,LY,CFL,BETA,T,UINF,RE,U(NY,1),V(NY,1),CRATIO,DUP(NY,1)
            BW=MOD(NO,2)
            IF (BW.EQ.1) THEN
                OPEN(1,FILE='SAVE_HISTORY0.DIB',STATUS='UNKNOWN')
            ELSE
                OPEN(1,FILE='SAVE_HISTORY1.DIB',STATUS='UNKNOWN')
            END IF
            WRITE(1,1000) PERIOD,NX,NY,NSTEP,START,STATE,LX,T,CFL,BETA,RE
            1,CRATIO,4
1000        FORMAT(6(I6,/),6(F15.8,/),I4/)
            DO 1 I=1,NY
                WRITE(1,1001) (U(I,J),J=1,NX)
1001            FORMAT(1000(2X,F15.8))
1           CONTINUE
            DO 2 I=1,NY
                WRITE(1,1001) (V(I,J),J=1,NX)
2           CONTINUE
            WRITE(1,*)
            WRITE(1,*)
            WRITE(1,*) '===> UP'
            WRITE(1,*)
            DO 3 I=1,NY
                WRITE(1,1007) I,(DUP(I,J),J=1,NX)
3           CONTINUE
```

```
1007            FORMAT(I6,1000(2X,F15.6))
        CLOSE(1)
        BW=MOD(NO,2)
        IF (BW.EQ.1) THEN
                WRITE (*,1002) START-1
1002            FORMAT('DATA AT STEP='I6' WRITE ON "SAVE_HISTORY0.DIB"')
        ELSE
                WRITE (*,1003) START-1
1003            FORMAT('DATA AT STEP='I6 ' WRITE ON "SAVE_HISTORY1.DIB"')
        END IF
        RETURN
        END SUBROUTINE
C
C       ////////////////////////////////////////////////////////
        SUBROUTINE SET_BC_CONDITION(V,U,C,D,DX1,DY1,OLD_V,OLD_U,DT,
        1DVB1,DVB2,UBX1,UBX2,UBY1,UBY2,NY,NX,NYP,NXP,CRATIO)
        IMPLICIT NONE
        INTEGER NX,NY,NXP,NYP,ERROR,I,J
        REAL*8 V(NYP,1),DX1(NXP,1),DY1(NYP,1),OLD_V(1),OLD_U(1),
        1C,D,DVB1(1),DVB2(1),UBX1(1),UBX2(1),UBY1(1),UBY2(1),DT
        REAL*8 VBP[ALLOCATABLE](:),S1[ALLOCATABLE](:,:),CRATIO,
        1T11[ALLOCATABLE](:,:),T12[ALLOCATABLE](:),U(NYP,1)
C
        ALLOCATE (S1(NYP,NXP),VBP(NYP),T11(NYP,NXP),T12(NYP),STAT = ERROR)
        IF (ERROR.NE.0) STOP 'ERROR ALLOCATE SET_BC_CONDITION'
C


        DO 10 I=1,NY
                DVB1(I)=0.
                UBX1(I)=0.
C               T11(I)=V(I,NX)
10      CONTINUE
C
        DO 20 J=1,NX
                UBY1(J)=0.
                UBY2(J)=0.
20      CONTINUE
C
c       CALL ZARB(DY1,T11,VBP,NY,NY,1,NYP,NYP)
        CALL ZARB(U,DX1,T11,NY,NX,NX,NYP,NXP)
        CALL ZARB(V,DX1,S1,NY,NX,NX,NYP,NXP)
        DO 30 I=1,NY
                VBP(I)=-CRATIO*T11(I,NX)
                T12(I)=-CRATIO*S1(I,NX)
30      CONTINUE
        CALL TIMEADVANCMENT(UBX2,VBP,OLD_U,DT,C,D,1,NY,NYP)
        CALL TIMEADVANCMENT(DVB2,T12,OLD_V,DT,C,D,1,NY,NYP)
C
        DEALLOCATE (S1,VBP,T11,T12,STAT = ERROR)
        IF (ERROR.NE.0) STOP 'ERROR DEALLOCATE SET_BC_CONDITION'
        RETURN
        END SUBROUTINE
C       ////////////////////////////////////////////////////////////
C       ////////////////////////////////////////////////////////////
        SUBROUTINE INITIALVALUES(STATE,NY,NX,NYP,BASEFLOW,LX,BETA)
```

```
C
C       STATE DETERMINE THAT UP AND VP, HOW SHOULD GENERATE
C       ==>STATE=0 JET FLOW
C       ==>STATE=1 گردابه FLOW
C       ==>STATE>=2 MIXING FLOW
C
C       REQUIRED VARIABLES
C
        INTEGER NY,NX,NYP,STATE,I,J
        REAL*8 BASEFLOW(NYP,1),LX,BETA,PI,DETA,Y
        PI=4.*ATAN(1._8)
        DETA=1.0/(NY-1)

C       STATE=0  ===> JET FLOW
        IF (STATE.EQ.0) THEN
                DO 10 I=1,NY
                        DO 20 J=1,NX
                        ETA=(I-1)*DETA
                        Y=-BETA/DTAN(PI*ETA)
                        BASEFLOW(I,J)=1.-DTANH(Y**2.)
20                      CONTINUE
10              CONTINUE

C       STATE=1  ===> گردابه FLOW


        ELSE IF(STATE.EQ.1) THEN
                DO 130 I=1,NY
                        DO 140 J=1,NX
                                ETA=(I-1)*DETA
                                Y=-BETA/DTAN(PI*ETA)
                                BASEFLOW(I,J)=1.-0.692*DEXP(-DLOG(2._8)*Y**2.)
140                     CONTINUE
130             CONTINUE

C       STATE=1  ===> MIXING FLOW

        ELSE IF(STATE.EQ.2) THEN
                        DO 230 I=1,NY
                        DO 240 J=1,NX
                                ETA=(I-1)*DETA
                                Y=-BETA/DTAN(PI*ETA)
                                BASEFLOW(I,J)=0.5*(3+DTANH(Y))
240                     CONTINUE
230             CONTINUE
        END IF

        RETURN
        END SUBROUTINE
C       ////////////////////////////////////////////////////////
C
        SUBROUTINE POISSONSOLVER_TEST()
        INTEGER NY,NX,NYP,NXP,I,J,ERROR
        REAL*8 DY2T[ALLOCATABLE](:,:),DX2T[ALLOCATABLE](:,:),LX,BETA,X,Y
```

```
      REAL*8 DY2[ALLOCATABLE](:,:),DX2[ALLOCATABLE](:,:),ETA,DETA,DX
      REAL*8 DY1[ALLOCATABLE](:,:),DX1[ALLOCATABLE](:,:)
      REAL*8 UBX1[ALLOCATABLE](:),UBX2[ALLOCATABLE](:),PI,LETA,
      1UBY1[ALLOCATABLE](:),UBY2[ALLOCATABLE](:),R[ALLOCATABLE](:,:),
      1DU[ALLOCATABLE](:,:),F[ALLOCATABLE](:,:),FS[ALLOCATABLE](:,:)
      BETA=6.
      LX=2.
      NY=200
      NX=200
      NYP=NY
      NXP=NX
      ALLOCATE (DY2T(NYP,NYP),DX2T(NXP,NXP),R(NYP,NXP),DY2(NYP,NYP),
      1DX2(NXP,NXP),DY1(NYP,NYP),DX1(NXP,NXP),UBX1(NYP),UBX2(NYP),
   1UBY1(NXP),UBY2(NXP),F(NYP,NXP),FS(NYP,NXP),STAT = ERROR)
      DX=LX/(NX-1)
      PI=4.*ATAN(1.0)
      DETA=1.0/(NY-1)
      CALL DRIVED_Y(4,DY1,DY2,NY,NYP,BETA)
      CALL DRIVED_X(4,DX1,DX2,NX,NXP,LX)
      CALL TRANSPOSE(DX2,NX,NXP)
      DO 11 I=2,NY-1
            DO 12 J=2,NY-1
                  DY2T(I-1,J-1)=DY2(I,J)
12          CONTINUE
11    CONTINUE
      DO 30 I=2,NX-1
            DO 40 J=2,NX-1
                  DX2T(I-1,J-1)=DX2(I,J)
40          CONTINUE
30    CONTINUE
      DO 10 I=1,NY
            ETA=(I-1)*DETA
            Y=BETA/DTAN(PI*ETA)
            DO 20 J=1,NX
                  X=(J-1)*DX
                  R(I,J)=DSIN(X)*(2.-Y**2)
                  F(I,J)=Y**2*DSIN(X)+5
20          CONTINUE
            UBX1(I)=Y**2*DSIN(.0_8)+5
            UBX2(I)=Y**2*DSIN(LX)+5
10    CONTINUE
      DO 110 J=1,NX
            X=(J-1)*DX
            UBY1(J)=0.
            UBY2(J)=0.
110   CONTINUE
      CALL POISSONSOLVER(R,FS,DY2,DX2,DY2T,DX2T,UBX1,UBX2
      1,UBY1,UBY2,NY,NX,NYP,NXP)
   OPEN(1,FILE='POISSONSOLVER_TEST.TXT',STATUS='UNKNOWN')
      DO 1 I=1,NY
            WRITE(1,100) ((FS(I,J)-F(I,J)),J=1,NX)
100         FORMAT(1000(2X,F15.6))
1     CONTINUE
      CLOSE(1)
      END SUBROUTINE
C
```

```
C     ////////////////////////////////////////////////////
C     THIS SUBROUTINE SOLVE POISSON DISCREATED EQATION
C     R  =>RIGHT HAND SIDE OF POISSON
C     DU  =>ON RETURN CONTAIN SOLUTION OF POISSON
C     DY2 =>SECOND ORDER OF تفاضلات محدود فشرده MATRIX AT Y DIRECTION ( نگاشت)
C     DX2 =>SECOND ORDER OF تفاضلات محدود فشرده MATRIX AT Y DIRECTION ( نگاشت)
C
      SUBROUTINE POISSONSOLVER(R,DU,DY2,DX2,DY2T,DX2T,UBX1,UBX2
      1,UBY1,UBY2,NY,NX,NYP,NXP)

C     DEFINE REQUIRED VARIABLE

      INTEGER NY,NX,NYP,NXP,I,J,ERROR,FAIL
      REAL*8 R(NYP,1),DU(NYP,1),DY2(NYP,1),DX2(NXP,1),UBX1(1),UBX2(1)
      1,UBY1(1),UBY2(1),DY2T(NYP,1),DX2T(NXP,1)


C     ALLOCATE REQUIRED ARRAYS.
      REAL*8 C[ALLOCATABLE](:,:)
      ALLOCATE (C(NYP,NXP),STAT = ERROR)
      IF (ERROR.NE.0) STOP

C     SET BOUNDARY CONDITION AT FIRST AND LAST STATION.
      DO 31 I=1,NY
            DU(I,1)=UBX1(I)
            DU(I,NX)=UBX2(I)
31    CONTINUE
      DO 32 J=1,NX
            DU(1,J)=UBY1(J)
            DU(NY,J)=UBY2(J)
32    CONTINUE
C     SET RIGHT HAND SIDE OF POISSON EQUATION AFTER BOUNDARY CONDITION SETTING
      DO 50 I=2,NY-1
            DO 60 J=2,NX-1
                  C(I-1,J-1)=R(I,J)-DU(I,1)*DX2(1,J)-DU(I,NX)*DX2(NX,J)-
      1           DY2(I,1)*DU(1,J)-DY2(I,NY)*DU(NY,J)
60          CONTINUE
50    CONTINUE
C     USE POISSON SUBROUTINE TO SOLVE AX+XB=C MATRIX EQATION.
      CALL POISSON(DY2T,DX2T,C,NY-2,NX-2,NYP,NXP,FAIL)
      DO 30 I=1,NY-2
            DO 40 J=1,NX-2
                  DU(I+1,J+1)=C(I,J)
40          CONTINUE
30    CONTINUE
C     DEALLOCATE REQUIRED ARRAYS.
      DEALLOCATE (C,STAT = ERROR)
      IF (ERROR.NE.0) STOP 'ERROR DEALLOCATE POISSONSOLVER'
      RETURN
      END SUBROUTINE

C     ////////////////////////////////////////////////////
C     ********************************************************************
```

```
C     ****************************************************************
C     *****************(TIME-ADVANCEMENT SUBROUTINES)******************
C     ****************************************************************
C     ****************************************************************
C
C     TEST PART OF THE TIME-ADVANCEMENT SCHEME
C
C     ////////////////////////////////////////////////////////////////
      SUBROUTINE TIME_SUB_TEST()

      INTEGER SUB,STEP,ERROR,NX1,NY1
      REAL*8 TIME,DT
      REAL*8 U[ALLOCATABLE](:,:),DUDT[ALLOCATABLE](:,:),
     1OLD_DUDT[ALLOCATABLE](:,:),RHS[ALLOCATABLE](:,:)
     REAL*8 OLD_RHS[ALLOCATABLE](:,:),ARKW[ALLOCATABLE](:),
     1BRKW[ALLOCATABLE](:)
      NX=1
      NY=1
      NX1=NX
      NY1=NY
      ALLOCATE (U(NY1,NX1),DUDT(NY1,NX1),OLD_DUDT(NY1,NX1),RHS(NY1,NX1),
     1OLD_RHS(NY1,NX1),ARKW(3),BRKW(3),STAT = ERROR)
      IF (ERROR.NE.0) STOP

      TIME=1.
      ARKW(1) = 8./15.
      ARKW(2) = 5./12.
      ARKW(3) = .75
      BRKW(1) = 0.
      BRKW(2) = -17./60.
      BRKW(3) = -5./12.
      NSUB= 3
      NSTEP=1000

C     SET INITIAL CONDITIONS

      U(1,1)=-EXP(-TIME)
      DUDT(1,1)=EXP(-TIME)
      DT=1./NSTEP

      DO 20 STEP=1,NSTEP
            DO 10 SUB=1,NSUB
            DUDT(1,1)=EXP(-TIME)
            CALL TIMEADVANCMENT(U,DUDT,OLD_DUDT,DT,ARKW(SUB),BRKW(SUB),
     1     NX,NY,NY1)
            TIME=TIME+DT*(ARKW(SUB)+BRKW(SUB))
10    CONTINUE

      WRITE(*,30) TIME,U(1,1),-EXP(-TIME),
     1ABS((-EXP(-TIME)-U(1,1))/(-EXP(-TIME)))
30    FORMAT(F18.7,F18.7,F18.7,F18.7)
20    CONTINUE
      DEALLOCATE (U,DUDT,OLD_DUDT,RHS,OLD_RHS,ARKW,BRKW,STAT = ERROR)
      IF (ERROR.NE.0) STOP
```

```
      END SUBROUTINE
C     ////////////////////////////////////////////////////////
C
C     ////////////////////////////////////////////////////////
      SUBROUTINE TIME_SUB_TEST2()

      INTEGER SUB,STEP,ERROR,NX1,NY1,I,J,K,L
      REAL*8 TIME,DT,LX,LY,DX,DY,X,Y
      REAL*8 U[ALLOCATABLE](:,:),DUDT[ALLOCATABLE](:,:),
     1OLD_DUDT[ALLOCATABLE](:,:),RHS[ALLOCATABLE](:,:)
     REAL*8 OLD_RHS[ALLOCATABLE](:,:),ARKW[ALLOCATABLE](:),
     1BRKW[ALLOCATABLE](:)
      PRINT *,"PLAESE ENTER A NY"
      READ *,NY
      PRINT *,"PLAESE ENTER A NX"
      READ *,NX
      NX1=NX
      NY1=NY
      ALLOCATE (U(NY1,NX1),DUDT(NY1,NX1),OLD_DUDT(NY1,NX1),RHS(NY1,NX1),
     1OLD_RHS(NY1,NX1),ARKW(3),BRKW(3),STAT = ERROR)
      IF (ERROR.NE.0) STOP
      LX=2
      LY=3
      TIME=1
      ARKW(1) = 8./15.
      ARKW(2) = 5./12.
      ARKW(3) = .75
      BRKW(1) = 0.
      BRKW(2) = -17./60.
      BRKW(3) = -5./12.
      NSUB= 3
      NSTEP=1000
      DX=LX/(NX-1)
      DY=LY/(NY-1)
C     SET INITIAL CONDITIONS
      DO 11 K=1,NY
            Y=(K-1)*DY
            DO 12 L=1,NX
                  X=(L-1)*DX
                  U(K,L)=X*Y*DSIN(TIME)*TIME
12          CONTINUE
11    CONTINUE
      DT=1./NSTEP
      OPEN(1,FILE='TIME_SUB_TEST2.TXT',STATUS='UNKNOWN')
      DO 20 STEP=1,10
            DO 10 SUB=1,NSUB
                  DO 13 K=1,NY
                        Y=(K-1)*DY
                        DO 14 L=1,NX
                              X=(L-1)*DX
                              DUDT(K,L)=X*Y*DCOS(TIME)*TIME+X*Y*DSIN(TIME)
14                      CONTINUE
13                CONTINUE
            CALL TIMEADVANCMENT(U,DUDT,OLD_DUDT,DT,ARKW(SUB),BRKW(SUB),
     1      NX,NY,NY1)
```

```
                TIME=TIME+DT*(ARKW(SUB)+BRKW(SUB))
10              CONTINUE
                DO 21 K=1,NY
                        Y=(K-1)*DY
                        DO 22 L=1,NX
                                X=(L-1)*DX
                                RHS(K,L)=X*Y*DSIN(TIME)*TIME
22                      CONTINUE
21              CONTINUE
                WRITE(1,1005) TIME
1005            FORMAT('----------'F10.6'----------')
                DO 17 K=1,NY
                        WRITE(1,30) (ABS(RHS(K,L)-U(K,L)),L=1,NX)
30                      FORMAT(1000(2X,F13.8))
17              CONTINUE
20      CONTINUE

        DEALLOCATE (U,DUDT,OLD_DUDT,RHS,OLD_RHS,ARKW,BRKW,STAT = ERROR)
        IF (ERROR.NE.0) STOP
        CLOSE(1)
        RETURN
        END SUBROUTINE
C       ////////////////////////////////////////////////////////////
C
C       ////////////////////////////////////////////////////////////

        SUBROUTINE TIMEADVANCMENT(UK,FUK,FUKMIN,DELT,C,D,NX,NY,NYP)

C   THIS ROUTINE PERFORM THE TIME ADVANCEMENT OF THE SIMULATION.
C   IT APPLIES TO A  MODEL EQUATION OF DU/DT = F(U)
C   EXPLICIT THIRD ORDER RUNGE KUTTA SCHEME OF WRAY

        INTEGER I,J,NYP,NX,NY
        REAL*8 ADT,BDT,UK(NYP,1),FUK(NYP,1),FUKMIN(NYP,1)
        REAL*8 DELT,C,D

        ADT=DELT*C
        BDT=DELT*D

        DO 10 I=1,NY
                DO 10 J=1,NX
                UK(I,J)=UK(I,J)+ADT*FUK(I,J)+BDT*FUKMIN(I,J)
                FUKMIN(I,J)=FUK(I,J)
10      CONTINUE
        END SUBROUTINE

C       ////////////////////////////////////////////////////////////
C
C       ********************************************************************
C       ********************************************************************
C       ********************************************************************
C       ******************(MASS EQATION )***************************
C       ********************************************************************
C       ********************************************************************
```

```
C     ************************************************************
C     ////////////////////////////////////////////////////////////
      SUBROUTINE V_FROM_UILL_TEST()
      INTEGER NX,NY,I,J,NXP,NYP,ERROR,ML,MU,LDA
      REAL*8 LX,LY,DETA,X,DX,Y,ETA,BETA
      REAL*8,ALLOCATABLE :: DY2T_INV(:,:)
      REAL*8,ALLOCATABLE :: INFLOWDU(:),OUTFLOWDU(:)
      REAL*8 U[ALLOCATABLE](:,:),V[ALLOCATABLE](:,:),
     1DX1[ALLOCATABLE](:,:),DX2[ALLOCATABLE](:,:),DY1[ALLOCATABLE](:,:)
     1,DY2[ALLOCATABLE](:,:),PI,DY2T[ALLOCATABLE](:,:)
      NX=100
      NY=100
      NXP=NX
      NYP=NY
      LX=2
      LY=2
      BETA=8.
      ML=NY-2
      MU=NY-2
      LDA=2*ML+MU+1
      ALLOCATE (U(NYP,NXP),V(NYP,NXP),DX1(NXP,NXP),DX2(NXP,NXP),
     1DY1(NYP,NYP),DY2(NYP,NYP),DY2T(NYP,NYP),INFLOWDU(NYP),
     1OUTFLOWDU(NYP),DY2(NYP,NYP),DY2T_INV(NYP,NYP),STAT = ERROR)
      IF (ERROR.NE.0) STOP
      DX=LX/(NX-1)
      PI=4.*DATAN(1._8)
      DETA=1./(NY-1)
      OPEN(1,FILE='V_FROM_UILL.TXT',STATUS='UNKNOWN')
      CALL DRIVED_Y(4,DY1,DY2,NY,NYP,BETA)
      CALL DRIVED_X(4,DX1,DX2,NX,NXP,LX)
      CALL TRANSPOSE(DX1,NX,NXP)
      DO 11 I=2,NY-1
            DO 12 J=2,NY-1
                  DY2T(I-1,J-1)=DY2(I,J)
12          CONTINUE
11    CONTINUE
      CALL INVERSE(DY2T,DY2T_INV,NY-2,NYP,ML,MU,LDA)
      DO 111 I=1,NY
            ETA=(I-1)*DETA
            Y=-BETA/DTAN(PI*ETA)
            INFLOWDU(I)=0.
            OUTFLOWDU(I)=0.
            DO 111 J=1,NX
            X=(J-1)*DX
            U(I,J)=2*X*Y*DEXP(-Y**2.)
111   CONTINUE
      CALL V_FROM_UILL(U,V,INFLOWDU,OUTFLOWDU,NX,NY,NXP,NYP
      1,DX1,DY1,DY2T_INV)
      DO 20 I=1,NY
            ETA=(I-1)*DETA
            Y=-BETA/DTAN(PI*ETA)
            WRITE(1,30) (V(I,J)-(4.-Y**2.),J=1,NX)
30          FORMAT(500(2X,F18.6))
20    CONTINUE
      DEALLOCATE (U,V,DX1,DX2,DY1,DY2,STAT = ERROR)
      IF (ERROR.NE.0) STOP
```

```
      RETURN
      END SUBROUTINE
C     ////////////////////////////////////////////////////////////
C     ////////////////////////////////////////////////////////////
      SUBROUTINE V_FROM_UILL(U,V,INFLOWDU,OUTFLOWDU,
     1      NX,NY,NXP,NYP,DX1,DY1,DY2T_INV)
            INTEGER NX,NY,ERROR,I,J,NYP,NXP
            REAL*8 U(NYP,1),V(NYP,1),DX1(NXP,1),DY1(NYP,1),DY2T_INV(NYP,1)
            REAL*8 INFLOWDU(1),OUTFLOWDU(1)
            REAL*8 TEMPT[ALLOCATABLE](:,:),TEMPV[ALLOCATABLE](:,:)
            ALLOCATE (TEMPT(NYP,NXP),TEMPV(NYP,NXP),STAT = ERROR)
            IF (ERROR.NE.0) STOP 'ERROR ALLOCATE V_FROM_UILL'
            CALL ZARB(U,DX1,TEMPT,NY,NX,NX,NYP,NXP)
            TEMPT(1:NY,1)=INFLOWDU(1:NY)
            TEMPT(1:NY,NX)=OUTFLOWDU(1:NY)
            CALL ZARB(DY1,TEMPT,TEMPV,NY,NY,NX,NYP,NYP)
            DO 200 J=1,NX
                  V(1,J)=0.
                  V(NY,J)=0.
200         CONTINUE
            DO 22 I=1,NY-2
                  DO 12 J=1,NX
                        TEMPT(I,J)=-TEMPV(I+1,J)
12                CONTINUE
22          CONTINUE

            CALL ZARB(DY2T_INV,TEMPT,TEMPV,NY-2,NY-2,NX,NYP,NXP)
            DO 20 I=2,NY-1
                  DO 10 J=1,NX
                        V(I,J)=TEMPV(I-1,J)
10                CONTINUE
20          CONTINUE
            DEALLOCATE (TEMPT,TEMPV,STAT = ERROR)
            IF (ERROR.NE.0) STOP 'ERROR DEALLOCATE V_FROM_UILL'
            RETURN
      END SUBROUTINE
C     ////////////////////////////////////////////////////////////
C     **************************************************************
C     **************************************************************
C     **************************************************************
C     **************************************************************
C     ******************( RIGHT HAND SIDE )**************************
C     **************************************************************
C     **************************************************************
C     COMPUTE RIGHT HAND SIDE OF ROTATIONAL FORM OF NAVIER EQATION
C     ////////////////////////////////////////////////////////////

      SUBROUTINE RHS(U,V,INFLOWDU,OUTFLOWDU,R,DX1T,DX2T,DY1,DY2
     1,AX2_INV,BX2,RE,NX,NY,NXP,NYP,DX,STEP,SUB,DT)

      IMPLICIT NONE
C     DEFINE REQIRED VARIABLES.

      INTEGER NX,NY,ERROR,NXP,NYP,I,J,STEP,SUB,K
```

```
      REAL*8 U(NYP,1),V(NYP,1),R(NYP,1),DX1T(NXP,1),DX2T(NXP,1),DX
      REAL*8 INFLOWDU(1),OUTFLOWDU(1),DT
      REAL*8 DY1(NYP,1),DY2(NYP,1),RE,AX2_INV(NXP,1),BX2(NXP,1)
      REAL*8 H1[ALLOCATABLE](:,:),H2[ALLOCATABLE](:,:),
     1T1[ALLOCATABLE](:,:),T2[ALLOCATABLE](:,:),T3[ALLOCATABLE](:,:)
      REAL*8   T4[ALLOCATABLE](:,:),TEMP[ALLOCATABLE](:,:),
      1UT[ALLOCATABLE](:,:)

      ALLOCATE (H1(NYP,NXP),H2(NYP,NXP),T1(NYP,NXP),T2(NYP,NXP),
     1T3(NYP,NXP),T4(NYP,NXP),TEMP(NYP,NXP),UT(NYP,NXP),STAT=ERROR)
      IF (ERROR.NE.0) STOP
C     CALCULATE DU/DY
      CALL ZARB(DY1,U,T1,NY,NY,NX,NYP,NYP)
C     CALCULATE DX/DV
      CALL ZARB(V,DX1T,T2,NY,NX,NX,NYP,NXP)
C     CALCULATE W=DV/DX-DU/DY
      CALL MINUS(T2,T1,T3,NY,NX,NYP)
      K=NX/5
      IF(SUB==1) THEN
      WRITE(100,100) (STEP-1)*DT,(U(NY/2+1,I) ,I=K,NX,K)
      WRITE(101,100) (STEP-1)*DT,(V(NY/2+1,I) ,I=K,NX,K)
      WRITE(102,100) (STEP-1)*DT,(T3(NY/2+1,I) ,I=K,NX,K)
100   FORMAT(100(2X,F13.8))
      END IF
C     CALCULATE H1=V.*W
      CALL E_ZARB_E(V,T3,H1,NY,NX,NYP)
C     CALCULATE D2H1/DY2
      CALL ZARB(DY2,H1,T2,NY,NY,NX,NYP,NYP)
C     CALCULATE -U.*W
      CALL GARINE(U,T4,NY,NX,NYP)
      CALL E_ZARB_E(T4,T3,H2,NY,NX,NYP)
C     CALCULATE D(-U.*W)/DY
      CALL ZARB(DY1,H2,T4,NY,NY,NX,NYP,NYP)
C     CALCULATE DD(-U.*W)/DYDX
      CALL ZARB(T4,DX1T,T3,NY,NX,NX,NYP,NXP)
      CALL MINUS(T2,T3,T1,NY,NX,NYP)
      CALL ZARB(U,BX2,TEMP,NY,NX,NX,NYP,NXP)
      DO 30 I=1,NY
            TEMP(I,1)=TEMP(I,1) -3./DX*INFLOWDU(I)
            TEMP(I,NX)=TEMP(I,NX) +3./DX*OUTFLOWDU(I)
30    CONTINUE
      CALL ZARB(TEMP,AX2_INV,T2,NY,NX,NX,NYP,NXP)
      CALL ZARB(DY2,U,T3,NY,NY,NX,NYP,NYP)
      CALL SUM(T2,T3,T4,NY,NX,NYP)
      CALL ZARB(DY2,T4,H1,NY,NY,NX,NYP,NYP)
      CALL ZARB(T4,DX2T,H2,NY,NX,NX,NYP,NXP)
      CALL SUM(H1,H2,T2,NY,NX,NYP)
      DO 10 I=1,NY
            DO 20 J=1,NX
                  R(I,J)=     T1(I,J)+T2(I,J)/RE
20          CONTINUE
10    CONTINUE
      DEALLOCATE (H1,H2,T1,T2,T3,T4,TEMP,UT,STAT = ERROR)
      IF (ERROR.NE.0) STOP
      RETURN
```

```
      END SUBROUTINE

C     ////////////////////////////////////////////////////////////
C     ***************************************************************
C     ******************(TEST OF DRIVATE MATRIX'S)*********************
C     ***************************************************************
C     ////////////////////////////////////////////////////////////
c     ////////////////////////////////////////////////////////////
C     ////////////////////////////////////////////////////////////
      SUBROUTINE DRIVATE_TEST_X1()
      INTEGER N,NP,I,ERROR,J
      REAL*8 LX,DX,X
      REAL*8 D2[ALLOCATABLE](:,:),F1[ALLOCATABLE](:,:),
     1F2[ALLOCATABLE](:,:),C[ALLOCATABLE](:,:),D1[ALLOCATABLE](:,:)
      PRINT *,'PLEASE ENTER N'
      READ *,N
      NP=N+1
      ALLOCATE (D2(NP,NP),F1(NP,NP),F2(NP,NP),C(NP,NP),
     1D1(NP,NP),STAT = ERROR)
      IF (ERROR.NE.0) STOP
      LX=1.
      DX=LX/(N-1)
      CALL DRIVED_X(4,D1,D2,N,NP,LX)
      CALL FUNC1(F1,N,NP,LX)
      CALL ZARB(D1,F1,C,N,N,1,NP,NP)
      CALL FUNCP1(F2,N,NP,LX)
      OPEN(1,FILE='DRIVATE_TEST_X1.TXT',STATUS='UNKNOWN')
      DO 21 I=1,N
            X=(I-1)*DX
            WRITE(1,22) X,F1(I,1),C(I,1),F2(I,1),DABS((C(I,1)-F2(I,1)))
22          FORMAT(400(2X,F18.8))
21    CONTINUE
      CLOSE(1)
      RETURN
      END SUBROUTINE
C     ////////////////////////////////////////////////////////////
C
      SUBROUTINE DRIVATE_TEST_X1_BOUNDARY()
      INTEGER N,NP,I,ERROR,J
      REAL*8 LX,DX,X
      REAL*8 D2[ALLOCATABLE](:,:),F1[ALLOCATABLE](:,:),
     1F2[ALLOCATABLE](:,:),C[ALLOCATABLE](:,:),D1[ALLOCATABLE](:,:)
      PRINT *,'PLEASE ENTER N'
      READ *,N
      NP=N+1
      ALLOCATE (D2(NP,NP),F1(NP,NP),F2(NP,NP),C(NP,NP),
     1D1(NP,NP),STAT = ERROR)
      IF (ERROR.NE.0) STOP
      LX=1.0
      DX=LX/(N-1)
      CALL DRIVED_X(4,D1,D2,N,NP,LX)
      CALL FUNC1(F1,N,NP,LX)
      CALL ZARB(D1,F1,C,N,N,1,NP,NP)
      CALL FUNCP1(F2,N,NP,LX)
C     C(1,1)=F2(1,1)
```

```
C       C(N,1)=F2(N,1)
        OPEN(1,FILE='DRIVATE_TEST_X1_BOUNDARY.TXT',STATUS='UNKNOWN')
        DO 21 I=1,N
                X=(I-1)*DX
                WRITE(1,22) X,F1(I,1),C(I,1),F2(I,1),DABS((C(I,1)-F2(I,1)))
22              FORMAT(400(2X,F18.8))
21      CONTINUE
        CLOSE(1)
        RETURN
        END SUBROUTINE
C       ////////////////////////////////////////////////////////////
C
C       ////////////////////////////////////////////////////////////
        SUBROUTINE DRIVATE_TEST_X1_2()
        INTEGER N,NP,I,ERROR,J
        REAL*8 LX,DX,X
        REAL*8 D2[ALLOCATABLE](:,:),F1[ALLOCATABLE](:,:),
     1F2[ALLOCATABLE](:,:),C[ALLOCATABLE](:,:),D1[ALLOCATABLE](:,:)
        PRINT *,'PLEASE ENTER N'
        READ *,N
        NP=N+1
        ALLOCATE (D2(NP,NP),F1(NP,NP),F2(NP,NP),C(NP,NP),
     1D1(NP,NP),STAT = ERROR)
         IF (ERROR.NE.0) STOP
        LX=2
        DX=LX/(N-1)
        CALL DRIVED_X(4,D1,D2,N,NP,LX)
        CALL FTEST(F1,F2,N,NP,LX)
        CALL ZARB(D1,F1,C,N,N,1,NP,NP)
        OPEN(1,FILE='DRIVATE_TEST_X1_2.TXT',STATUS='UNKNOWN')
        DO 21 I=1,N
                X=(I-1)*DX
                WRITE(1,22) X,C(I,1),F2(I,1),DABS((C(I,1)-F2(I,1))/F2(I,1))
22      FORMAT(5F18.7)
21      CONTINUE
        CLOSE(1)
        END SUBROUTINE
C       ////////////////////////////////////////////////////////////
C       ////////////////////////////////////////////////////////////
        SUBROUTINE DRIVATE_TEST_X2()
        INTEGER N,NP,I,ERROR,J
        REAL*8 LX,DX,X
        REAL*8 D2[ALLOCATABLE](:,:),F1[ALLOCATABLE](:,:),
     1F2[ALLOCATABLE](:,:),C[ALLOCATABLE](:,:),D1[ALLOCATABLE](:,:)
        PRINT *,'PLEASE ENTER N'
        READ *,N
        NP=N+1
        ALLOCATE (D2(NP,NP),F1(NP,NP),F2(NP,NP),C(NP,NP),
     1D1(NP,NP),STAT = ERROR)
        IF (ERROR.NE.0) STOP
        LX=1.
        DX=LX/(N-1)
        CALL DRIVED_X(4,D1,D2,N,NP,LX)
        CALL FUNC2(F1,N,NP,LX)
        CALL ZARB(D2,F1,C,N,N,1,NP,NP)
        CALL FUNCP2(F2,N,NP,LX)
```

```
      OPEN(1,FILE='DRIVATE_TEST_X2.TXT',STATUS='UNKNOWN')
      DO 21 I=1,N
            X=(I-1)*DX
            WRITE(1,22) X,F1(I,1),C(I,1),F2(I,1),DABS((C(I,1)-F2(I,1)))
22          FORMAT(400(2X,F18.8))
21    CONTINUE
      CLOSE(1)
      RETURN
      END SUBROUTINE
C     ////////////////////////////////////////////////////////
C
C     ////////////////////////////////////////////////////////
      SUBROUTINE DRIVATE_TEST_X2_BOUNDARY()
      INTEGER N,NP,I,ERROR,J
      REAL*8 LX,DX,AL,X
      REAL*8 B[ALLOCATABLE](:,:),F1[ALLOCATABLE](:,:),
     1F2[ALLOCATABLE](:,:),C[ALLOCATABLE](:,:),A[ALLOCATABLE](:,:)
      REAL*8   G[ALLOCATABLE](:,:),G1[ALLOCATABLE](:,:)
      PRINT *,'PLEASE ENTER N'
      READ *,N
      NP=N
      ALLOCATE (B(NP,NP),F1(NP,NP),F2(NP,NP),C(NP,NP),
     1A(NP,NP),G(NP,NP),G1(NP,NP),STAT = ERROR)
      IF (ERROR.NE.0) STOP
      LX=1.5
      DX=LX/(N-1)
      AL=1./4.
      CALL PADEBUELL2(2,AL,DX,N,NP,A,B)
      DO 10 I=1,N
            X=(I-1)*DX
            F1(I,1)=DSIN(X)
10    CONTINUE
      CALL ZARB(B,F1,C,N,N,1,NP,NP)
            C(1,1)=C(1,1)-3./DX*DCOS(DBLE(0))
            C(N,1)=C(N,1)+3./DX*DCOS(DBLE(1.5))
      CALL INVERSE(A,G,N,NP,1,1,4)
      CALL ZARB(G,C,F2,N,N,1,NP,NP)
      CALL DRIVED_X(4,A,B,N,NP,LX)
      CALL ZARB(B,F1,G1,N,N,N,NP,NP)
      OPEN(1,FILE='DRIVATE_TEST_X2_BOUNDARY.TXT',STATUS='UNKNOWN')
      DO 21 I=1,N
      X=(I-1)*DX
      WRITE(1,22) X,DABS((-DSIN(X)-F2(I,1))),DABS((-DSIN(X)-G1(I,1)))
22    FORMAT(500(F18.10,2X))
21    CONTINUE
      CLOSE(1)
      END SUBROUTINE
C     ////////////////////////////////////////////////////////
C
C     ////////////////////////////////////////////////////////
      SUBROUTINE DRIVATE_TEST_Y1()
      INTEGER N,NP,I,ERROR,J
      REAL*8 BETA,ETA,DETA,Y
      REAL*8 D2[ALLOCATABLE](:,:),F1[ALLOCATABLE](:,:),
     1F2[ALLOCATABLE](:,:),C[ALLOCATABLE](:,:),D1[ALLOCATABLE](:,:)
      PRINT *,'PLEASE ENTER N'
```

```
      READ *,N
      NP=N+1
      ALLOCATE (D2(NP,NP),F1(NP,NP),F2(NP,NP),C(NP,NP),
     1D1(NP,NP),STAT = ERROR)
       IF (ERROR.NE.0) STOP
      BETA=4.0
      PI=4.*DATAN(DBLE(1.0))
      DETA=1./(N-1)
      CALL DRIVED_Y(4,D1,D2,N,NP,BETA)
      CALL FMAPING1(F1,N,NP,BETA)
      CALL FMAPINGP1(F2,N,NP,BETA)
      CALL ZARB(D1,F1,C,N,N,1,NP,NP)
      OPEN(1,FILE='DRIVATE_TEST_Y1.TXT',STATUS='UNKNOWN')
      DO 21 I=1,N
            ETA=(I-1)*DETA
            Y=-BETA/DTAN(PI*ETA)
            WRITE(1,22) Y,F1(I,1),F2(I,1),C(I,1) ,DABS((C(I,1)-F2(I,1)))
22          FORMAT(400(2X,F18.6))
21    CONTINUE
      CLOSE(1)
      END SUBROUTINE
C     ////////////////////////////////////////////////////////////
C
C     ////////////////////////////////////////////////////////////
      SUBROUTINE DRIVATE_TEST_Y2()
      INTEGER N,NP,I,ERROR,J
      REAL*8 BETA,ETA,DETA,Y,PI
      REAL*8 D2[ALLOCATABLE](:,:),F1[ALLOCATABLE](:,:),
     1F2[ALLOCATABLE](:,:),C[ALLOCATABLE](:,:),D1[ALLOCATABLE](:,:)
      PRINT *,'PLEASE ENTER N'
      READ *,N
      NP=N+1
      ALLOCATE (D2(NP,NP),F1(NP,NP),F2(NP,NP),C(NP,NP),
     1D1(NP,NP),STAT = ERROR)
      IF (ERROR.NE.0) STOP
      BETA=4.0
      PI=4.*DATAN(1.0_8)
      DETA=1./(N-1)
      CALL DRIVED_Y(4,D1,D2,N,NP,BETA)
      CALL FMAPING1(F1,N,NP,BETA)
      CALL FMAPINGP2(F2,N,NP,BETA)
      CALL ZARB(D2,F1,C,N,N,1,NP,NP)
      OPEN(1,FILE='DRIVATE_TEST_Y2.TXT',STATUS='UNKNOWN')
      DO 21 I=1,N
            ETA=(I-1)*DETA
            Y=-BETA/DTAN(PI*ETA)
            WRITE(1,22) Y,F2(I,1),C(I,1),DABS((C(I,1)-F2(I,1)))
22    FORMAT(4F18.7)
21    CONTINUE
      CLOSE(1)
      END SUBROUTINE
C     ////////////////////////////////////////////////////////////
C     ////////////////////////////////////////////////////////////
C
      SUBROUTINE FTEST(F,F1,N,NP,LX)
      INTEGER N,NP,I,J,M1,ERROR
```

```
      INTEGER H,M,S,MS
      REAL P
      REAL*8 LX,F(NP,1),F1(NP,1),A[ALLOCATABLE](:)
      REAL*8 SUM1,SUM2,X,DX
      ALLOCATE (A(NP),STAT=ERROR)
      IF (ERROR.NE.0) STOP
      CALL GETTIM(H,M,S,MS)
   CALL SEED(MS)
      DX=LX/(N-1)
      M1=30
      DO 30 J=1,M1
            CALL RANDOM(P)
            A(J)=P*10
30    CONTINUE
      DO 10 I=1,N
      SUM1=0.
      SUM2=0.
      X=(I-1)*DX
      DO 20 J=1,M1
            SUM1=SUM1+A(J)*(DSIN(X)+X*J*DCOS(J*X))
            SUM2=SUM2+A(J)*(DCOS(X)-X*(J**2)*DSIN(J*X)+J*DCOS(J*X))
20    CONTINUE
      F(I,1)=SUM1
      F1(I,1)=SUM2
10    CONTINUE
      RETURN
      END SUBROUTINE
C     ////////////////////////////////////////////////////////
C
C     ////////////////////////////////////////////////////////
      SUBROUTINE FUNCP1(F2,N,NP,LX)
            INTEGER N,NP,I
            REAL*8 F2(NP,1),DX,LX,X
            DX=LX/(N-1)
            DO 10 I=1,N
                  X=(I-1)*DX
                  F2(I,1)=DEXP(X)+X*DEXP(X)
10          CONTINUE
            RETURN
      END SUBROUTINE

C     ////////////////////////////////////////////////////////

      SUBROUTINE FUNC1(F2,N,NP,LX)
            INTEGER N,NP,I
            REAL*8 F2(NP,1),DX,X,LX
            DX=LX/(N-1)
            DO 10 I=1,N
                  X=(I-1)*DX
                  F2(I,1)=DEXP(X)*X
10          CONTINUE
            RETURN
      END SUBROUTINE

C     ////////////////////////////////////////////////////////
```

```
C       ////////////////////////////////////////////////////////////
        SUBROUTINE FUNCP2(F2,N,NP,LX)
                INTEGER N,NP,I
                REAL*8 F2(NP,1),DX,LX,X
                DX=LX/(N-1)
                DO 10 I=1,N
                        X=(I-1)*DX
                        F2(I,1)=4*X**2*DEXP(X**2)+2.*DEXP(X**2)
10              CONTINUE
                RETURN
        END SUBROUTINE

C       ////////////////////////////////////////////////////////////

        SUBROUTINE FUNC2(F2,N,NP,LX)
                INTEGER N,NP,I
                REAL*8 F2(NP,1),DX,X,LX
                DX=LX/(N-1)
                DO 10 I=1,N
                        X=(I-1)*DX
                        F2(I,1)=DEXP(X**2)
10              CONTINUE
                RETURN
        END SUBROUTINE

C       ////////////////////////////////////////////////////////////
C       ////////////////////////////////////////////////////////////
        SUBROUTINE FMAPINGP2(F2,N,NP,BETA)
                INTEGER N,NP,I
                REAL*8 F2(NP,1),BETA,DETA,ETA,Y,PI
                PI=4.*DATAN(1._8)
                DETA=1.0/(N-1)
                DO 10 I=1,N
                        ETA=(I-1)*DETA
                        Y=-BETA/DTAN(PI*ETA)
                        F2(I,1)=-2.*DEXP(-Y**2.)+4.*Y**2.*DEXP(-Y**2.)
10              CONTINUE
                RETURN
        END SUBROUTINE

C       ////////////////////////////////////////////////////////////

C       ////////////////////////////////////////////////////////////
        SUBROUTINE FMAPINGP1(F2,N,NP,BETA)
                INTEGER N,NP,I
                REAL*8 F2(NP,1),BETA,DETA,PI,ETA,Y
                PI=4.*DATAN(1._8)
                DETA=1.0/(N-1)
                DO 10 I=1,N
                        ETA=(I-1)*DETA
                        Y=-BETA/DTAN(PI*ETA)
                        F2(I,1)=-2.*Y*DEXP(-Y**2)
10              CONTINUE
                RETURN
```

```
      END SUBROUTINE

C     ////////////////////////////////////////////////////////
      SUBROUTINE FMAPING1(F2,N,NP,BETA)
            INTEGER N,NP,I
            REAL*8 F2(NP,1),DETA,Y,ETA,BETA,PI,LY,LETA
            PI=4.*ATAN(1.0)
            DETA=1.0/(N-1)
            DO 10 I=1,N
                  ETA=(I-1)*DETA
                  Y=-BETA/DTAN(PI*ETA)
                  F2(I,1)=DEXP(-Y**2.)
10          CONTINUE
            RETURN
      END SUBROUTINE
C     ////////////////////////////////////////////////////////
C     ////////////////////////////////////////////////////////
C     ****************************************************************
C     ****************************************************************
C     ********************( LINEAR EQUATIONS SOLVERS)********************
C     ****************************************************************
C     ****************************************************************
      SUBROUTINE GAUSSJ_TEST()
      INTEGER I,J,N,ERROR,H,M1,S,MS,M,NP,MP
      REAL*8 A0[ALLOCATABLE](:,:),B0[ALLOCATABLE](:,:),
     1C0[ALLOCATABLE](:,:),D0[ALLOCATABLE](:,:)
      PRINT *,"PLEASE ENTER NUMBER OF ROWS OR COLUMNS(A)"
      READ(*,*) N
      NP=N+1
      PRINT *,"PLEASE ENTER NUMBER OF COLUMNS(B)"
      READ(*,*) M
      MP=M+1
      ALLOCATE (A0(NP,NP),B0(NP,MP),C0(NP,MP),D0(NP,NP),
     1STAT=ERROR)
      IF (ERROR.NE.0) STOP

      DO 100 I=1,N
            DO 100 J=1,M
                  IF (I.EQ.J) THEN
                        B0(I,I)=1.
                  ENDIF
100   CONTINUE
      CALL GETTIM(H,M1,S,MS)
     CALL SEED(MS)
      DO 110 I=1,N
            DO 110 J=1,N
            CALL RANDOM(P)
            A0(I,J)=P
110   CONTINUE
      DO 200 I=1,N
            DO 200 J=I,N
                  D0(I,J)=A0(I,J)
200   CONTINUE
      CALL GAUSSJ(A0,N,NP,B0,M)
      CALL ZARB(D0,B0,C0,N,N,M,NP,NP)
```

```
      OPEN(2,FILE='GAUSSJ_TEST.TXT',STATUS='UNKNOWN')
      DO 10 I=1,N
            WRITE(2,20)(C0(I,J),J=1,M)
20          FORMAT(100(2X,F20.10))
10    CONTINUE
      DEALLOCATE (A0,B0,C0,D0,STAT = ERROR)
      IF (ERROR.NE.0) STOP
      RETURN
      END SUBROUTINE
C
C     //////////////////////////////////////////////////////////
C
C
C     //////////////////////////////////////////////////////////
      SUBROUTINE GAUSSJ(A,N,NP,B,M)
      INTEGER M,N,NP,ERROR
      REAL*8 A(NP,1),B(NP,1)
      REAL*8 BIG,DUM,PIVINV
      INTEGER I,ICOL,IROW,J,K,L,LL
      INTEGER INDXC[ALLOCATABLE](:),INDXR[ALLOCATABLE](:),
     1IPIV[ALLOCATABLE](:)
     ALLOCATE (INDXC(NP),INDXR(NP),IPIV(NP),STAT=ERROR)

      IF (ERROR.NE.0) STOP
      DO 11 J=1,N
            IPIV(J)=0
11    CONTINUE
      DO 22 I=1,N
            BIG=0.
            DO 13 J=1,N
                  IF(IPIV(J).NE.1)THEN
                  DO 12 K=1,N
                        IF (IPIV(K).EQ.0) THEN
                              IF (ABS(A(J,K)).GE.BIG)THEN
                                    BIG=ABS(A(J,K))
                                    IROW=J
                                    ICOL=K
                              ENDIF
                        ENDIF
12                CONTINUE
                  ENDIF
13          CONTINUE
            IPIV(ICOL)=IPIV(ICOL)+1
            IF (IROW.NE.ICOL) THEN
                  DO 14 L=1,N
                        DUM=A(IROW,L)
                        A(IROW,L)=A(ICOL,L)
                        A(ICOL,L)=DUM
14                CONTINUE
                  DO 15 L=1,M
                        DUM=B(IROW,L)
                        B(IROW,L)=B(ICOL,L)
                        B(ICOL,L)=DUM
15                CONTINUE
            ENDIF
```

```
                INDXR(I)=IROW
                INDXC(I)=ICOL
                IF (A(ICOL,ICOL).EQ.0.) PAUSE
                PIVINV=1./A(ICOL,ICOL)
                A(ICOL,ICOL)=1.
                DO 16 L=1,N
                        A(ICOL,L)=A(ICOL,L)*PIVINV
16              CONTINUE
                DO 17 L=1,M
                        B(ICOL,L)=B(ICOL,L)*PIVINV
17              CONTINUE
                DO 21 LL=1,N
                        IF(LL.NE.ICOL) THEN
                                DUM=A(LL,ICOL)
                                A(LL,ICOL)=0.
                                DO 18 L=1,N
                                        A(LL,L)=A(LL,L)-A(ICOL,L)*DUM
18                              CONTINUE
                                DO 19 L=1,M
                                        B(LL,L)=B(LL,L)-B(ICOL,L)*DUM
19                              CONTINUE
                        ENDIF
21              CONTINUE
22      CONTINUE
        DO 24 L=N,1,-1
                IF(INDXR(L).NE.INDXC(L))THEN
                        DO 23 K=1,N
                                DUM=A(K,INDXR(L))
                                A(K,INDXR(L))=A(K,INDXC(L))
                                A(K,INDXC(L))=DUM
23                      CONTINUE
                ENDIF
24      CONTINUE
        DEALLOCATE (INDXC,INDXR,IPIV,STAT = ERROR)
        IF (ERROR.NE.0) STOP
        RETURN
        END
C       ////////////////////////////////////////////////////////////
C       ////////////////////////////////////////////////////////////
C
C       ////////////////////////////////////////////////////////////
        SUBROUTINE INVERSE_TEST()
        INTEGER I,J,ML,MU,LDA,NX,NY,ERROR
        REAL*8 A[ALLOCATABLE](:,:),B[ALLOCATABLE](:,:),C[ALLOCATABLE](:,:)
  1,D[ALLOCATABLE](:,:)
        REAL*8 LX,BETA
        INTEGER H,M,S,MS
        PRINT *,"PLEASE ENTER NUMBER OF ROWS OR COLUMNS"
        READ(*,*) NX
        NXP=NX
        ALLOCATE (A(NXP,NXP),B(NXP,NXP),C(NXP,NXP),D(NXP,NXP),STAT=ERROR)
        IF (ERROR.NE.0) STOP
        LX=2.
        BETA=1.
        NY=NX
        CALL GETTIM(H,M,S,MS)
```

```
      CALL SEED(MS)
            DO 10 I=1,NX
                  DO 10 J=1,NY
                  CALL RANDOM(P)
                  A(I,J)=P
10    CONTINUE
            ML=10
            MU=ML
            LDA=2*ML+MU+1
            CALL ZEROS(C,NX,NX,NX)
            CALL INVERSE(A,D,NX,NXP,ML,MU,LDA)
            CALL ZARB(D,A,C,NX,NX,NY,NXP,NXP)
            OPEN(1,FILE='INVERSE_TEST.TXT',STATUS='UNKNOWN')
            DO 20 I=1,NY
                  WRITE(1,30) (C(I,J),J=1,NX)
30                FORMAT(100(2X,F18.5))
20          CONTINUE
            DEALLOCATE (A,B,C,D,STAT = ERROR)
            IF (ERROR.NE.0) STOP
            RETURN
            END SUBROUTINE

C           ////////////////////////////////////////////////////////
C           ////////////////////////////////////////////////////////
            SUBROUTINE INVERSE(A,C,N,NP,ML,MU,LDA)
            INTEGER N,NP,ML,MU,LDA,LJ,INFO,ERROR
            DOUBLE PRECISION A(NP,1),C(NP,1)
            REAL*8 ABD[ALLOCATABLE](:,:)
            INTEGER IPVT[ALLOCATABLE](:)
            ALLOCATE (ABD(LDA,NP),IPVT(NP),STAT = ERROR)
            IF (ERROR.NE.0) STOP
C            COMPUTES C(NP,NP) AS THE INVERSE OF A(NP,NP)
C     WHERE A IS A DIAGOANL MATRIX
C     WITH ML SUB-DIAGONAL AND MU SUPPERDIAGONAL MATRIX




C                 INITIALIZE C
                  DO 10 I=1,N
                              C(I,I)=1.
10                CONTINUE
                  CALL BANDFORM(N,ML,MU,A,ABD,NP,LDA)
                  CALL DGBFA(ABD,LDA,N,ML,MU,IPVT,INFO)

                  IF (INFO.NE.0.) THEN
                        IF(INFO.GT.0) THEN
                              WRITE(*,15) INFO
15                            FORMAT('THE 'I3' TH ARGUMENT HAD AN ILLEGAL VALUE')
                        ELSE
                              WRITE(*,16)
16                            FORMAT('FACTOR U IS SINGULAR')
                        END IF
```

```
                END IF

                J=0
                DO 17 I=1,N

                        CALL DGBSL(ABD,LDA,N,ML,MU,IPVT,C(1,I),J)

17              CONTINUE
C               AX=C -->X=INVERSE(A)C (IF C=I (IDENTITY MATRIX))
C               THEN X=INVERSE(A)
                DEALLOCATE (ABD,IPVT)
                IF (ERROR.NE.0) STOP
                RETURN
                END
C       ////////////////////////////////////////////////////////////
C
C       ////////////////////////////////////////////////////////////
        SUBROUTINE BANDFORM(N,ML,MU,A,ABD,NP,LDA)
C               RELATES ABD(2*ML+MU+1,NP) TO A(NP,NP) IN ORDER
C               TO USE SGI SOLVER (DGBFA,DGBSL)
                INTEGER N,ML,MU,NP,LDA
                REAL*8 A(NP,1),ABD(LDA,1)
                INTEGER I,I1,I2,J,K,M
                M=ML+MU+1
                DO 11 J=1,N
                        I1=MAX0(1,J-MU)
                        I2=MIN0(N,J+ML)
                        DO 11 I=I1,I2
                                K=I-J+M
                                ABD(K,J)=A(I,J)
11              CONTINUE
                RETURN
        END
C       ////////////////////////////////////////////////////////////
C
C       ////////////////////////////////////////////////////////////

        SUBROUTINE DGBFA(ABD,LDA,N,ML,MU,IPVT,INFO)
    INTEGER LDA,N,ML,MU,IPVT(1),INFO
    DOUBLE PRECISION ABD(LDA,1)
C
C   DGBFA FACTORS A DOUBLE PRECISION BAND MATRIX BY ELIMINATION.
C
C   DGBFA IS USUALLY CALLED BY DGBCO, BUT IT CAN BE CALLED
C   DIRECTLY WITH A SAVING IN TIME IF  RCOND  IS NOT NEEDED.
C
C   ON ENTRY
C
C     ABD   DOUBLE PRECISION(LDA, N)
C           CONTAINS THE MATRIX IN BAND STORAGE.  THE COLUMNS
C           OF THE MATRIX ARE STORED IN THE COLUMNS OF  ABD  AND
C           THE DIAGONALS OF THE MATRIX ARE STORED IN ROWS
C           ML+1 THROUGH 2*ML+MU+1 OF  ABD .
C           SEE THE COMMENTS BELOW FOR DETAILS.
```

```
C
C     LDA    INTEGER
C            THE LEADING DIMENSION OF THE ARRAY  ABD .
C            LDA MUST BE .GE. 2*ML + MU + 1 .
C
C     N      INTEGER
C            THE ORDER OF THE ORIGINAL MATRIX.
C
C     ML     INTEGER
C            NUMBER OF DIAGONALS BELOW THE MAIN DIAGONAL.
C            0 .LE. ML .LT. N .
C
C     MU     INTEGER
C            NUMBER OF DIAGONALS ABOVE THE MAIN DIAGONAL.
C            0 .LE. MU .LT. N .
C            MORE EFFICIENT IF  ML .LE. MU .
C   ON RETURN
C
C     ABD    AN UPPER TRIANGULAR MATRIX IN BAND STORAGE AND
C            THE MULTIPLIERS WHICH WERE USED TO OBTAIN IT.
C            THE FACTORIZATION CAN BE WRITTEN  A = L*U  WHERE
C            L  IS A PRODUCT OF PERMUTATION AND UNIT LOWER
C            TRIANGULAR MATRICES AND  U  IS UPPER TRIANGULAR.
C
C     IPVT   INTEGER(N)
C            AN INTEGER VECTOR OF PIVOT INDICES.
C
C     INFO   INTEGER
C            = 0  NORMAL VALUE.
C            = K  IF  U(K,K) .EQ. 0.0 .  THIS IS NOT AN ERROR
C                 CONDITION FOR THIS SUBROUTINE, BUT IT DOES
C                 INDICATE THAT DGBSL WILL DIVIDE BY ZERO IF
C                 CALLED.  USE  RCOND  IN DGBCO FOR A RELIABLE
C                 INDICATION OF SINGULARITY.
C
C   BAND STORAGE
C
C      IF  A  IS A BAND MATRIX, THE FOLLOWING PROGRAM SEGMENT
C      WILL SET UP THE INPUT.
C
C              ML = (BAND WIDTH BELOW THE DIAGONAL)
C              MU = (BAND WIDTH ABOVE THE DIAGONAL)
C              M = ML + MU + 1
C              DO 20 J = 1, N
C                I1 = MAX0(1, J-MU)
C                I2 = MIN0(N, J+ML)
C                DO 10 I = I1, I2
C                  K = I - J + M
C                  ABD(K,J) = A(I,J)
C           10   CONTINUE
C           20 CONTINUE
C
C      THIS USES ROWS  ML+1  THROUGH  2*ML+MU+1  OF  ABD .
C      IN ADDITION, THE FIRST  ML  ROWS IN  ABD  ARE USED FOR
C      ELEMENTS GENERATED DURING THE TRIANGULARIZATION.
C      THE TOTAL NUMBER OF ROWS NEEDED IN  ABD  IS  2*ML+MU+1 .
```

```
C        THE  ML+MU BY ML+MU  UPPER LEFT TRIANGLE AND THE
C        ML BY ML  LOWER RIGHT TRIANGLE ARE NOT REFERENCED.
C
C     LINPACK. THIS VERSION DATED 08/14/78 .
C     CLEVE MOLER, UNIVERSITY OF NEW MEXICO, ARGONNE NATIONAL LAB.
C
C     SUBROUTINES AND FUNCTIONS
C
C     BLAS DAXPY,DSCAL,IDAMAX
C     FORTRAN MAX0,MIN0
C
C     INTERNAL VARIABLES
C
      DOUBLE PRECISION T
      INTEGER I,IDAMAX,I0,J,JU,JZ,J0,J1,K,KP1,L,LM,M,MM,NM1
C
C
      M = ML + MU + 1
      INFO = 0
C
C     ZERO INITIAL FILL-IN COLUMNS
C
      J0 = MU + 2
      J1 = MIN0(N,M) - 1
      IF (J1 .LT. J0) GO TO 30
      DO 20 JZ = J0, J1
        I0 = M + 1 - JZ
        DO 10 I = I0, ML
          ABD(I,JZ) = 0.0D0
   10   CONTINUE
   20 CONTINUE
   30 CONTINUE
      JZ = J1
      JU = 0
C
C     GAUSSIAN ELIMINATION WITH PARTIAL PIVOTING
C
      NM1 = N - 1
      IF (NM1 .LT. 1) GO TO 130
      DO 120 K = 1, NM1
        KP1 = K + 1
C
C       ZERO NEXT FILL-IN COLUMN
C
        JZ = JZ + 1
        IF (JZ .GT. N) GO TO 50
        IF (ML .LT. 1) GO TO 50
          DO 40 I = 1, ML
            ABD(I,JZ) = 0.0D0
   40     CONTINUE
   50   CONTINUE
C
C       FIND L = PIVOT INDEX
C
```

```
      LM = MIN0(ML,N-K)
      L = IDAMAX(LM+1,ABD(M,K),1) + M - 1
      IPVT(K) = L + K - M
C
C     ZERO PIVOT IMPLIES THIS COLUMN ALREADY TRIANGULARIZED
C
      IF (ABD(L,K) .EQ. 0.0D0) GO TO 100
C
C        INTERCHANGE IF NECESSARY
C
         IF (L .EQ. M) GO TO 60
           T = ABD(L,K)
           ABD(L,K) = ABD(M,K)
           ABD(M,K) = T
   60    CONTINUE
C
C        COMPUTE MULTIPLIERS
C
         T = -1.0D0/ABD(M,K)
         CALL DSCAL(LM,T,ABD(M+1,K),1)
C
C        ROW ELIMINATION WITH COLUMN INDEXING
C
         JU = MIN0(MAX0(JU,MU+IPVT(K)),N)
         MM = M
         IF (JU .LT. KP1) GO TO 90
         DO 80 J = KP1, JU
           L = L - 1
           MM = MM - 1
           T = ABD(L,J)
           IF (L .EQ. MM) GO TO 70
             ABD(L,J) = ABD(MM,J)
             ABD(MM,J) = T
   70      CONTINUE
           CALL DAXPY(LM,T,ABD(M+1,K),1,ABD(MM+1,J),1)
   80    CONTINUE
   90    CONTINUE
      GO TO 110
  100 CONTINUE
         INFO = K
  110 CONTINUE
  120 CONTINUE
  130 CONTINUE
      IPVT(N) = N
      IF (ABD(M,N) .EQ. 0.0D0) INFO = N
      RETURN
      END
      SUBROUTINE DGBSL(ABD,LDA,N,ML,MU,IPVT,B,JOB)
      INTEGER LDA,N,ML,MU,IPVT(1),JOB
      DOUBLE PRECISION ABD(LDA,1),B(1)
C
C     DGBSL SOLVES THE DOUBLE PRECISION BAND SYSTEM
C     A * X = B  OR  TRANS(A) * X = B
C     USING THE FACTORS COMPUTED BY DGBCO OR DGBFA.
C
C     ON ENTRY
```

```
C
C     ABD    DOUBLE PRECISION(LDA, N)
C            THE OUTPUT FROM DGBCO OR DGBFA.
C
C     LDA    INTEGER
C            THE LEADING DIMENSION OF THE ARRAY  ABD .
C
C     N      INTEGER
C            THE ORDER OF THE ORIGINAL MATRIX.
C
C     ML     INTEGER
C            NUMBER OF DIAGONALS BELOW THE MAIN DIAGONAL.
C
C     MU     INTEGER
C            NUMBER OF DIAGONALS ABOVE THE MAIN DIAGONAL.
C
C     IPVT   INTEGER(N)
C            THE PIVOT VECTOR FROM DGBCO OR DGBFA.
C
C     B      DOUBLE PRECISION(N)
C            THE RIGHT HAND SIDE VECTOR.
C
C     JOB    INTEGER
C            = 0        TO SOLVE  A*X = B ,
C            = NONZERO  TO SOLVE  TRANS(A)*X = B , WHERE
C                       TRANS(A)  IS THE TRANSPOSE.
C
C   ON RETURN
C
C     B      THE SOLUTION VECTOR  X .
C
C   ERROR CONDITION
C
C     A DIVISION BY ZERO WILL OCCUR IF THE INPUT FACTOR CONTAINS A
C     ZERO ON THE DIAGONAL.  TECHNICALLY THIS INDICATES SINGULARITY
C     BUT IT IS OFTEN CAUSED BY IMPROPER ARGUMENTS OR IMPROPER
C     SETTING OF LDA .  IT WILL NOT OCCUR IF THE SUBROUTINES ARE
C     CALLED CORRECTLY AND IF DGBCO HAS SET RCOND .GT. 0.0
C     OR DGBFA HAS SET INFO .EQ. 0 .
C
C   TO COMPUTE  INVERSE(A) * C  WHERE  C  IS A MATRIX
C   WITH  P  COLUMNS
C        CALL DGBCO(ABD,LDA,N,ML,MU,IPVT,RCOND,Z)
C        IF (RCOND IS TOO SMALL) GO TO ...
C        DO 10 J = 1, P
C           CALL DGBSL(ABD,LDA,N,ML,MU,IPVT,C(1,J),0)
C     10 CONTINUE
C
C   LINPACK. THIS VERSION DATED 08/14/78 .
C   CLEVE MOLER, UNIVERSITY OF NEW MEXICO, ARGONNE NATIONAL LAB.
C
C   SUBROUTINES AND FUNCTIONS
C
C   BLAS DAXPY,DDOT
C   FORTRAN MIN0
C
```

```
C   INTERNAL VARIABLES
C
    DOUBLE PRECISION DDOT,T
    INTEGER K,KB,L,LA,LB,LM,M,NM1
C
    M = MU + ML + 1
    NM1 = N - 1
    IF (JOB .NE. 0) GO TO 50
C
C     JOB = 0 , SOLVE  A * X = B
C     FIRST SOLVE L*Y = B
C
      IF (ML .EQ. 0) GO TO 30
      IF (NM1 .LT. 1) GO TO 30
        DO 20 K = 1, NM1
          LM = MIN0(ML,N-K)
          L = IPVT(K)
          T = B(L)
          IF (L .EQ. K) GO TO 10
            B(L) = B(K)
            B(K) = T
 10       CONTINUE
          CALL DAXPY(LM,T,ABD(M+1,K),1,B(K+1),1)
 20     CONTINUE
 30   CONTINUE
C
C     NOW SOLVE  U*X = Y
C
      DO 40 KB = 1, N
        K = N + 1 - KB
        B(K) = B(K)/ABD(M,K)
        LM = MIN0(K,M) - 1
        LA = M - LM
        LB = K - LM
        T = -B(K)
        CALL DAXPY(LM,T,ABD(LA,K),1,B(LB),1)
 40   CONTINUE
    GO TO 100
 50 CONTINUE
C
C     JOB = NONZERO, SOLVE  TRANS(A) * X = B
C     FIRST SOLVE  TRANS(U)*Y = B
C
      DO 60 K = 1, N
        LM = MIN0(K,M) - 1
        LA = M - LM
        LB = K - LM
        T = DDOT(LM,ABD(LA,K),1,B(LB),1)
        B(K) = (B(K) - T)/ABD(M,K)
 60   CONTINUE
C
C     NOW SOLVE TRANS(L)*X = Y
C
    IF (ML .EQ. 0) GO TO 90
    IF (NM1 .LT. 1) GO TO 90
      DO 80 KB = 1, NM1
```

```
          K = N - KB
          LM = MIN0(ML,N-K)
          B(K) = B(K) + DDOT(LM,ABD(M+1,K),1,B(K+1),1)
          L = IPVT(K)
          IF (L .EQ. K) GO TO 70
            T = B(L)
            B(L) = B(K)
            B(K) = T
   70     CONTINUE
   80   CONTINUE
   90  CONTINUE
  100 CONTINUE
      RETURN
      END



            SUBROUTINE DAXPY(N,DA,DX,INCX,DY,INCY)
C
C    CONSTANT TIMES A VECTOR PLUS A VECTOR.
C    USES UNROLLED LOOPS FOR INCREMENTS EQUAL TO ONE.
C    JACK DONGARRA, LINPACK, 3/11/78.
C
      DOUBLE PRECISION DX(1),DY(1),DA
      INTEGER I,INCX,INCY,IX,IY,M,MP1,N
C
      IF(N.LE.0)RETURN
      IF (DA .EQ. 0.0D0) RETURN
      IF(INCX.EQ.1.AND.INCY.EQ.1)GO TO 20
C
C      CODE FOR UNEQUAL INCREMENTS OR EQUAL INCREMENTS
C        NOT EQUAL TO 1
C
      IX = 1
      IY = 1
      IF(INCX.LT.0)IX = (-N+1)*INCX + 1
      IF(INCY.LT.0)IY = (-N+1)*INCY + 1
      DO 10 I = 1,N
        DY(IY) = DY(IY) + DA*DX(IX)
        IX = IX + INCX
        IY = IY + INCY
   10 CONTINUE
      RETURN
C
C      CODE FOR BOTH INCREMENTS EQUAL TO 1
C
C
C      CLEAN-UP LOOP
C
   20 M = MOD(N,4)
      IF( M .EQ. 0 ) GO TO 40
      DO 30 I = 1,M
        DY(I) = DY(I) + DA*DX(I)
   30 CONTINUE
      IF( N .LT. 4 ) RETURN
```

```
   40 MP1 = M + 1
      DO 50 I = MP1,N,4
       DY(I) = DY(I) + DA*DX(I)
       DY(I + 1) = DY(I + 1) + DA*DX(I + 1)
       DY(I + 2) = DY(I + 2) + DA*DX(I + 2)
       DY(I + 3) = DY(I + 3) + DA*DX(I + 3)
   50 CONTINUE
      RETURN
      END
            SUBROUTINE  DSCAL(N,DA,DX,INCX)
C
C     SCALES A VECTOR BY A CONSTANT.
C     USES UNROLLED LOOPS FOR INCREMENT EQUAL TO ONE.
C     JACK DONGARRA, LINPACK, 3/11/78.
C     MODIFIED 3/93 TO RETURN IF INCX .LE. 0.
C
      DOUBLE PRECISION DA,DX(1)
      INTEGER I,INCX,M,MP1,N,NINCX
C
      IF( N.LE.0 .OR. INCX.LE.0 )RETURN
      IF(INCX.EQ.1)GO TO 20
C
C        CODE FOR INCREMENT NOT EQUAL TO 1
C
      NINCX = N*INCX
      DO 10 I = 1,NINCX,INCX
       DX(I) = DA*DX(I)
   10 CONTINUE
      RETURN
C
C        CODE FOR INCREMENT EQUAL TO 1
C
C
C        CLEAN-UP LOOP
C
   20 M = MOD(N,5)
      IF( M .EQ. 0 ) GO TO 40
      DO 30 I = 1,M
       DX(I) = DA*DX(I)
   30 CONTINUE
      IF( N .LT. 5 ) RETURN
   40 MP1 = M + 1
      DO 50 I = MP1,N,5
       DX(I) = DA*DX(I)
       DX(I + 1) = DA*DX(I + 1)
       DX(I + 2) = DA*DX(I + 2)
       DX(I + 3) = DA*DX(I + 3)
       DX(I + 4) = DA*DX(I + 4)
   50 CONTINUE
      RETURN
      END
            INTEGER FUNCTION IDAMAX(N,DX,INCX)
C
C     FINDS THE INDEX OF ELEMENT HAVING MAX. ABSOLUTE VALUE.
C     JACK DONGARRA, LINPACK, 3/11/78.
C     MODIFIED 3/93 TO RETURN IF INCX .LE. 0.
```

```
C
      DOUBLE PRECISION DX(1),DMAX
      INTEGER I,INCX,IX,N
C
      IDAMAX = 0
      IF( N.LT.1 .OR. INCX.LE.0 ) RETURN
      IDAMAX = 1
      IF(N.EQ.1) RETURN
      IF(INCX.EQ.1)GO TO 20
C
C     CODE FOR INCREMENT NOT EQUAL TO 1
C
      IX = 1
      DMAX = DABS(DX(1))
      IX = IX + INCX
      DO I0 I = 2,N
        IF(DABS(DX(IX)).LE.DMAX) GO TO 5
        IDAMAX = I
        DMAX = DABS(DX(IX))
    5   IX = IX + INCX
   10 CONTINUE
      RETURN
C
C     CODE FOR INCREMENT EQUAL TO I
C
   20 DMAX = DABS(DX(1))
      DO 30 I = 2,N
        IF(DABS(DX(I)).LE.DMAX) GO TO 30
        IDAMAX = I
        DMAX = DABS(DX(I))
   30 CONTINUE
      RETURN
      END
            DOUBLE PRECISION FUNCTION DDOT(N,DX,INCX,DY,INCY)
C
C  FORMS THE DOT PRODUCT OF TWO VECTORS.
C  USES UNROLLED LOOPS FOR INCREMENTS EQUAL TO ONE.
C  JACK DONGARRA, LINPACK, 3/11/78.
C
      DOUBLE PRECISION DX(1),DY(1),DTEMP
      INTEGER I,INCX,INCY,IX,IY,M,MP1,N
C
      DDOT = 0.0D0
      DTEMP = 0.0D0
      IF(N.LE.0)RETURN
      IF(INCX.EQ.1.AND.INCY.EQ.1)GO TO 20
C
C     CODE FOR UNEQUAL INCREMENTS OR EQUAL INCREMENTS
C       NOT EQUAL TO 1
C
      IX = 1
      IY = 1
      IF(INCX.LT.0)IX = (-N+1)*INCX + 1
      IF(INCY.LT.0)IY = (-N+1)*INCY + 1
      DO 10 I = 1,N
        DTEMP = DTEMP + DX(IX)*DY(IY)
```

```
      IX = IX + INCX
      IY = IY + INCY
   10 CONTINUE
      DDOT = DTEMP
      RETURN
C
C     CODE FOR BOTH INCREMENTS EQUAL TO 1
C
C
C     CLEAN-UP LOOP
C
   20 M = MOD(N,5)
      IF( M .EQ. 0 ) GO TO 40
      DO 30 I = 1,M
        DTEMP = DTEMP + DX(I)*DY(I)
   30 CONTINUE
      IF( N .LT. 5 ) GO TO 60
   40 MP1 = M + 1
      DO 50 I = MP1,N,5
        DTEMP = DTEMP + DX(I)*DY(I) + DX(I + 1)*DY(I + 1) +
     *   DX(I + 2)*DY(I + 2) + DX(I + 3)*DY(I + 3) + DX(I + 4)*DY(I + 4)
   50 CONTINUE
   60 DDOT = DTEMP
      RETURN
      END
C           ////////////////////////////////////////////////////////////
            SUBROUTINE LEG_TEST2()
            INTEGER I,J,NX,NY,ERROR,H,M1,S,MS,KODE
            REAL*8 A0[ALLOCATABLE](:,:),B0[ALLOCATABLE](:),
     1C0[ALLOCATABLE](:,:),D0[ALLOCATABLE](:,:),X[ALLOCATABLE](:),
     1X0[ALLOCATABLE](:),X00[ALLOCATABLE](:),B00[ALLOCATABLE](:)
            REAL*8 LX,BETA
            INTEGER M,T[ALLOCATABLE](:)
            PRINT *,"PLEASE ENTER NUMBER OF ROWS OR COLUMNS(A)"
            READ(*,*) NX
            M=NX
            NXP=NX
            KODE=4
            ALLOCATE (A0(NXP,NXP),B0(NXP),C0(NXP,NXP),D0(NXP,NXP),
     1T(NXP),X(NXP),X0(NXP),X00(NXP),B00(NXP),STAT=ERROR)
            OPEN(1,FILE='LEG_TEST2.TXT',STATUS='UNKNOWN')
            IF (ERROR.NE.0) STOP
            LX=2.
            BETA=1.
            NY=NX
            CALL GETTIM(H,M1,S,MS)
     CALL SEED(MS)
            DO 100 I=1,NX
                  CALL RANDOM(P)
                  B0(1)=1.
                  B00(I)=P
                  X00(I)=B00(I)
  100       CONTINUE
C
            CALL DRIVED_X(KODE,A0,C0,NX,NXP,LX)
            DO 200 I=1,NX-1
```

```
      DO 200 J=1,NX-1
            CALL RANDOM(P)
            A0(I,J)=A0(I+1,J+1)
            D0(I,J)=A0(I,J)
200   CONTINUE

      CALL LEGS (A0,NX-1,NXP,B00,X,T)
      DO 500 I=1,NX
C           X0(T(I))=X(I)
500   CONTINUE
      CALL ZARB(D0,X,X0,NX,NX,1,NXP,NXP)
      OPEN(2,FILE='LEG_TEST2_1.TXT',STATUS='UNKNOWN')



      DO 10 I=1,NX-1



            WRITE(2,20) (D0(I,J),J=1,NX-1)
20          FORMAT(100(2X,F30.15))

10    CONTINUE
      RETURN
      END SUBROUTINE
C
C     ////////////////////////////////////////////////////////////

      SUBROUTINE LEGS_TEST()
  REAL*8 X(10),B(10),A(10,10)
      INTEGER INDX(10),N,NP
      N=3
      NP=10
      A(1,1)=100.0
      A(1,2)=100.0
      A(1,3)=100.0
      A(2,1)=-100.0
      A(2,2)=300.0
      A(2,3)=-100.0
      A(3,1)=-100.0
      A(3,2)=-100.0
      A(3,3)=300.0
      B(1)=200.0
      B(2)=0.0
      B(3)=0.0
  CALL LEGS (A,N,NP,B,X,INDX)
C
  WRITE (6, 999) (X(I), I=1,N)
  STOP
999 FORMAT (F16.8)
      RETURN
  END SUBROUTINE
C     ////////////////////////////////////////////////////////////
C
      SUBROUTINE ILLEQATIONSOLVER_TEST()
      INTEGER ERROR,I,J,H,M1,S,MS,N,M,NP,MP,INDX[ALLOCATABLE](:,:)
```

```
      REAL*8 A0[ALLOCATABLE](:,:),B0[ALLOCATABLE](:,:),
     1C[ALLOCATABLE](:,:),X[ALLOCATABLE](:,:),
     1P1[ALLOCATABLE](:,:),LX
      N=200
      M=15
      NP=N
      MP=M
      LX=2.
      ALLOCATE (A0(NP,NP),B0(NP,MP),X(NP,MP),INDX(NP,MP),C(NP,NP),
      1P1(NP,MP),STAT=ERROR)
      IF (ERROR.NE.0) STOP
      CALL GETTIM(H,M1,S,MS)
    CALL SEED(MS)
      DO 10 I=1,N-1
            DO 20 J=1,M
                  CALL RANDOM(P)
                  B0(I,J)=P
20          CONTINUE
10    CONTINUE
      CALL DRIVED_X(4,A0,C,N,NP,LX)
      DO 30 I=1,N-1
            DO 40 J=1,N-1
                  A0(I,J)=A0(I+1,J+1)
40          CONTINUE
30    CONTINUE
      CALL ILLEQATIONSOLVER(A0,N-1,NP,B0,X,M,MP,INDX)
      OPEN(1,FILE='ILLEQATIONSOLVER_TEST.TXT',STATUS='UNKNOWN')
      CALL ZARB(A0,X,P1,N-1,N-1,M,NP,NP)
      DO 50 I=1,N-1
                  WRITE(1,1001) (P1(I,J)-B0(I,J),J=1,M)
1001              FORMAT(500(2X,F15.6))

50    CONTINUE

      CLOSE(1)
      RETURN
      END SUBROUTINE
C     ////////////////////////////////////////////////////////////
      SUBROUTINE ILLEQATIONSOLVER(A,N,NP,B,X,M,MP,INDX)
      INTEGER N,NP,M,MP,ERROR,I,J,K,INDX(NP,1)
      REAL*8 A(NP,NP),B(NP,MP),X(NP,MP)
      REAL*8 A0[ALLOCATABLE](:,:),B0[ALLOCATABLE](:,:)
      ALLOCATE (A0(NP,NP),B0(NP,MP),STAT=ERROR)
      IF (ERROR.NE.0) STOP
      DO 10 I=1,N
            DO 20 J=1,N
                  A0(I,J)=A(I,J)
20          CONTINUE
            DO 30 J=1,M
                  B0(I,J)=B(I,J)
30          CONTINUE
10    CONTINUE
      DO 40 J=1,M
            CALL LEGS(A0,N,NP,B0(1,J),X(1,J),INDX(1,J))
            DO    40 K=1,N
```

```
                    DO 40 I=1,N
                        A0(K,I)=A(K,I)
40          CONTINUE
            RETURN
            END SUBROUTINE
C           ////////////////////////////////////////////////////////
      SUBROUTINE LEGS(A,N,NP,B,X,INDX)
C
C SUBROUTINE TO SOLVE THE EQUATION A(N,N)*X(N) = B(N) WITH THE
C PARTIAL-PIVOTING GAUSSIAN ELIMINATION SCHEME.
C
            INTEGER N,NP
      REAL*8  A(NP,NP),B(1),X(1)
            INTEGER INDX(1)
C
      CALL ELGS(A,N,NP,INDX)
C
      DO    100 I = 1, N-1
       DO   90 J = I+1, N
         B(INDX(J)) = B(INDX(J))
     *             -A(INDX(J),I)*B(INDX(I))
   90  CONTINUE
  100 CONTINUE
C
      X(N) = B(INDX(N))/A(INDX(N),N)
      DO    200 I = N-1, 1, -1
       X(I) = B(INDX(I))
       DO   190 J = I+1, N
        X(I) = X(I)-A(INDX(I),J)*X(J)
  190  CONTINUE
       X(I) =  X(I)/A(INDX(I),I)
  200 CONTINUE
C
      RETURN
      END
C
C           ////////////////////////////////////////////////////////
      SUBROUTINE ELGS(A,N,NP,INDX)
C
C SUBROUTINE TO PERFORM THE PARTIAL-PIVOTING GAUSSIAN ELIMINATION.
C A(N,N) IS THE ORIGINAL MATRIX IN THE INPUT AND TRANSFORMED
C MATRIX PLUS THE PIVOTING ELEMENT RATIOS BELOW THE DIAGONAL IN
C THE OUTPUT.  INDX(N) RECORDS THE PIVOTING ORDER.

C
            INTEGER N,NP
     REAL*8 A(NP,NP),C[ALLOCATABLE](:) ,C1
            INTEGER INDX(1),ERROR
            ALLOCATE (C(NP),STAT=ERROR)
            IF (ERROR.NE.0) STOP
C INITIALIZE THE INDEX
C
     DO     50   I = 1, N
      INDX(I) = I
   50 CONTINUE
```

```
C
C FIND THE RESCALING FACTORS, ONE FROM EACH ROW
C
      DO    100  I = 1, N
       C1= 0.0
       DO   90  J = 1, N
        C1 = DMAX1(C1,ABS(A(I,J)))
   90   CONTINUE
       C(I) = C1
  100  CONTINUE
C
C SEARCH THE PIVOTING (LARGEST) ELEMENT FROM EACH COLUMN
C
     DO    200  J = 1, N-1
      PI1 = 0.0
      DO   150  I = J, N
       PI = ABS(A(INDX(I),J))/C(INDX(I))
       IF (PI.GT.PI1) THEN
        PI1 = PI
        K  = I
       ELSE
       ENDIF
  150  CONTINUE
C
C INTERCHANGE THE ROWS VIA INDX(N) TO RECORD PIVOTING ORDER
C
      ITMP   = INDX(J)
      INDX(J) = INDX(K)
      INDX(K) = ITMP
      DO   170  I = J+1, N
       PJ  = A(INDX(I),J)/A(INDX(J),J)
C
C RECORD PIVOTING RATIOS BELOW THE DIAGONAL
C
       A(INDX(I),J) = PJ
C
C MODIFY OTHER ELEMENTS ACCORDINGLY
C
       DO 160  K = J+1, N
        A(INDX(I),K) = A(INDX(I),K)-PJ*A(INDX(J),K)
  160   CONTINUE
  170  CONTINUE
  200 CONTINUE
C
     RETURN
     END

C          ********************************************************************
C          ********************************************************************
C          ************************(DRIVATE'S COMPUTE)*************************
C          ********************************************************************
C          ********************************************************************

C          ////////////////////////////////////////////////////////////////////
C          THIS SUBROTINE COMPUTE FIRST AND SECOND ORDER OF DRIVATE
```

```
C       AT X DIRECTION
C       D1 --> REAL*8 INPUT MATRIX(NP,NP) .ON RETURN FIRST DRIVATE MATRIX
C       D2 --> REAL*8 INPUT MATRIX(NP,NP).ON RETURN SECOND DRIVATE MATRIX
C       N  --> INTEGER FIRST DIMENSION OF D1 & D2
C       NP --> INTEGER THE ORDER OF D1 & D2
C       LX --> REAL*8 LONG OF SPACE AT X DIRECTION

        SUBROUTINE DRIVED_X(KODE,D1,D2,N,NP,LX)
        INTEGER N,NP,ML,MU,LDA,ERROR,I,J,KODE
        REAL*8 D1(NP,1),D2(NP,1),LX,DX,AL
        REAL*8 A1[ALLOCATABLE](:,:),B1[ALLOCATABLE](:,:),
     1C1[ALLOCATABLE](:,:)
        ALLOCATE (A1(NP,NP),B1(NP,NP),C1(NP,NP),STAT = ERROR)
        IF (ERROR.NE.0) STOP

                AL=1./3.
                ML=1
                MU=ML
                LDA=2*ML+MU+1
                DX=LX/(N-1)
                D1(1:N,1:N)=0._8
                D2(1:N,1:N)=0._8
                A1(1:N,1:N)=0._8
                B1(1:N,1:N)=0._8
                C1(1:N,1:N)=0._8
                CALL PADEBUELL1(KODE,AL,DX,N,NP,A1,B1)
                CALL INVERSE(A1,C1,N,NP,ML,MU,LDA)
                CALL ZARB(C1,B1,D1,N,N,N,NP,NP)
                DO 80 I=1,NP
                        DO 80 J=1,NP
                                A1(I,J)=0.
                                B1(I,J)=0.
                                C1(I,J)=0.
                                D2(I,J)=0.
80                      CONTINUE
                AL=1./4.
                CALL PADEBUELL2(KODE,AL,DX,N,NP,A1,B1)
                CALL INVERSE(A1,C1,N,NP,ML,MU,LDA)
                CALL ZARB(C1,B1,D2,N,N,N,NP,NP)
                DEALLOCATE (A1,B1,C1, STAT = ERROR)
                IF (ERROR.NE.0) STOP
        END SUBROUTINE



C       ////////////////////////////////////////////////////////////
C       ////////////////////////////////////////////////////////////
C       THIS SUBROTINE COMPUTE FIRST AND SECOND ORDER OF DRIVATE
C       AT Y DIRECTION (MAPPED CONDITION)
C       D1 --> REAL*8 INPUT MATRIX(NP,NP) .ON RETURN FIRST DRIVATE MATRIX
C       D2 --> REAL*8 INPUT MATRIX(NP,NP).ON RETURN SECOND DRIVATE MATRIX
C       N  --> INTEGER FIRST DIMENSION OF D1 & D2
C       NP --> INTEGER THE ORDER OF D1 & D2
C       BETA --> REAL*8 STERTCHED PARAMETER IF BETA->INFINITE=>SPACE IS EQUAL
```

```
C       LY --> REAL*8 LONG OF SPACE AT Y DIRECTION


        SUBROUTINE DRIVED_Y(KODE,D1,D2,N,NP,BETA)
        INTEGER N,NP,LDA,ML,MU,ERROR,KODE
        REAL*8 D1(NP,1),D2(NP,1),AL,BETA,DETA,PI
        REAL*8 A1[ALLOCATABLE](:,:),B1[ALLOCATABLE](:,:),
     1C1[ALLOCATABLE](:,:),LAND1[ALLOCATABLE](:,:),
     1LAND2[ALLOCATABLE](:,:),LAND3[ALLOCATABLE](:,:),
        1JA1[ALLOCATABLE](:,:),JA2[ALLOCATABLE](:,:),A2[ALLOCATABLE](:,:),
        1B2[ALLOCATABLE](:,:)

        ALLOCATE (A1(NP,NP),B1(NP,NP),C1(NP,NP),LAND1(NP,NP),JA1(NP,NP),
        1JA2(NP,NP),LAND2(NP,NP),LAND3(NP,NP),A2(NP,NP),B2(NP,NP),
     1STAT = ERROR)
        IF (ERROR.NE.0) STOP
                AL=1./3.0
                ML=1
                MU=ML
                LDA=2*ML+MU+1
                PI=4.*ATAN(1.0)
                DETA=1./(N-1)
                        DO 20 I=1,NP
                        DO 20 J=1,NP
                                LAND1(I,J)=0.
                                LAND2(I,J)=0.
                                LAND3(I,J)=0.
                                C1(I,J)=0.
                                A1(I,J)=0.
                                B1(I,J)=0.
                                A2(I,J)=0.
                                B2(I,J)=0.
20              CONTINUE

                CALL PADEBUELL1(KODE,AL,DETA,N,NP,A1,B1)
                CALL INVERSE(A1,C1,N,NP,ML,MU,LDA)
                CALL MAPING_1(LAND1,LAND2,LAND3,BETA,N,NP)
                CALL ZARB(LAND1,C1,JA1,N,N,N,NP,NP)
                CALL ZARB(JA1,B1,D1,N,N,N,NP,NP)
                DO 10 I=1,NP
                        DO 10 J=1,NP
                                A1(I,J)=0.
                                JA1(I,J)=0.
10              CONTINUE
                AL=1./4.0
                CALL PADEBUELL2(KODE,AL,DETA,N,NP,A2,B2)
                CALL INVERSE(A2,A1,N,NP,ML,MU,LDA)
                CALL ZARB(LAND2,A1,JA2,N,N,N,NP,NP)
                CALL ZARB(JA2,B2,JA1,N,N,N,NP,NP)
                CALL ZARB(LAND3,C1,A2,N,N,N,NP,NP)
                CALL ZARB(A2,B1,JA2,N,N,N,NP,NP)
                CALL SUM(JA1,JA2,D2,N,N,NP)
                DEALLOCATE (A1,B1,C1,LAND1,JA1,JA2,LAND2,LAND3,A2,B2
     1  ,STAT = ERROR)
```

```
            IF (ERROR.NE.0) STOP
            RETURN
      END
C     ////////////////////////////////////////////////////////
C     ////////////////////////////////////////////////////////
C
C
C     ////////////////////////////////////////////////////////

            SUBROUTINE PADEBUELL1(KODE,ALPHA,DX,N,NP,A,B)
            INTEGER KODE,N,NP,I
            REAL*8 A(NP,1),B(NP,1),ALPHA,DX,Z,ALPHA1
            Z=1.0/ALPHA
            DO 10 I=4,N-3
                  A(I,I)=Z
                  A(I,I+1)=1.
                  A(I,I-1)=A(I,I+1)
                  B(I,I+2)=(4.-Z)/(12.*DX)
                  B(I,I-2)=-B(I,I+2)
                  B(I,I+1)=(1.+2.*Z)/(3.*DX)
                  B(I,I-1)=-B(I,I+1)
10          CONTINUE
            I=3
            ALPHA1=(16.+32.*Z)/(40.-Z)
C           ALPHA1=1./ALPHA1
            A(I,I)=ALPHA1
            A(I,I+1)=1.
            A(I,I-1)=A(I,I+1)
            B(I,I+2)=(4.-ALPHA1)/(12.*DX)
            B(I,I-2)=-B(I,I+2)
            B(I,I+1)=(1.+2.*ALPHA1)/(3.*DX)
            B(I,I-1)=-B(I,I+1)
            I=N-2
            ALPHA1=(16.+32.*Z)/(40.-Z)
C           ALPHA1=1./ALPHA1
            A(I,I)=ALPHA1
            A(I,I+1)=1.
            A(I,I-1)=A(I,I+1)
            B(I,I+2)=(4.-ALPHA1)/(12.*DX);
            B(I,I-2)=-B(I,I+2);
            B(I,I+1)=(1.+2.*ALPHA1)/(3.*DX);
            B(I,I-1)=-B(I,I+1);

            I=2
            ALPHA1=4.
            A(I,I)=ALPHA1
            A(I,I+1)=1.
            A(I,I-1)=A(I,I+1)
            B(I,I+1)=(1.+2.*ALPHA1)/(3.*DX);
            B(I,I-1)=-B(I,I+1);

            I=N-1
            A(I,I)=ALPHA1
            A(I,I+1)=1.
            A(I,I-1)=A(I,I+1)
```

```
      B(I,I+1)=(1.+2.*ALPHA1)/(3.*DX)
      B(I,I-1)=-B(I,I+1)

      IF(KODE.EQ.0) THEN
            A(1,1)=1.
            A(N,N)=1.
            A(N,N-1)=2.
            B(1,1)=1.
            B(N,N)=2.5/DX
            B(N,N-1)=-2./DX
            B(N,N-2)=-0.5/DX

      ELSE IF(KODE.EQ.1) THEN
            A(1,1)=1.
            A(1,2)=2.
            B(1,1)=-2.5/DX
            B(1,2)=2./DX
            B(1,3)=0.5/DX
            A(N,N)=1.
            B(N,N)=1.
      ELSE IF  (KODE.EQ.2) THEN
            A(1,1)=1.
            B(1,1)=1.
            A(N,N)=1.
            B(N,N)=1.
      ELSE
            A(1,1)=1.
            A(N,N)=A(1,1)
            A(1,2)=2.
            A(N,N-1)=A(1,2)
            B(1,1)=-2.5/DX
            B(1,2)=2./DX
            B(1,3)=0.5/DX
            B(N,N)=-1.0*B(1,1)
            B(N,N-1)=-1.0*B(1,2)
            B(N,N-2)=-1.0*B(1,3)
      END IF
      RETURN
      END SUBROUTINE
C     ////////////////////////////////////////////////////////////
C
C     ////////////////////////////////////////////////////////////
      SUBROUTINE PADEBUELL2(KODE,ALPHA,DX,N,NP,A,B)

C     PREPARE A & B FOR 2ND DERIVATIVE MATRICES OF EQ 14 OF BUELL
C     A(NP,NP),B(NP,NP) ARE SQUARE MATRICES IN AF"= BF
C     DX --> SPATIAL INCREMENT IN X DIRECTION
C     GAMMA --> NUMBER IN EQ 14
C     KODE  --> IF ONE ==>PREPARE MATRICES AT BOUNARIES AS SUGGESTED BY 17
C           OTHERWISE AS 16
C
C
      INTEGER KODE,N,NP,J
      REAL*8 A(NP,1),B(NP,1),ALPHA,DX,Z
      Z=1.0/ALPHA
```

```
      DO 10 J=3,N-2
            A(J,J)=Z
            A(J,J+1)=1.
            A(J,J-1)=A(J,J+1)
            B(J,J+2)=(10.-Z)/(12.*DX*DX);
            B(J,J-2)=B(J,J+2)
            B(J,J+1)=4.*(Z-1.)/β .*DX*DX)
            B(J,J-1)=B(J,J+1)
            B(J,J)=2 .*(B(J,J+2)+B(J,J+1))
10    CONTINUE
      A(N-1,N-1)=10.
      A(2,2)=A(N-1,N-1)
      J=2
      Z=10.
      A(J,J+1)=1.
      A(J,J-1)=A(J,J+1)
      B(J,J+1)=4.*(Z-1.)/β .*DX*DX);
      B(J,J-1)=B(J,J+1)
      B(J,J)=2 .*B(J,J+1)

      J=N-1
      A(J,J+1)=1.
      A(J,J-1)=A(J,J+1)
      B(J,J+1)=4.*(Z-1.)/β .*DX*DX);
      B(J,J-1)=B(J,J+1)
      B(J,J)=2 .*B(J,J+1)

      IF    (KODE.EQ.0) THEN    ! TO USE DERIVATIVES AT BOUNDARIES
            A(1,1)=1.
            A(1,2)=2.
            A(N,N)=1.
            A(N,N-1)=11.
            B(1,1)=-1.5/(DX*DX)
            B(1,3)=1.5/(DX*DX)
            B(N,N)=13./(DX*DX)
            B(N,N-1)=-27./(DX*DX)
            B(N,N-2)=15./(DX*DX)
            B(N,N-3)=-1./(DX*DX)
      ELSE IF (KODE.EQ.1) THEN
            A(1,1)=1.
            A(1,2)=11.
            B(1,1)=13./(DX*DX)
            B(1,2)=-27./(DX*DX)
            B(1,3)=15./(DX*DX)
            B(1,4)=-1./(DX*DX)
            A(N,N)=1.
            A(N,N-1)=2.
            B(N,N)=-1.5/(DX*DX)
            B(N,N-2)=1.5/(DX*DX)
      ELSE IF (KODE.EQ.2) THEN
            A(1,1)=1.
            A(1,2)=2.
            B(1,1)=-1.5/(DX*DX)
            B(1,3)=1.5/(DX*DX)
            A(N,N)=1.
```

```
                    A(N,N-1)=2.
                    B(N,N)=-1.5/(DX*DX)
                    B(N,N-2)=1.5/(DX*DX)
                ELSE
                  A(1,1)=1.
                    A(N,N)=A(1,1)
                    A(1,2)=11.
                    A(N,N-1)=A(1,2)
                    B(N,N)=13./(DX*DX)
                    B(1,1)=B(N,N)
                    B(N,N-1)=-27./(DX*DX)
                    B(1,2)=B(N,N-1)
                    B(N,N-2)=15./(DX*DX)
                    B(1,3)=B(N,N-2)
                    B(N,N-3)=-1./(DX*DX)
                    B(1,4)=B(N,N-3)
                END IF
                END SUBROUTINE
C       ////////////////////////////////////////////////////////////
C
C
C       ////////////////////////////////////////////////////////////
C       THIS SUBROUTINE COMUTE MATRIX LANDA1,LANDA2,LANDA3 IN نگاشت
C       METHOD .
C       LAND1 -->REAL*8 INPUT MATRIX(NP,NP) LANDA1
C       LAND2 -->REAL*8 INPUT MATRIX(NP,NP) LANDA2
C       LAND3 -->REAL*8 INPUT MATRIX(NP,NP) LANDA3
C       N --> INTEGER FIRST DIMENSION OF LAND1 & LAND2,LAND3
C       NP --> INTEGER THE ORDER OF LAND1 & LAND2,LAND3

        SUBROUTINE MAPING_1(LAND1,LAND2,LAND3,BETA,N,NP)
                INTEGER N,NP,I
                REAL*8 LAND1(NP,1),LAND2(NP,1),LAND3(NP,1),BETA,ETA,Y,DETA
                REAL*8 PI
                PI=4.*DATAN(1._8)
                DETA=1./(N-1)
                DO 10 I=1,N
                        ETA=(I-1)*DETA
                        Y=-BETA/DTAN(PI*ETA)
                        LAND1(I,I)=(DSIN(PI*ETA))**2./(PI*BETA)
                        LAND2(I,I)=((DSIN(PI*ETA))**2./(PI*BETA))**2.
                        LAND3(I,I)=2.*(DSIN(PI*ETA))**3.*DCOS(PI*ETA)/(PI*BETA**2.)
10              CONTINUE
        RETURN
        END SUBROUTINE


C       ////////////////////////////////////////////////////////////
C       ************************************************************
C       ************************************************************
C       ************************(MATRIX OPERATORES)*******************
C       ************************************************************
C       ************************************************************
C
```

```
C       ////////////////////////////////////////////////////////
        SUBROUTINE SUM(A,B,C,M,N,MA)
  REAL*8 A(MA,1),B(MA,1),C(MA,1)
  DO 10 I=1,M
        DO 10 J=1,N
10        C(I,J)=A(I,J)+B(I,J)
  END
C       ////////////////////////////////////////////////////////
        SUBROUTINE MINUS(A,B,C,M,N,MA)
  REAL*8 A(MA,1),B(MA,1),C(MA,1)
  DO 10 I=1,M
        DO 10 J=1,N
10        C(I,J)=A(I,J)-B(I,J)
  END

C       ////////////////////////////////////////////////////////
C       ////////////////////////////////////////////////////////
        SUBROUTINE MTRANSE_TEST()
        REAL*8 A(10,10),B(10,10)
        INTEGER I,J
        DO 10 I=1,3
                READ(*,*) (A(I,J),J=1,2)
10      CONTINUE
        CALL MTRANSPOSE(A,B,2,2,10,10)
        DO 20 I=1,2

                        WRITE(*,30) (B(I,J),J=1,3)
30                      FORMAT(3(2X,F8.3))

20      CONTINUE
        END SUBROUTINE

C
C       ////////////////////////////////////////////////////////
C
C       ZARB MULTIPLY A & B (A*B)
C       A --->FISRT MATRIX
C       B --->SECOND MATRIX
C       C --->(A*B)
C       M --->NUMBER OF ROWS OF FIRST MATRIX(A)
C       N --->NUMBER OF COLUMNS OF FIRST MATRIX(A)
C       L --->NUMBER OF COLUMNS OF SECOND MATRIX(B)
C       MA--->ORDER OF MATRIX(A)
C       NB--->ORDER OF MATRIX(B)
C
C
C
        SUBROUTINE ZARB(A,B,C,M,N,L,MA,NB)
  REAL*8 A(MA,1),B(NB,1),C(MA,1)
  DO 30 I=1,M
        DO 20 J=1,L
          C(I,J)=0.
                DO 10 K=1,N
                        C(I,J)=C(I,J)+A(I,K)*B(K,J)
```

```
10                CONTINUE
20          CONTINUE
30          CONTINUE
    END

C           ////////////////////////////////////////////////////////
C           ////////////////////////////////////////////////////////
            SUBROUTINE E_ZARB_E_TEST()
            REAL*8 A(20,20),B(20,20),C(20,20)
            A(1,1)=1.
            A(1,2)=2.
            A(2,1)=3.
            A(2,2)=4.
            B(1,1)=3.
            B(1,2)=2.
            B(2,1)=1.
            B(2,2)=0.
            CALL E_ZARB_E(A,B,C,2,2,20)
            PRINT     *,C(1,1),C(1,2)
            PRINT     *,C(2,1),C(2,2)
            RETURN
            END SUBROUTINE
C           ////////////////////////////////////////////////////////
C
C           ////////////////////////////////////////////////////////
C
C           E_ZARB_E MULTIPLY A & B (A.*B)[MULTIPLY ELEMENT BY ELEMENT]
C           A --->FISRT MATRIX
C           B --->SECOND MATRIX
C           C --->(A.*B)
C           M --->NUMBER OF ROWS OF FIRST MATRIX(A)
C           N --->NUMBER OF COLUMNS OF FIRST MATRIX(A)
C           MA--->ORDER OF MATRIX(A)
C
C
            SUBROUTINE E_ZARB_E(A,B,C,M,N,MA)
    REAL*8 A(MA,1),B(MA,1),C(MA,1)
    DO 30 I=1,M
                    DO 20 J=1,N
                            C(I,J)=A(I,J)*B(I,J)
20                  CONTINUE
30          CONTINUE
    END

C           ////////////////////////////////////////////////////////
C
            SUBROUTINE ZEROS(A,N,M,NP)
                    INTEGER NP,N,M,I,J
                    REAL*8 A(NP,1)
                    DO 10 I=1,N
                            DO 20 J=1,M
                                    A(I,J)=0.
20                          CONTINUE
10          CONTINUE
            RETURN
```

```
      END SUBROUTINE
C     ////////////////////////////////////////////////////////
C     ////////////////////////////////////////////////////////
      SUBROUTINE GARINE(A,B,M,N,NP)
      INTEGER M,N,I,J
      REAL*8 A(NP,1),B(NP,1)
      DO 10 I=1,M
            DO 10 J=1,N
                  B(I,J)=-1.*A(I,J)
10    CONTINUE
      RETURN
      END SUBROUTINE
C     ////////////////////////////////////////////////////////
C     SUBROUTINE TRANSPOSE IS A COMPAQ VISUAL FORTRAN FOR TRANSPOSE A
C     MATRIX NXN ,TRANSPOSE OF MATRIX OVERWRITE ON MATRIX A,ON RETURN.
C     ////////////////////////////////////////////////////////
      SUBROUTINE TRANSPOSE(A,N,NP)
      INTEGER N,I,J
      REAL*8 A(NP,1),TEMP
      DO 10 I=1,N
            DO 20 J=1,I-1
                  TEMP=A(I,J)
                  A(I,J)=A(J,I)
                  A(J,I)=TEMP
20          CONTINUE
10    CONTINUE
      RETURN
      END SUBROUTINE
C     SUBROUTINE MTRANSPOSE IS A COMPAQ VISUAL FORTRAN FOR TRANSPOSE A
C     MATRIX NXN ,TRANSPOSE OF MATRIX OVERWRITE ON MATRIX A,ON RETURN.
C     ////////////////////////////////////////////////////////
      SUBROUTINE MTRANSPOSE(A,B,NX,NY,NXP,NYP)
      INTEGER NX,NY,I,J
      REAL*8 A(NXP,1),B(NYP,1)
      DO 30 I=1,NYP
            DO 30 J=1,NXP
                  B(I,J)=0.
30    CONTINUE
      DO 10 I=1,NX
            DO 20 J=1,NY
                  B(J,I)=A(I,J)
20          CONTINUE
10    CONTINUE
      RETURN
      END SUBROUTINE
C     ////////////////////////////////////////////////////////
      SUBROUTINE F(X,VALUE)
                  VALUE=SIN(X)
                  RETURN
            END SUBROUTINE
            SUBROUTINE FPRIM(X,VALUE)
                  VALUE=COS(X)
                  RETURN
            END SUBROUTINE
```

```
C       ////////////////////////////////////////////////////////////
C
C       ***************************************************************
C       ***************************************************************
C       *************************(SOLVER FOR AX+XB=C)********************
C       ***************************************************************
C       ***************************************************************
C       ////////////////////////////////////////////////////////////
        SUBROUTINE POISSON_TEST()
        INTEGER M1,N1,M11
        INTEGER NA1,NU1,FAIL1,NC1,NV1,NB1
      INTEGER I,J,ERROR
      INTEGER H,M,S,MS
        REAL*8 EA1,EB1,SUM1
        REAL*8 A1[ALLOCATABLE](:,:),B1[ALLOCATABLE](:,:),
     1C1[ALLOCATABLE](:,:),X1[ALLOCATABLE](:,:),T1[ALLOCATABLE](:,:),
     1T2[ALLOCATABLE](:,:),X0[ALLOCATABLE](:,:)
      REAL*8 U1[ALLOCATABLE](:,:),V1[ALLOCATABLE](:,:)
        CALL GETTIM(H,M,S,MS)
      CALL SEED(MS)
        PRINT *,"PLEASE ENTER NUMBER OF ROWS(MARTX X):"
        READ *,M1
        PRINT *,"PLEASE ENTER NUMBER OF COLUMNS(MARTX X):"
        READ *,N1
        NA1=M1
      NU1=M1
      NB1=N1
      NV1=N1
      NC1=M1
      ALLOCATE (A1(NA1,NA1),B1(NB1,NB1),X1(NA1,NB1),X0(NA1,NB1),
     1 C1(NA1,NB1),U1(NA1,NA1),V1(NB1,NB1),T1(NA1,NB1),
     1 T2(NA1,NB1),STAT = ERROR)
      IF (ERROR.NE.0) STOP

      DO 10 I=1,M1
                    DO 10 J=1,M1
                    CALL RANDOM(P)
                    A1(I,J)=P*10+5
10    CONTINUE
      DO 13 I=1,N1
                    DO 13 J=1,N1

      CALL RANDOM(P)
      B1(I,J)=P*10+5.
13    CONTINUE
        DO 14 I=1,M1
        DO 14 J=1,N1
                CALL RANDOM(P)
                X1(I,J)=P*10+5.
14      CONTINUE
        CALL ZARB(A1,X1,T1,M1,M1,N1,NA1,NA1)
     CALL ZARB(X1,B1,T2,M1,N1,N1,NA1,NB1)
     CALL SUM(T1,T2,C1,M1,N1,NA1)
     OPEN(1,FILE='POISSON_TEST.TXT',STATUS='UNKNOWN')
     WRITE(1,201)
```

```
201  FORMAT('THIS PROGRAM WAS DEVELOPED BY M.H.DIBAEE TO SOLVE')
     WRITE(1,202)
202  FORMAT('THE MATRIX EQUATION  AX+XB=C  WHERE A & B & C ARE')
     WRITE(1,203)
203  FORMAT('REAL MATRIX')
     WRITE(1,20)
20   FORMAT('-------------------------------------------------------')
          CALL POISSON(A1,B1,C1,M1,N1,NA1,NB1,FAIL1)
          WRITE(1,47) FAIL1
47        FORMAT(I2)
     DO 21 I=1,M1
          WRITE(1,22) (ABS((X1(I,J)-C1(I,J))/X1(I,J)),J=1,N1)
22        FORMAT( 1000(2X,F15.8))
21   CONTINUE
     CLOSE(1)
          RETURN
     END SUBROUTINE
C          ----------------------------------------
C          ////////////////////////////////////////////////////////////
C
          SUBROUTINE POISSON(A,B,C,M,N,MP,NP,FAIL)
          INTEGER M,N,MP,NP,FAIL,ERROR,I,J,M11
          INTEGER NA1,NU1,FAIL1,NC1,NV1,NB1
          REAL*8 A(MP,1),B(NP,1),C(MP,1),EA1,EB1
          REAL*8 U[ALLOCATABLE](:,:),V[ALLOCATABLE](:,:)
          REAL*8 A1[ALLOCATABLE](:,:),B1[ALLOCATABLE](:,:)
          IF (M.GT.N) THEN
                    M11=M+1
          ELSE
                    M11=N+1
          END IF
          NA1=M11
     NU1=M11
     NB1=M11
     NV1=M11
     NC1=M11
          EA1=10E-18
     EB1=EA1
          ALLOCATE (A1(NA1,NA1),B1(NB1,NB1),
    1U(NA1,NA1),V(NB1,NB1),STAT = ERROR)
     IF (ERROR.NE.0) STOP
          DO 10 I=1,M
                    DO 20 J=1,M
                              A1(I,J)=A(I,J)
20                  CONTINUE
10        CONTINUE
          DO 40 I=1,N
                    DO 50 J=1,N
                              B1(I,J)=B(I,J)
50                  CONTINUE
40        CONTINUE

          CALL AXPXB(A1,U,M,NA1,NU1,B1,V,N,NB1,NV1,C,MP,EA1,EB1,FAIL)
          DEALLOCATE (U,V,A1,B1, STAT = ERROR)
          IF (ERROR.NE.0) STOP
```

```
      RETURN
      END SUBROUTINE
C
C     ////////////////////////////////////////////////////
      SUBROUTINE AXPXB_TEST()
      INTEGER M1,N1,M11
      INTEGER NA1,NU1,FAIL1,NC1,NV1,NB1
  INTEGER I,J,ERROR
  INTEGER H,M,S,MS
      REAL*8 EA1,EB1,SUM1
      REAL*8 A1[ALLOCATABLE](:,:),B1[ALLOCATABLE](:,:),
 1C1[ALLOCATABLE](:,:),X1[ALLOCATABLE](:,:),T1[ALLOCATABLE](:,:),
 1T2[ALLOCATABLE](:,:)
  REAL*8 U1[ALLOCATABLE](:,:),V1[ALLOCATABLE](:,:)
      CALL GETTIM(H,M,S,MS)
  CALL SEED(MS)
      PRINT *,"PLEASE ENTER NUMBER OF ROWS(MARTX X):"
      READ *,M1
      PRINT *,"PLEASE ENTER NUMBER OF COLUMNS(MARTX X):"
      READ *,N1
      IF (M1.GT.N1) THEN
            M11=M1+1
      ELSE
            M11=N1+1
      END IF
      SUM1=0.
      NA1=M11
  NU1=M11
  NB1=M11
  NV1=M11
  NC1=M11
  ALLOCATE (A1(NA1,NA1),B1(NB1,NB1),X1(NA1,NB1),
 1 C1(NA1,NB1),U1(NA1,NA1),V1(NB1,NB1),T1(NA1,NB1),
 1 T2(NA1,NB1),STAT = ERROR)
  IF (ERROR.NE.0) STOP
  DO 10 I=1,M1
            DO 10 J=1,M1
            CALL RANDOM(P)
            A1(I,J)=P*10+5
10  CONTINUE
  DO 13 I=1,N1
            DO 13 J=1,N1

  CALL RANDOM(P)
  B1(I,J)=P*10+5.
13  CONTINUE
      DO 14 I=1,M1
      DO 14 J=1,N1
            CALL RANDOM(P)
            X1(I,J)=P*10+5.
14    CONTINUE
      CALL ZARB(A1,X1,T1,M1,M1,N1,NA1,NA1)
  CALL ZARB(X1,B1,T2,M1,N1,N1,NA1,NB1)
  CALL SUM(T1,T2,C1,M1,N1,NA1)
  OPEN(1,FILE='AXPXB_TEST.TXT',STATUS='UNKNOWN')
```

```
      WRITE(1,201)
  201 FORMAT('THIS PROGRAM WAS DEVELOPED BY M.H.DIBAEE TO SOLVE')
      WRITE(1,202)
  202 FORMAT('THE MATRIX EQUATION  AX+XB=C  WHERE A & B & C ARE')
      WRITE(1,203)
  203 FORMAT('REAL MATRIX')
      WRITE(1,20)
   20  FORMAT('------------------------------------------------------------')
      DO 24 I=1,M1
            DO 24 J=1,M1
            WRITE(1,23) I,J,A1(I,J)
   23       FORMAT(' A('I2','I2')='F8.4)
   24  CONTINUE
      DO 26 I=1,N1
            DO 26 J=1,NI
            WRITE(1,25) I,J,B1(I,J)
   25       FORMAT(' B('I2','I2')='F8.4)
   26  CONTINUE
      DO 36 I=1,M1
            DO 36 J=1,N1
            WRITE(1,35) I,J,X1(I,J)
   35       FORMAT(' X0('I2','I2')='F8.4)
   36  CONTINUE
      DO 46 I=1,M1
            DO 46 J=1,N1
            WRITE(1,45) I,J,C1(I,J)
   45       FORMAT(' C1('I3','I3')='F10.4)
   46  CONTINUE
      EA1=0.000000001
      EB1=0.000000001
            CALL AXPXB(A1,U1,M1,NA1,NU1,B1,V1,N1,NB1,NV1,C1,NC1,EA1,EB1,FAIL1)
            WRITE(1,47) FAIL1
   47       FORMAT(I2)
      DO 21 I=1,M1
            DO 21 J=1,N1
            SUM1=SUM1+(C1(I,J)-X1(I,J))**2
            WRITE(1,22) I,J,(C1(I,J)),ABS((C1(I,J)-X1(I,J))/X1(I,J))
   22       FORMAT(' X('I2','I2')='F10.4'  ' F15.6)
   21  CONTINUE
            WRITE(1,28) SQRT(SUM1)/((N1*M1))
   28       FORMAT('ERROR(RMS)='F10.5)
      CLOSE(1)
            RETURN
      END SUBROUTINE
C           ------------------------------------------
C           ------------------------------------------
      SUBROUTINE AXPXB(A,U,M,NA,NU,B,V,N,NB,NV,C,NC,EPSA,
     1EPSB,FAIL)
C
C AXPXB IS A FORTRAN IV SUBROUTINE TO SOLVE THE REAL MATRIX
C EQUATION AX + XB = C.  THE MATRICES A AND B ARE TRANS-
C FORMED INTO REAL SCHUR FORM, AND THE TRANSFORMED SYSTEM IS
C SOLVED BY BACK SUBSTITUTION.  THE PROGRAM REQUIRES THE
C AUXILIARY SUBROUTINES HSHLDR, BCKMLT, SCHUR, AND SHRSLV.
C THE PARAMETERS IN THE ARGUMENT LIST ARE
C
```

```
C       A     A DOUBLY SUBSCRIPTED ARRAY CONTAINING THE
C             MATRIX A.  ON RETURN, THE LOWER TRIANGLE
C             AND SUPERDIAGONAL OF THE ARRAY A CONTAIN
C             A LOWER REAL SCHUR FORM OF A.  THE ARRAY
C             A MUST BE DIMENSIONED AT LEAST M+1 BY
C             M+1.
C       U     A DOUBLY SUBSCRIPTED ARRAY THAT, ON
C             RETURN, CONTAINS THE ORTHOGONAL MATRIX
C             THAT REDUCES A TO REAL SCHUR FORM.
C       M     THE ORDER OF THE MATRIX A.
C       NA    THE FIRST DIMENSION OF THE ARRAY A.
C       NU    THE FIRST DIMENSION OF THE ARRAY U.
C       B     A DOUBLY SUBSCRIPTED ARRAY CONTAINING THE
C             MATRIX B.  ON RETURN, THE UPPER TRIANGLE
C             AND SUBDIAGONAL OF THE ARRAY B CONTAIN AN
C             UPPER REAL SCHUR FORM OF B.  THE ARRAY B
C             MUST BE DIMENSIONED AT LEAST M+1 BY M+1.
C       V     A DOUBLY SUBSCRIPTED ARRAY THAT, ON
C             RETURN, CONTAINS THE ORTHOGONAL MATRIX
C             THAT REDUCES B TO REAL SCHUR FORM.
C       N     THE ORDER OF THE MATRIX B.
C       NB    THE FIRST DIMENSION OF THE ARRAY B.
C       NV    THE FIRST DIMENSION OF THE ARRAY V.
C       C     A DOUBLY SUBSCRIPTED ARRAY CONTAINING THE
C             MATRIX C.  ON RETURN, C CONTAINS THE
C             SOLUTION MATRIX X.
C       NC    THE FIRST DIMENSION OF THE ARRAY C.
C       EPSA  A CONVERGENCE CRITERION FOR THE REDUCTION
C             OF A TO SCHUR FORM.  EPSA SHOULD BE SET
C             SLIGHTLY SMALLER THAN 10.**(-N), WHERE N
C             IS THE NUMBER OF SIGNIFICANT DIGITS IN
C             THE ELEMENTS OF THE MATRIX A.
C       EPSB  A CONVERGENCE CRITERION FOR THE REDUCTION
C             OF B TO REAL SCHUR FORM.
C       FAIL  AN INTEGER VARIABLE THAT, ON RETURN,
C             CONTAINS AN ERROR SIGNAL.  IF FAIL IS
C             POSITIVE (NEGATIVE) THEN THE PROGRAM WAS
C             UNABLE TO REDUCE A (B) TO REAL SCHUR
C             FORM.  IF FAIL IS ZERO, THE REDUCTIONS
C             PROCEEDED WITHOUT MISHAP.
C
C WHEN EPSA IS NEGATIVE THE REDUCTION OF A TO REAL SCHUR
C FORM IS SKIPPED AND THE ARRAYS A AND U ARE ASSUMED TO
C CONTAIN THE SCHUR FORM AND ACCOMPANYING ORTHOGONAL MATRIX.
C THIS PERMITS THE EFFICIENT SOLUTION OF SEVERAL EQUATIONS
C OF THE FORM AX + BX = C WHEN A DOES NOT CHANGE.  LIKEWISE,
C IF EPSB IS NEGATIVE, THE REDUCTION OF B TO REAL SCHUR FORM
C IS SKIPPED.
C
      REAL*8
     1A(NA,1),U(NU,1),B(NB,1),V(NV,1),C(NC,1),EPSA,EPSB,TEMP
      INTEGER
     1M,NA,NU,N,NB,NV,NC,FAIL,M1,MM1,N1,NM1,I,J,K
      M1 = M+1
      MM1 = M-1
      N1 = N+1
```

```
      NM1 = N-1
C
C IF REQUIRED, REDUCE A TO LOWER REAL*8 SCHUR FORM.
C
      IF(EPSA .LT. 0.) GO TO 35
      DO 10 I=1,M
       DO 10 J=I,M
        TEMP = A(I,J)
        A(I,J) = A(J,I)
        A(J,I) = TEMP
   10 CONTINUE
      CALL HSHLDR(A,M,NA)
      CALL BCKMLT(A,U,M,NA,NU)
      IF(MM1 .EQ. 0) GO TO 25
      DO 20 I=1,MM1
       A(I+1,I) = A(I,M1)
   20 CONTINUE
      CALL SCHUR(A,U,M,NA,NU,EPSA,FAIL)
      IF(FAIL .NE. 0) RETURN
   25 DO 30 I=1,M
       DO 30 J=I,M
        TEMP = A(I,J)
        A(I,J) = A(J,I)
        A(J,I) = TEMP
   30 CONTINUE
C
C IF REQUIRED, REDUCE B TO UPPER REAL*8 SCHUR FORM.
C
   35 IF(EPSB .LT. 0.) GO TO 45
      CALL HSHLDR(B,N,NB)
      CALL BCKMLT(B,V,N,NB,NV)
      IF(NM1 .EQ. 0) GO TO 45
      DO 40 I=1,NM1
       B(I+1,I) = B(I,N1)
   40 CONTINUE
      CALL SCHUR(B,V,N,NB,NV,EPSB,FAIL)
      FAIL = -FAIL
      IF(FAIL .NE. 0) RETURN
C
C TRANSFORM C.
C
   45 DO 60 J=1,N
       DO 50 I=1,M
        A(I,M1) = 0.
        DO 50 K=1,M
         A(I,M1) = A(I,M1) + U(K,I)*C(K,J)
   50  CONTINUE
       DO 60 I=1,M
        C(I,J) = A(I,M1)
   60 CONTINUE
      DO 80 I=1,M
       DO 70 J=1,N
        B(N1,J) = 0.
        DO 70 K=1,N
         B(N1,J) = B(N1,J) + C(I,K)*V(K,J)
   70  CONTINUE
```

```
      DO 80 J=1,N
       C(I,J) = B(N1,J)
   80 CONTINUE
C
C SOLVE THE TRANSFORMED SYSTEM.
C
      CALL SHRSLV(A,B,C,M,N,NA,NB,NC)
C
C TRANSFORM C BACK TO THE SOLUTION.
C
      DO 100 J=1,N
       DO 90 I=1,M
        A(I,M1) = 0.
        DO 90 K=1,M
         A(I,M1) = A(I,M1) + U(I,K)*C(K,J)
   90  CONTINUE
       DO 100 I=1,M
        C(I,J) = A(I,M1)
  100 CONTINUE
      DO 120 I=1,M
       DO 110 J=1,N
        B(N1,J) = 0.
        DO 110 K=1,N
         B(N1,J) = B(N1,J) + C(I,K)*V(J,K)
  110  CONTINUE
       DO 120 J=1,N
        C(I,J) = B(N1,J)
  120 CONTINUE
      RETURN
      END
      SUBROUTINE SHRSLV(A,B,C,M,N,NA,NB,NC)
C SHRSLV IS A FORTRAN IV SUBROUTINE TO SOLVE THE REAL*8 MATRIX
C EQATION AX + XB = C, WHERE A IS IN LOWER REAL*8 SCHUR FORM
C AND B IS IN UPPER REAL*8 SCHUR FORM.  SHRSLV USES THE AUX-
C ILIARY SUBROUTINE SYSSLV, WHICH IT COMMUNICATES WITH
C THROUGH THE COMMON BLOCK SLVBLK.  THE PARAMETERS IN THE
C ARGUMENT LIST ARE
C       A    A DOUBLY SUBSCRIPTED ARRAY CONTAINING THE
C            MATRIX A IN LOWER REAL*8 SCHUR FORM.
C       B    A DOUBLY SUBSCRIPTED ARRAY CONTAINING THE
C            MATRIX B IN UPPER REAL*8 SCHUR FORM.
C       C    A DOUBLY SUBSCRIPTED ARRAY CONTAINING THE
C            MATRIX C.
C       M    THE ORDER OF THE MATRIX A.
C       N    THE ORDER OF THE MATRIX B.
C       NA   THE FIRST DIMENSION OF THE ARRAY A.
C       NB   THE FIRST DIMENSION OF THE ARRAY B.
C       NC   THE FIRST DIMENSION OF THE ARRAY C.
C
      REAL*8
     1A(NA,1),B(NB,1),C(NC,1),T,P
      INTEGER
     1M,N,NA,NB,NC,K,KM1,DK,KK,L,LM1,DL,LL,I,IB,J,JA,NSYS
      COMMON/SLVBLK/T(5,5),P(5),NSYS
      L = 1
   10 LM1 = L-1
```

```
      DL = 1
      IF(L .EQ. N) GO TO 15
      IF(B(L+1,L) .NE. 0.) DL = 2
   15 LL = L+DL-1
      IF(L .EQ. 1) GO TO 30
      DO 20 J=L,LL
       DO 20 I=1,M
        DO 20 IB=1,LM1
         C(I,J) = C(I,J) - C(I,IB)*B(IB,J)
   20 CONTINUE
   30 K = 1
   40  KM1 = K-1
      DK = 1
      IF(K .EQ. M) GO TO 45
      IF(A(K,K+1) .NE. 0.) DK = 2
   45  KK = K+DK-1
      IF(K .EQ. 1) GO TO 60
      DO 50 I=K,KK
       DO 50 J=L,LL
        DO 50 JA=1,KM1
         C(I,J) = C(I,J) - A(I,JA)*C(JA,J)
   50  CONTINUE
   60  IF(DL .EQ. 2) GO TO 80
      IF(DK .EQ. 2) GO TO 70
      T(1,1) = A(K,K) + B(L,L)
      IF(T(1,1) .EQ. 0.) STOP
      C(K,L) = C(K,L)/T(1,1)
      GO TO 100
   70  T(1,1) = A(K,K) + B(L,L)
      T(1,2) = A(K,KK)
      T(2,1) = A(KK,K)
      T(2,2) = A(KK,KK) + B(L,L)
      P(1) = C(K,L)
      P(2) = C(KK,L)
      NSYS = 2
      CALL SYSSLV
      C(K,L) = P(1)
      C(KK,L) = P(2)
      GO TO 100
   80  IF(DK .EQ. 2) GO TO 90
      T(1,1) = A(K,K) + B(L,L)
      T(1,2) = B(LL,L)
      T(2,1) = B(L,LL)
      T(2,2) = A(K,K) + B(LL,LL)
      P(1) = C(K,L)
      P(2) = C(K,LL)
      NSYS = 2
      CALL SYSSLV
      C(K,L) = P(1)
      C(K,LL) = P(2)
      GO TO 100
   90  T(1,1) = A(K,K) + B(L,L)
      T(1,2) = A(K,KK)
      T(1,3) = B(LL,L)
      T(1,4) = 0.
      T(2,1) = A(KK,K)
```

```
      T(2,2) = A(KK,KK) + B(L,L)
      T(2,3) = 0.
      T(2,4) = T(1,3)
      T(3,1) = B(L,LL)
      T(3,2) = 0.
      T(3,3) = A(K,K) + B(LL,LL)
      T(3,4) = T(1,2)
      T(4,1) = 0.
      T(4,2) = T(3,1)
      T(4,3) = T(2,1)
      T(4,4) = A(KK,KK) + B(LL,LL)
      P(1) = C(K,L)
      P(2) = C(KK,L)
      P(3) = C(K,LL)
      P(4) = C(KK,LL)
      NSYS = 4
      CALL SYSSLV
      C(K,L) = P(1)
      C(KK,L) = P(2)
      C(K,LL) = P(3)
      C(KK,LL) = P(4)
  100 K = K + DK
      IF(K .LE. M) GO TO 40
      L = L + DL
      IF(L .LE. N) GO TO 10
      RETURN
      END
      SUBROUTINE ATXPXA(A,U,C,N,NA,NU,NC,EPS,FAIL)
C
C ATXPXA IS A FORTRAN IV SUBROUTINE TO SOLVE THE REAL*8 MATRIX
C EQUATION TRANS(A)*X + X*A = C, WHERE C IS SYMMETRIC AND
C TRANS(A) DENOTES THE TRANSPOSE OF A.  THE EQUATION IS
C TRANSFORMED SO THAT A IS IN UPPER REAL*8 SCHUR FORM, AND THE
C TRANSFORMED EQUATION IS SOLVED BY A RECURSIVE PROCEDURE.
C THE PROGRAM REQUIRES THE AUXILIARY SUBROUTINES HSHLDR,
C BCKMLT, SCHUR, AND SYMSLV.  THE PARAMETERS IN THE ARGUMENT
C LIST ARE
C       A    A DOUBLY SUBSCRIPTED ARRAY CONTAINING THE
C            MATRIX A.  ON RETURN, THE UPPER TRIANGLE
C            AND THE FIRST SUBDIAGONAL OF THE ARRAY A
C            CONTAIN AN UPPER REAL*8 SCHUR FORM OF A.
C            THE ARRAY A MUST BE DIMENSIONED AT LEAST
C            N+1 BY N+1.
C       U    A DOUBLY SUBSCRIPTED ARRAY THAT, ON
C            RETURN, CONTAINS THE ORTHOGONAL MATRIX
C            THAT REDUCES A TO UPPER REAL*8 SCHUR FORM.
C       C    A DOUBLY SUBSCRIPTED ARRAY CONTAINING THE
C            MATRIX C.  ON RETURN, C CONTAINS THE
C            SOLUTION MATRIX X.
C       N    THE ORDER OF THE MATRIX A.
C       NA   THE FIRST DIMENSION OF THE ARRAY A.
C       NU   THE FIRST DIMENSION OF THE ARRAY U.
C       NC   THE FIRST DIMENSION OF THE ARRAY C.
C       EPS  A CONVERGENCE CRITERION FOR THE REDUCTION
C            OF A TO REAL*8 SCHUR FORM.  EPS SHOULD BE
C            SET SLIGHTLY SMALLER THAN 10.**(-N),
```

```
C            WHERE N IS THE NUMBER OF SIGNIFICANT
C            DIGITS IN THE ELEMENTS OF THE MATRIX A.
C       FAIL  AN INTEGER VARIABLE THAT, ON RETURN,
C            CONTAINS AN ERROR SIGNAL.  IF FAIL IS
C            NONZERO, THEN THE PROGRAM WAS UNABLE TO
C            REDUCE A TO REAL*8 SCHUR FORM.  IF FAIL IS
C            ZERO, THE REDUCTION PROCEEDED WITHOUT
C            MISHAP.
C
C WHEN EPS IS NEGATIVE, THE REDUCTION OF A TO REAL*8 SCHUR
C FORM IS SKIPPED AND THE ARRAYS A AND U ARE ASSUMED TO
C CONTAIN THE SCHUR FORM AND ACCOMPANYING ORTHOGONAL MATRIX.
C THIS PERMITS THE EFFICIENT SOLUTION OF SEVERAL EQUATIONS
C WITH DIFFERENT RIGHT HAND SIDES.
C
      REAL*8
     1A(NA,1),U(NU,1),C(NC,1),EPS
      INTEGER
     1N,NA,NU,NC,FAIL,N1,NM1,I,J,K
      N1 = N+1
      NM1 = N-1
C
C IF REQUIRED, REDUCE A TO UPPER REAL*8 SCHUR FORM.
C
      IF(EPS .LT. 0.) GO TO 15
      CALL HSHLDR(A,N,NA)
      CALL BCKMLT(A,U,N,NA,NU)
      DO 10 I=1,NM1
        A(I+1,I) = A(I,N1)
   10 CONTINUE
      CALL SCHUR(A,U,N,NA,NU,EPS,FAIL)
      IF(FAIL .NE. 0) RETURN
C
C TRANSFORM C.
C
   15 DO 20 I=1,N
        C(I,I) = C(I,I)/2.
   20 CONTINUE
      DO 40 I=1,N
        DO 30 J=1,N
          A(N1,J) = 0.
          DO 30 K=I,N
            A(N1,J) = A(N1,J) + C(I,K)*U(K,J)
   30   CONTINUE
        DO 40 J=1,N
          C(I,J) = A(N1,J)
   40 CONTINUE
      DO 60 J=1,N
        DO 50 I=1,N
          A(I,N1) = 0.
          DO 50 K=1,N
            A(I,N1) = A(I,N1) + U(K,I)*C(K,J)
   50   CONTINUE
        DO 60 I=1,N
          C(I,J) = A(I,N1)
   60 CONTINUE
```

```
      DO 70 I=1,N
       DO 70 J=I,N
        C(I,J) = C(I,J) + C(J,I)
        C(J,I) = C(I,J)
   70 CONTINUE
C
C SOLVE THE TRANSFORMED SYSTEM.
C
      CALL SYMSLV(A,C,N,NA,NC)
C
C TRANSFORM C BACK TO THE SOLUTION.
C
      DO 80 I=1,N
       C(I,I) = C(I,I)/2.
   80 CONTINUE
      DO 100 I=1,N
       DO 90 J=1,N
        A(N1,J) = 0.
        DO 90 K=I,N
         A(N1,J) = A(N1,J) + C(I,K)*U(J,K)
   90  CONTINUE
       DO 100 J=1,N
        C(I,J) = A(N1,J)
  100 CONTINUE
      DO 120 J=1,N
       DO 110 I=1,N
        A(I,N1) = 0.
        DO 110 K=1,N
         A(I,N1) = A(I,N1) + U(I,K)*C(K,J)
  110  CONTINUE
       DO 120 I=1,N
        C(I,J) = A(I,N1)
  120 CONTINUE
      DO 130 I=1,N
       DO 130 J=I,N
        C(I,J) = C(I,J) + C(J,I)
        C(J,I) = C(I,J)
  130 CONTINUE
      RETURN
      END
      SUBROUTINE SYMSLV(A,C,N,NA,NC)
C
C SYMSLV IS A FORTRAN IV SUBROUTINE TO SOLVE THE REAL*8 MATRIX
C EQUATION TRANS(A)*X + X*A = C, WHERE C IS SYMMETRIC, A IS
C IN UPPER REAL*8 SCHUR FORM, AND TRANS(A) DENOTES THE TRANS-
C POSE OF A.  SYMSLV USES THE AUXILIARY SUBROUTINE SYSSLV,
C WHICH IT COMMUNICATES WITH THROUGH THE COMMON BLOCK
C SLVBLK.  THE PARAMETERS IN THE ARGUMENT LIST ARE
C       A    A DOUBLY SUBSCRIPTED ARRAY CONTAINING THE
C            MATRIX A IN UPPER REAL*8 SCHUR FORM.
C       C    A DOUBLY SUBSCRIPTED ARRAY CONTAINING THE
C            MATRIX C.
C       N    THE ORDER OF THE MATRIX A.
C       NA   THE FIRST DIMENSION OF THE ARRAY A.
C       NC   THE FIRST DIMENSION OF THE ARRAY C.
C
```

```
      REAL*8
     1A(NA,1),C(NC,1),T,P
      INTEGER
     1N,NA,NC,K,KK,DK,KM1,L,LL,DL,LDL,I,IA,J,NSYS
      COMMON/SLVBLK/T(5,5),P(5),NSYS
      L = 1
   10 DL = 1
      IF(L .EQ. N) GO TO 20
      IF(A(L+1,L) .NE. 0.) DL = 2
   20 LL = L+DL-1
      K = L
   30   KM1 = K-1
        DK = 1
        IF(K .EQ. N) GO TO 35
        IF(A(K+1,K) .NE. 0.) DK = 2
   35   KK = K+DK-1
        IF(K .EQ. L) GO TO 45
        DO 40 I=K,KK
          DO 40 J=L,LL
            DO 40 IA=L,KM1
              C(I,J) = C(I,J) - A(IA,I)*C(IA,J)
   40   CONTINUE
   45   IF(DL .EQ. 2) GO TO 60
        IF(DK .EQ. 2 ) GO TO 50
        T(1,1) = A(K,K) + A(L,L)
        IF(T(1,1) .EQ. 0.) STOP
        C(K,L) = C(K,L)/T(1,1)
        GO TO 90
   50   T(1,1) = A(K,K) + A(L,L)
        T(1,2) = A(KK,K)
        T(2,1) = A(K,KK)
        T(2,2) = A(KK,KK) + A(L,L)
        P(1) = C(K,L)
        P(2) = C(KK,L)
        NSYS = 2
        CALL SYSSLV
        C(K,L) = P(1)
        C(KK,L) = P(2)
        GO TO 90
   60   IF(DK .EQ. 2) GO TO 70
        T(1,1) = A(K,K) + A(L,L)
        T(1,2) = A(LL,L)
        T(2,1) = A(L,LL)
        T(2,2) = A(K,K) + A(LL,LL)
        P(1) = C(K,L)
        P(2) = C(K,LL)
        NSYS = 2
        CALL SYSSLV
        C(K,L) = P(1)
        C(K,LL) = P(2)
        GO TO 90
   70   IF(K .NE. L) GO TO 80
        T(1,1) = A(L,L)
        T(1,2) = A(LL,L)
        T(1,3) = 0.
        T(2,1) = A(L,LL)
```

```
      T(2,2) = A(L,L) + A(LL,LL)
      T(2,3) = T(1,2)
      T(3,1) = 0.
      T(3,2) = T(2,1)
      T(3,3) = A(LL,LL)
      P(1) = C(L,L)/2.
      P(2) = C(LL,L)
      P(3) = C(LL,LL)/2.
      NSYS = 3
      CALL SYSSLV
      C(L,L) = P(1)
      C(LL,L) = P(2)
      C(L,LL) = P(2)
      C(LL,LL) = P(3)
      GO TO 90
   80   T(1,1) = A(K,K) + A(L,L)
      T(1,2) = A(KK,K)
      T(1,3) = A(LL,L)
      T(1,4) = 0.
      T(2,1) = A(K,KK)
      T(2,2) = A(KK,KK) + A(L,L)
      T(2,3) = 0.
      T(2,4) = T(1,3)
      T(3,1) = A(L,LL)
      T(3,2) = 0.
      T(3,3) = A(K,K) + A(LL,LL)
      T(3,4) = T(1,2)
      T(4,1) = 0.
      T(4,2) = T(3,1)
      T(4,3) = T(2,1)
      T(4,4) = A(KK,KK) + A(LL,LL)
      P(1) = C(K,L)
      P(2) = C(KK,L)
      P(3) = C(K,LL)
      P(4) = C(KK,LL)
      NSYS = 4
      CALL SYSSLV
      C(K,L) = P(1)
      C(KK,L) = P(2)
      C(K,LL) = P(3)
      C(KK,LL) = P(4)
   90 K = K + DK
     IF(K .LE. N) GO TO 30
     LDL = L + DL
     IF(LDL .GT. N) RETURN
     DO 120 J=LDL,N
      DO 100 I=L,LL
       C(I,J) = C(J,I)
  100   CONTINUE
      DO 120 I=J,N
       DO 110 K=L,LL
        C(I,J) = C(I,J) - C(I,K)*A(K,J) - A(K,I)*C(K,J)
  110    CONTINUE
       C(J,I) = C(I,J)
  120 CONTINUE
    L = LDL
```

```
      GO TO 10
      END
      SUBROUTINE HSHLDR(A,N,NA)
C
C HSHLDR IS A FORTRAN IV SUBROUTINE TO REDUCE A MATRIX TO
C UPPER HESSENBERG FORM BY ELEMENTARY HERMITIAN TRANSFORMA-
C TIONS (THE METHOD OF HOUSEHOLDER).  THE PARAMETERS IN THE
C ARGUMENT LIST ARE
C       A    A DOUBLY SUBSCRIPTED ARRAY CONTAINING THE
C            MATRIX A.  ON RETURN, THE UPPER TRIANGLE
C            OF THE ARRAY A MATRIX AND THE (N+1)-TH
C            COLUMN CONTAIN THE SUBDIAGONAL ELEMENTS
C            OF THE TRANSFORMED MATRIX.  ON RETURN,
C            THE LOWER TRIANGLE AND THE (N+1)-TH ROW
C            OF THE ARRAY A CONTAIN A HISTORY OF THE
C            TRANSFORMATIONS.
C       N    THE ORDER OF THE MATRIX A.
C       NA   THE FIRST DIMENSION OF THE ARRAY A.
C
      REAL*8
     1A(NA,1),MAX,SUM,S,P
      INTEGER
     1N,NA,NM2,N1,L,L1,I,J
      NM2 = N-2
      N1 = N+1
      IF(N .EQ. 1) RETURN
      IF(N .GT. 2) GO TO 5
      A(1,N1) = A(2,1)
      RETURN
    5 DO 80 L=1,NM2
        L1 = L+1
        MAX = 0.
        DO 10 I=L1,N
          MAX = DMAX1(MAX,DABS(A(I,L)))
   10   CONTINUE
        IF(MAX .NE. 0.) GO TO 20
        A(L,N1) = 0.
        A(N1,L) = 0.
        GO TO 80
   20   SUM = 0.
        DO 30 I=L1,N
          A(I,L) = A(I,L)/MAX
          SUM = SUM + A(I,L)**2
   30   CONTINUE
        S = DSIGN(DSQRT(SUM),A(L1,L))
        A(L,N1) = -MAX*S
        A(L1,L) = S + A(L1,L)
        A(N1,L) = S*A(L1,L)
        DO 50 J=L1,N
          SUM = 0.
          DO 40 I=L1,N
            SUM = SUM + A(I,L)*A(I,J)
   40     CONTINUE
          P = SUM/A(N1,L)
          DO 50 I=L1,N
            A(I,J) = A(I,J) - A(I,L)*P
```

```
 50   CONTINUE
      DO 70 I=1,N
       SUM = 0.
       DO 60 J=L1,N
        SUM = SUM + A(I,J)*A(J,L)
 60    CONTINUE
       P = SUM/A(N1,L)
       DO 70 J=L1,N
        A(I,J) = A(I,J) - P*A(J,L)
 70   CONTINUE
 80  CONTINUE
     A(N-1,N1) = A(N,N-1)
     RETURN
     END
     SUBROUTINE BCKMLT(A,U,N,NA,NU)
C
C BCKMLT IS A FORTRAN IV SUBROUTINE THAT, GIVEN THE OUTPUT
C OF THE SUBROUTINE HSHLDR, COMPUTES THE ORTHOGONAL MATRIX
C THAT REDUCES A TO UPPER HESSENBERG FORM.  THE PARAMETERS
C IN THE ARGUMENT LIST ARE
C       A     A DOUBLY SUBSCRIPTED ARRAY CONTAINING THE
C             OUTPUT FROM HSHLDR.
C       U     A DOUBLY SUBSCRIPTED ARRAY THAT, ON
C             RETURN, CONTAINS THE ORTHOGONAL MATRIX.
C       N     THE ORDER OF THE MATRIX A IN HSHLDR.
C       NA    THE FIRST DIMENSION OF THE ARRAY A.
C       NU    THE FIRST DIMENSION OF THE ARRAY U.
C
C THE ARRAYS A AND U MAY BE IDENTIFIED IN THE CALLING
C SEQUENCE.  IF THIS IS DONE, THE ELEMENTS OF THE ORTHOGONAL
C MATRIX WILL OVERWRITE THE OUTPUT OF HSHLDR.
C
     REAL*8
    1A(NA,1),U(NU,1),SUM,P
     INTEGER
    1N,NA,N1,NM1,NM2,LL,L,L1,I,J
     N1 = N+1
     NM1 = N-1
     NM2 = N-2
     U(N,N)= 1.
     IF(NM1 .EQ. 0) RETURN
     U(NM1,N) = 0.
     U(N,NM1) = 0.
     U(NM1,NM1) = 1.
     IF(NM2 .EQ. 0) RETURN
     DO 40 LL=1,NM2
      L = NM2-LL+1
      L1 = L+1
      IF(A(N1,L) .EQ. 0.) GO TO 25
      DO 20 J=L1,N
       SUM = 0.
       DO 10 I=L1,N
        SUM = SUM + A(I,L)*U(I,J)
 10    CONTINUE
       P = SUM/A(N1,L)
       DO 20 I=L1,N
```

```
        U(I,J) = U(I,J) - A(I,L)*P
  20  CONTINUE
  25  DO 30 I=L1,N
        U(I,L) = 0.
        U(L,I) = 0.
  30  CONTINUE
      U(L,L) = 1.
  40 CONTINUE
    RETURN
    END
    SUBROUTINE SCHUR(H,U,NN,NH,NU,EPS,FAIL)
C
C SCHUR IS A FORTRAN IV SUBROUTINE TO REDUCE AN UPPER
C HESSENBERG MATRIX TO REAL*8 SCHUR FORM BY THE QR METHOD WITH
C IMPLICIT ORIGIN SHIFTS.  THE PRODUCT OF THE TRANSFORMA-
C TIONS USED IN THE REDUCTION IS ACCUMULATED.  SCHUR IS AN
C ADAPTATION OF THE ALGOL PROGRAM HQR BY MARTIN, PETERS, AND
C WILKINSON (NUMER. MATH. 14 (1970) 219-231).  THE PARA-
C METERS IN THE ARGUMENT LIST ARE
C       H     A DOUBLY SUBSCRIPTED ARRAY CONTAINING THE
C             UPPER HESSENBERG MATRIX H.  ON RETURN, H
C             CONTAINS AN UPPER REAL*8 SCHUR FORM OF H.
C             THE ELEMENTS OF THE ARRAY H BELOW THE
C             THIRD SUBDIAGONAL ARE UNDISTURBED.
C       U     A DOUBLY SUBSCRIPTED ARRAY CONTAINING ANY
C             MATRIX.  ON RETURN, U CONTAINS THE MATRIX
C             U*R(1)*R(2)..., WHERE R(I) ARE THE TRANS-
C             FORMATIONS USED IN THE REDUCTION OF H.
C       NN    THE ORDER OF THE MATRICES H AND U.
C       NH    THE FIRST DIMENSION OF THE ARRAY H.
C       NU    THE FIRST DIMENSION OF THE ARRAY U.
C       EPS   A NUMBER USED IN DETERMINING WHEN AN
C             ELEMENT OF H IS NEGLIGIBLE.  H(I,J) IS
C             NEGLIGIBLE IF ABS(H(I,J)) IS LESS THAN OR
C             EQUAL TO EPS TIMES THE INFINITY NORM OF
C             H.
C       FAIL  AN INTEGER VARIABLE THAT, ON RETURN,
C             CONTAINS AN ERROR SIGNAL. IF FAIL IS
C             POSITIVE, THEN THE PROGRAM FAILED TO MAKE
C             THE FAIL-1 OR FAIL-2 SUBDIAGONAL ELEMENT
C             NEGLIGIBLE AFTER 30 ITERATIONS.
C
    REAL*8
   1H(NH,1),U(NU,1),EPS,HN,RSUM,TEST,P,Q,R,S,W,X,Y,Z
    INTEGER
   1NN,NA,NH,FAIL,I,ITS,J,JL,K,L,LL,M,MM,M2,M3,N
    LOGICAL
   1LAST
    N = NN
    HN = 0.
    DO 20 I=1,N
     JL = MAX0(1,I-1)
     RSUM = 0.
     DO 10 J=JL,N
      RSUM = RSUM + DABS(H(I,J))
  10  CONTINUE
```

```
      HN = DMAX1(HN,RSUM)
   20 CONTINUE
      TEST = EPS*HN
      IF(HN .EQ. 0.) GO TO 230
   30 IF(N .LE. 1) GO TO 230
      ITS = 0
      NA = N-1
      NM2 = N-2
   40 DO 50 LL=2,N
      L = N-LL+2
       IF(DABS(H(L,L-1)) .LE. TEST) GO TO 60
   50 CONTINUE
      L = 1
      GO TO 70
   60 H(L,L-1) = 0.
   70 IF(L .LT. NA) GO TO 72
      N = L-1
      GO TO 30
   72 X = H(N,N)/HN
      Y = H(NA,NA)/HN
      R = (H(N,NA)/HN)*(H(NA,N)/HN)
      IF(ITS .LT. 30) GO TO 75
      FAIL = N
      RETURN
   75 IF(ITS.EQ.10 .OR. ITS.EQ.20) GO TO 80
      S = X + Y
      Y = X*Y - R
      GO TO 90
   80 Y = (DABS(H(N,NA)) + DABS(H(NA,NM2)))/HN
      S = 1.5*Y
      Y = Y**2
   90 ITS = ITS + 1
      DO 100 MM=L,NM2
       M = NM2-MM+L
       X = H(M,M)/HN
       R = H(M+1,M)/HN
       Z = H(M+1,M+1)/HN
       P = X*(X-S) + Y + R*(H(M,M+1)/HN)
       Q = R*(X+Z-S)
       R = R*(H(M+2,M+1)/HN)
       W = DABS(P) + DABS(Q) + DABS(R)
       P = P/W
       Q = Q/W
       R = R/W
       IF(M .EQ. L) GO TO 110
      IF(DABS(H(M,M-1))*(DABS(Q)+DABS(R)) .LE. DABS(P)*TEST)
     1GO TO 110
  100 CONTINUE
  110 M2 = M+2
      M3 = M+3
      DO 120 I=M2,N
       H(I,I-2) = 0.
  120 CONTINUE
      IF(M3 .GT. N) GO TO 140
      DO 130 I=M3,N
       H(I,I-3) = 0.
```

```
 130 CONTINUE
 140 DO 220 K=M,NA
     LAST = K.EQ.NA
     IF(K .EQ. M) GO TO 150
     P = H(K,K-1)
     Q = H(K+1,K-1)
     R = 0.
     IF(.NOT.LAST) R = H(K+2,K-1)
     X = DABS(P) + DABS(Q) + DABS(R)
     IF(X .EQ. 0.) GO TO 220
     P = P/X
     Q = Q/X
     R = R/X
 150  S = DSQRT(P**2 + Q**2 + R**2)
     IF(P .LT. 0.) S = -S
     IF(K .NE. M) H(K,K-1) = -S*X
     IF(K.EQ.M .AND. L.NE.M) H(K,K-1) = -H(K,K-1)
     P = P + S
     X = P/S
     Y = Q/S
     Z = R/S
     Q = Q/P
     R = R/P
     DO 170 J=K,NN
      P = H(K,J) + Q*H(K+1,J)
      IF(LAST) GO TO 160
      P = P + R*H(K+2,J)
      H(K+2,J) = H(K+2,J) - P*Z
 160   H(K+1,J) = H(K+1,J) - P*Y
      H(K,J) = H(K,J) - P*X
 170  CONTINUE
     J = MIN0(K+3,N)
     DO 190 I=1,J
      P = X*H(I,K) + Y*H(I,K+1)
      IF(LAST) GO TO 180
      P = P + Z*H(I,K+2)
      H(I,K+2) = H(I,K+2) - P*R
 180   H(I,K+1) = H(I,K+1) - P*Q
      H(I,K) = H(I,K) - P
 190  CONTINUE
     DO 210 I=1,NN
      P = X*U(I,K) + Y*U(I,K+1)
      IF(LAST) GO TO 200
      P = P + Z*U(I,K+2)
      U(I,K+2) = U(I,K+2) - P*R
 200   U(I,K+1) = U(I,K+1) - P*Q
      U(I,K) = U(I,K) - P
 210  CONTINUE
 220 CONTINUE
    GO TO 40
 230 FAIL = 0
    RETURN
    END
    SUBROUTINE SYSSLV
C
C SYSSLV IS A FORTRAN IV SUBROUTINE THAT SOLVES THE LINEAR
```

```
C SYSTEM AX = B OF ORDER N LESS THAN 5 BY CROUT REDUCTION
C FOLLOWED BY BACK SUBSTITUTION.  THE MATRIX A, THE VECTOR
C B, AND THE ORDER N ARE CONTAINED IN THE ARRAYS A,B, AND
C THE VARIABLE N OF THE COMMON BLOCK SLVBLK.  THE SOLUTION
C IS RETURNED IN THE ARRAY B.
C
      REAL*8 A,B,AA,TEMP
      INTEGER N,NM1,N1,K,KM1,KP1,INTR,I,J,II
    COMMON/SLVBLK/A(5,5),B(5),N
    REAL*8 MAX
   1 NM1 = N-1
    N1 = N+1
C
C COMPUTE THE LU FACTORIZATION OF A.
C
    DO 80 K=1,N
     KM1 = K-1
     IF(K.EQ.1) GO TO 20
     DO 10 I=K,N
      DO 10 J=1,KM1
       A(I,K) = A(I,K) - A(I,J)*A(J,K)
  10  CONTINUE
  20  IF(K.EQ.N) GO TO 100
     KP1 = K+1
    MAX = DABS(A(K,K))
     INTR = K
     DO 30 I=KP1,N
      AA = DABS(A(I,K))
      IF(AA .LE. MAX) GO TO 30
      MAX = AA
      INTR = I
  30  CONTINUE
     IF(MAX .EQ. 0.) STOP
     A(N1,K) = INTR
     IF(INTR .EQ. K) GO TO 50
     DO 40 J=1,N
      TEMP = A(K,J)
      A(K,J) = A(INTR,J)
      A(INTR,J) = TEMP
  40  CONTINUE
  50  DO 80 J=KP1,N
      IF(K.EQ.1) GO TO 70
      DO 60 I=1,KM1
       A(K,J) = A(K,J) - A(K,I)*A(I,J)
  60   CONTINUE
  70   A(K,J) = A(K,J)/A(K,K)
  80 CONTINUE
C
C INTERCHANGE THE COMPONENTS OF B.
C
 100 DO 110 J=1,NM1
     INTR = A(N1,J)
     IF(INTR .EQ. J) GO TO 110
     TEMP = B(J)
     B(J) = B(INTR)
     B(INTR) = TEMP
```

```
 110 CONTINUE
C
C SOLVE LX = B.
C
 200 B(1) = B(1)/A(1,1)
     DO 220 I=2,N
      IM1 = I-1
      DO 210 J=1,IM1
       B(I) = B(I) - A(I,J)*B(J)
 210  CONTINUE
      B(I) = B(I)/A(I,I)
 220 CONTINUE
C
C SOLVE UX = B.
C
 300 DO 310 II=1,NM1
      I = NM1-II+1
      II = I+1
      DO 310 J=I1,N
       B(I) = B(I) - A(I,J)*B(J)
 310 CONTINUE
     RETURN
     END
C     **********************************************************************
C     **********************************************************************
C     **********************************************************************
C     **********************************************************************
C     **********************************************************************
```

فهرست مراجع:

۱-جرالد، وتیلی (ترجمهٔ علی محمد پورپاک) ، محاسبات عددی ؛ آنالیز عددی کاربردی برای رشته‌های مهندسی و علوم همراه با ۱۰۰ برنامهٔ کامپیوتری به زبانهای C و Pascal و Fortran و Basic شامل ۳۰۰ مسئله با جواب ، جهاد دانشگاهی ۱۳۸۰.

۲-ایروینگ اچ شیمز (ترجمهٔ مهندس علیرضا انتظاری). مکانیک سیالات ، انتشارات دانشگاه آزاد اسلامی واحد کرج، ۱۳۷۲.

۳-ک. ا. هافمن (ترجمهٔ دکتر احمد رضا عظیمیان). دینامیک سیالات محاسباتی برای مهندسان ، مرکز نشر دانشگاهی صنعتی اصفهان

۴-یان م. اسمیت (ترجمهٔ دکتر محمود مشعل)، برنامه نویسی در فرترن ۹۰ ، انتشارات جهاد دانشگاهی تهران ،۱۳۸۲

۵-دیتل پل ،تم نیتو (ترجمهٔ مهندس بهرام پاشایی)، راهنمای جامع برنامه نویسان ویژوال بیسیک۶، انتشارات جهان نو، ۱۳۸۱

۶-ریچارد ال.بوردن،ج. دوگلاس فیرز، آلبرت سی.رینولدز(ترجمهٔ علی اکبر عالم زاده – اسماعیل بابلیان- محمدرضا امیدوار) ، آنالیز ریاضی؛ انتشارات منصوری ،۱۳۷۳

۷-حسین ایزی، حل مستقیم عددی جریان جت ، دانشکدهٔ مکانیک دانشگاه صنعتی شاهرود، شاهرود ، ایران .۱۳۸۵

۸-ورستیگ و مالالاسکرا(ترجمهٔ محمد حسن شجاعی فرد و مهندس علیرضا نورپور هشترودی)، دینامیک سیالات محاسباتی CFD ، انتشارات دانشگاه علم وصنعت ایران، تهران، ۱۳۷۹

1- A.R. Ansari , Shishkin Meshes and their applications" , Department of Mathematics, Gulf university for science & technology , Hawally 32093 , P.O. Box 7207 , Kuwait.

2- RH. Bartles & GW. Stewart, solution of the matrix equation AX+XB=C [F4], Communications of the ACM, Vol15 , Number 9,1972.

3- S.K. Lele, Compact Finite Difference Scheme with Spectral- Like Resolutions, Journal of Computational physics, 103, 16-42, 1992.

4- M.J. Maghrebi, A study of evolution of intense focal structures in spatially developing three- dimensional planer گردابه, PhD thesis, Department of Mechanical Engineering, Monash University, Melbourne, Australia, 1999.

5- M.J. Maghrebi, Orr Summerfeld Solver Using Mapped Finite Difference Scheme for Plan Wake flow,Department of Mechanical Engineering Shahrood University of Technology, Shahrood ,IR.IRAN. 2004.

6- Bickley, W.G. and Mcnamee .J, Matrix and other direct methods for the solution. of systems of linear difference equations. Philos. *Trans. Roy. Soc.* [London, Ser. A, 232] (1960).

7- Dorr, Fred W. The direct solution of the discrete Poisson equation on a rectangle. *SIAM Rev. 12 (1970).*

8- Martin. R.S. Peters .G. And Wilkinson. J.H. The QR algorithm for real Hessenberg matrices, (Handbook series linear algebra.) *Numer, Math. 14 (1970).*

9- Willkinson. J.H. *The Algebraic Eigenvalue problem. Oxford, 1965.*

10-H.Schlichting, Boundary Layer Thoeory, Mc Graw Hill, 1979.

11-A. Agrawal, A. K. Prasad, Integral Solution for the Mean Flow Profiles of Turbulent Jets,Plumes, and Wakes, Journal of Fluids Engineering,Vol. 125, 813-822, 2003.

12-A. Wary & M.Y. Hussaini, Numerical Experiments in Boundary Layer Stability, Proc. R. Soc. Lond. A, Vol. 392,pp 373-389. 1984.

13-H.Schlichting, Laminar Strahlenausbreitung, ZAMM 13,260-263, 1933.

14-A.R. Ansari, A.F. Hegarty, G.I. shishkin, "Parameter-uniform numerical methods for a laminar jet problem, International Journal for Numerical Methods in Fluids 43(2003) ,937-951.

15-H. Gortler, Berechnung von Aufgaben der Freien Turbulenz auf Grundeines Neaennahe Rungsansatzes, ZAMM 22,244-254,1942.

16-RJ. Grade, Turbulent Fow, New Age International Limited , 2000.

17- Pellerin,S , Dulieu,A. Evaluation of Time and Space Scales in a Soatially Developing 3D Turbulence Incompressible Mixing Layer by Using LES. 2000.

18- J. T. Stuart. On finite amplitude oscillations in laminar mixing layer. Jornal of Fluid Mechanics,29 (3) :417 – 440,1967

Abstract

Numerical solutions have been expanded in difference matters as result of development in computer sciences and coming to existence computers with high speed. Niether numerical solutions don't the laboratory and experimental costs nor gained results with lessdetails .

In this studey the general class of free shear flow been solved by mean of direct numerical simulation. Two dimensional incompressable plan jet flow, mixing layer and wake flow with no forcing were examined. A software was developed which use the method of compact finite difference in streamwise direction (x) and mapped compact finite difference scheme in cross-stream direction. The methods were employed to solve Navier-Stokes eqations in rotational form. Neumann and Dirichlit type boundary conditions were imposed at the inlet and outlet boundary of the computational domain. The resultes of simulations thoroughly verified indicates the high order accuracy of the results. Results also indicate the self similarity, when they were viewed through the self-semilar coordinates.