دانشگاه صنعتی شاهرود

پایان نامه کارشناسی ارشد

مهندسی مکانیک – تبدیل انرژی

عنوان :

مطالعه عملکرد برجهای خنک کن خشک با در نظر گرفتن اثرات باد متقاطع و دمای بالای محیط به روش عددی

استاد راهنما :

دکتر محمد حسن کیهانی

استاد مشاور:

دکتر محمد شاهسون

دانشجو :

علی عباس نژاد

تاییدیه اعضای هیات داوران حاضر در جلسه دفاع

عنوان پایان نامه: مطالعه عملکرد برجهای خنک کن خشک با در نظر گرفتن اثرات باد متقاطع و دمای بالای محیط به روش عددی

دانشجو: علی عباس نژاد

شماره دانشجویی: ۸۲۴۰۱۰۲

جلسه دفاع پایان نامه فوق در تاریخ ۱۳۸۴/۶/۱۵ با حضور اساتید زیر برگزار گردید وبا درجه عالی ارزیابی گردید.

۱	دکتر محمد حسن کیهانی	استاد راهنما
۲	دکتر محمد شاهسون	استاد مشاور
۳	دکتر محمد جواد مغربی	استاد داور
۴	دکتر محمد محسن شاهمردان	استاد داور
۵	دکتر محمود شریعتی	نماینده تحصیلات تکمیلی دانشکده

تقدیم به

پدر و مادر

اسوه های

ایثار و فداکاری

تشکر و قدردانی

پس از حمد، سپاس و ستایش خداوند لازم می دانم تقدیر و تشکر ویژهٔ خود را از اساتید ارجمند جناب آقای دکتر محمد حسن کیهانی و جناب آقای دکتر محمد شاهسون که راهنمائی ها و حمایت‌های ارزنده شان همواره راهگشا و روشنی بخش مسیر پیشبرد اهداف این پایان نامه بود، ابراز دارم.

چکیده

برج های خنک کن خشک یکی از متداول ترین برج ها در نیروگاههای موجود در مناطق کم آب می باشد . یکی از مهمترین عواملی که در رابطه با عملکرد این برج ها مطرح است اثرات شرایط محیطی است . از مؤثرترین این عوامل ، سرعت وزش باد و دمای بالای محیط است که باعث افت عملکرد این برجها می شوند.

در این پایان نامه ابتدا مشکلاتی که نیروگاهها در هنگام وزش باد دچار آن می شوند معرفی می شود. سپس با استفاده از روش تفاضلات محدود پیشرو زمانی و روش بالا دستی برای مشتقات مکانی، معادلات حاکم در حالت جابجایی طبیعی(بدون وزش باد) به صورت عددی حل شده است. در ادامه با توجه به اینکه در هنگام وزش باد مساله به صورت سه بعدی می باشد، معادلات حاکم با استفاده از نرم افزار FLUENT مدلسازی شده و عوامل افت راندمان برجهای خنک کن تحت اثر باد تحلیل می شوند. همچنین عملکرد برجها در دماهای مختلف محیط بررسی شده است.

در انتها برای بهبود عملکرد برجها تحت شرایط باد متقاطع، ایجاد تغییراتی در شکل خارجی برج (استفاده از دیوارهای بادشکن) پیشنهاد شده و اثرات دیوارهای مختلف مورد ارزیابی قرار گرفته است. نتایج بدست آمده نشان می دهد که استفاده از دیوارهای باد شکن در ورودی و خروجی برج باعث کاهش اثر نامطلوب باد می شود.

فهرست مطالب

عنوان **صفحه**

فهرست شکلها

فهرست نشانه ها

c	سرعت صوت در گاز
C_p	ضریب فشار
d	تانسور تغییر شکل
g	شتاب گرانش
I	تانسور یکه
k	ضریب هدایت گرمایی
L	گذر هوای عبوری از واحد سطح مبدلهای حرارتی
M	عدد ماخ
p	فشار
Pr	عدد پرانتل
T	دما
u	مولفه سرعت در امتداد محور افقی
v	مولفه سرعت در امتداد محور عمودی
V	بردار سرعت
ρ	چگالی
β	ضریب انبساط حجمی
μ	لزجت
σ	تانسور تنش
η	مولفه عمودی در فضای محاسباتی
ξ	مولفه افقی ضای محاسباتی
λ	مقدار ویژه
γ	نسبت ظرفیت گرمایی

فصل اول

مقدمه

۱ـ۱ـ فرآیندهای خنک‌کن

بخش قابل توجهی از کاربرد سیستم‌های خنک‌کن در نیروگاه های حرارتی می‌باشد کـه انـرژی غیـر قابل دسترس بخار خروجی از توربین بطور مستقیم یا به واسطهٔ آب به اتمسفر منتقل می‌گردد . در فرآیند خنک کردن منبع دریافت کننده اصلی حرارت ، اتمسفر است که گرما را بطور مستقیم و یا به کمک یـک سیال واسطه دریافت می‌کند . روش‌های انجام فرآیند خنک کردن معمولاً به یکی از سه شکل زیر است :

الف ـ برای دریافت گرما از آب رودخانه ، دریا ، دریاچه ، دریاچه مصنوعی و استخر استفاده می‌شـود. در این تحول بخش قابل توجهی از انتقال حرارت از طریق انتقال جرم انجام می‌شود . در استخر خنک‌کن جهت افزایش نرخ انتقال حرارت از چند فواره مصنوعی برای پاشش آب استفاده می‌کننـد. در اسـتفاده از آب رودخانه‌ها و دریاها جریان آب دارای کمترین درجه حرارت است و از این نظر بهترین انتخاب ممکـن است .

ب ـ فرآیند خنک‌کن تر : در این روش نیز ابتدا جریان آب ، گرما را از سیسـتم دریافـت مـی‌کنـد و سپس در مجاورت با هوا ، گرما را در اثر اختلاف درجه حرارت محسوس و انتقال جرم به هوا پس می‌دهد.

ج ـ فرآیند خنک‌کن خشک : سیال گرم از میان یکسری مبدل حرارتی عبـور مـی‌نمایـد و بـا عبـور جریان هوا به صورت طبیعی یا اجباری ، گرما را بدون واسطه دریافت می‌کند .

۱-۱-۱- سیستم بسته یا فرآیند خنک‌کن خشک

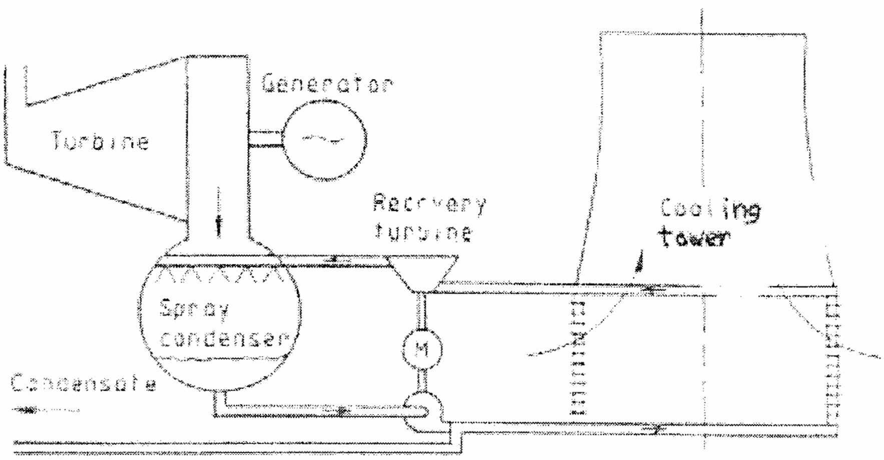
در فرآیند خنک‌کن خشک ، بخار یا آب گرم از درون لوله‌های فین‌دار مـی‌گـذرد و عبـور طبیعـی یـا اجباری جریان هوا از روی این لوله‌ها گرما را بدون واسطه دریافت می‌کند . فرآیند خنک‌کن خشک به دو روش کلی صورت می‌گیرد :

الف ـ برج خشک غیرمستقیم

ب ـ برج خشک مستقیم یا کندانسور هوائی

الف- برج خشک غیرمستقیم

در این روش آبی که از کندانسور خارج می‌شود به طرف برج هدایت شده و از میان یکسری مبدل‌های حرارتی عبور می‌نماید و توسط هوای آزاد یا اجباری (فن) خنک می‌گردد . آب خنک شده از مبدل‌های حرارتی برج خارج شده و به طرف کندانسور می‌رود . معمولاً در برج‌های خشک کندانسور از نوع تماس مستقیم می‌باشد ، در نتیجه آب خنک شده مستقیماً روی بخار خروجی از توربین پاشیده شده و بخار کندانسه می‌شود . شکل(۱-۱) یک نمونه از همین برج‌ها را با جریان هوای آزاد نشان می‌دهد .



شکل(۱-۱): برج خشک غیر مستقیم و کندانسور تماس مستقیم[17]

این سیستم اولین بار بوسیله پرفسور Lazlo Heller از دانشگاه فنی بوداپست مجارستان در سال ۱۹۵۶ میلادی در کنفرانس جهانی نیرو در وین مطرح گردید و بدین جهت از این برج‌ها به نام هلر مشهور

می‌باشند . در این برج‌ها به علت در تماس نبودن هوا و آب ، انتقال حرارت در اثر تبخیر وجود نـدارد ، در نتیجه آب جبرانی کمی نیاز دارد . اما به علت اینکه از طریق اختلاف دما خنک می‌گردد ، به اندازه‌ای کـه با همان شرایط یک برج تر می‌تواند خنک کند ، این سیستم نمی‌تواند این کار را انجام دهد . البته در این روش اگر درجه حرارت بالا باشد با پاشش آب روی مبدل‌های حرارتی درجه حرارت محیط را در اثر تبخیر پائین می‌آورند و در نتیجه می‌توان آب را بهتر خنک کرد . نیروگاه حرارتی اصفهان و شهید رجـائی دارای برج خنک‌کن غیرمستقیم با کندانسور تماس مستقیم می‌باشد . جنس سطوح حرارتـی و فـین‌هـای بکـار رفته در آن باید طوری باشد که انتقال حرارت بخوبی انجام گیرد .

۱-۲-اثر شرایط محیطی

عملکرد تمام مبدلهای حرارتی که با هوا خنک می شوند و برجهای خنک کـن تحـت تـأثیر شـرایط محیطی قرار دارد . تغییرات در درجه حرارت ، رطوبت ، باد ، باران ، برف و تشعشع خورشید ، همگـی بـر عملکرد برجهای خنک کن تاثیر می گذارند که از این بین جز دمای محیط و سرعت باد نقش بقیه عوامل بسیار کم است . در برجهای آلومینیومی جریان داخل برج و حرارت دفع شده به میزان کمی بـه تشعشـع خورشید بستگی دارد . هنگام طراحی برجهای صنعتی بزرگ این شرایط محیطی بایستی در نظـر گرفتـه شوند .

۱-۲-۱-دمای محیط

یکی از عوامل بسیار مهم محیطی که نقش بسیار مهمی در عملکرد برجهای خنک کـن دارد، دمـای محیط می باشد. در طی زمانهایی که درجه حرارت محیط خیلی کم است ، یخ زدن آب داخل رادیاتورهـا مشکلات جدی ایجاد می کند اما ابزار و روشهای مختلفـی از جملـه تعبیـه کرکـره هـایی (Louver) در ورودی برج برای جلوگیری از یخ زدگی رادیاتورها بکار گرفته می شـود. در فصـول گـرم سـال و هنگـام

افزایش درجه حرارت محیط راندمان برجها به مقدار بسیار زیادی افت می کند. جهت رفع این مشکل نیز از خنک کنهای اضطراری (پیک کولر) استفاده می شود.

پیک کولرها برجهای خنک کن کوچکی هستند که در داخل برجهای خنک کن خشک قرار می گیرند، از نوع جریان اجباری بوده و مجهز به سیستم پاشش آب نیز می باشند.

۱-۲-۲-اثر باد بر عملکرد برجهای خنک کن خشک

اندازه گیریهای انجام شده روی برجهای خنک کن طبیعی که تحت وزش باد متقاطع قرار داشته اند نشان داده است که با یک نرخ حرارت دفع شده، دمای آب خروجی از برج افزایش پیدا کرده است . مطالعات روی برجهای خنک کن خشک رفتار مشابهی با آنچه در برجهای تر اتفاق می‌افتد نشان داده است**[17]** . در یک برج خشک ، افزایش دمای آب خروجی یاتغییر در اختلاف بین دمای آب خروجی ودمای هوای ورودی به برج (تغییر در دمای **approach**) به صورت زیر بیان می شود :

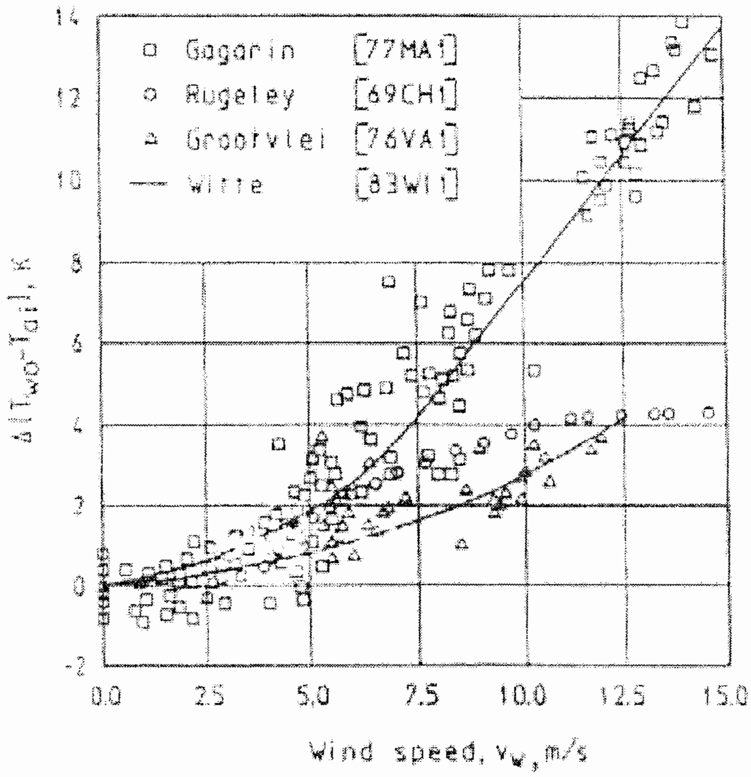
$$\Delta T_{wo} = \Delta(T_{wo} - T_{ai}) = (T_{wo} - T_{ai})_w - (T_{wo} - T_{ai}) \qquad (۱-۱)$$

که ترم اول درسمت راست معادله مربوط می شود به اختلاف دما در حضور باد و ترم دوم اختلاف دما در غیاب باد. پارامترهای T_{wo}, T_{ai} به ترتیب مربوط هستند به دمای هوای ورودی به برج و دمای آب خروجی از کندانسور و ورودی به برج.

بعد از نصب برجهای خنک کن خشک نیروگاههای Ibb-enbaren , Rugeley ، گزارش شد که عملکرد آنها در زمان ورزش باد دچار مشکل شده است . در نیروگاه Gagarin در مجارستان نیز چنین نتایجی گزارش شد.[17]

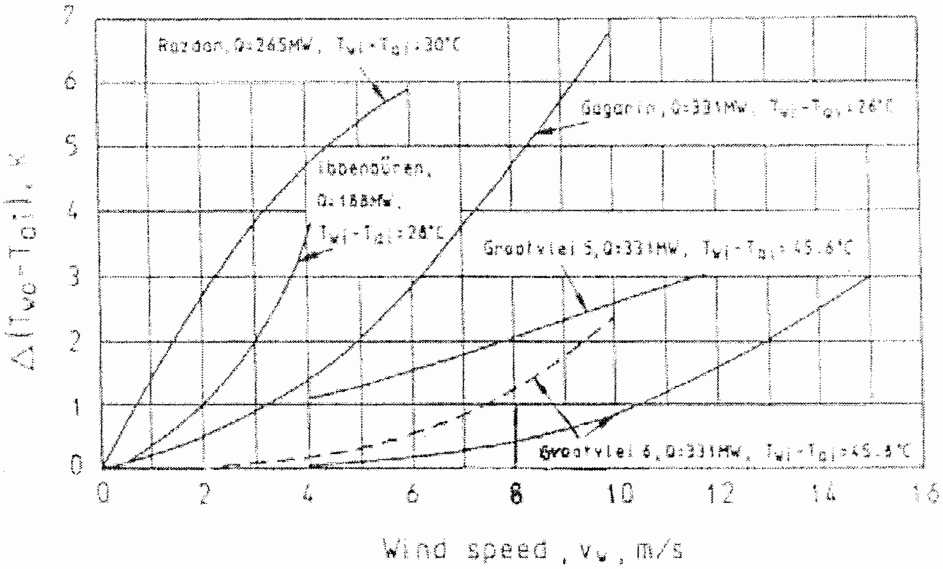
رفتار اطلاعات بدست آمده در نیروگاه Gagarin تا سرعت 6m/s مشابه نتایج نیروگاه Rageley بود . بالاتر از این مقدار سرعت باد ، همانطور که در شکل (۱-۲) نشان داده شده است دو نمودار متفاوت

خواهند شد . انـدازه گیریهـایی کـه توسـط Vander walt و همکـارانش روی بـرج شـماره ۵ نیروگـاه Grootvlei که در آن مبدلهای حرارتی به صورت افقی قرار گرفته اند ، نشان داد که این برج نسبت بـه وزش باد دارای حساسیت کمتری است .



شکل (۱-۲): افزایش دمای آب خروجی از برج[17]

Marjoczy ، اختلاف دمای بسیار زیاد در برجهای این نیروگاه را که باعـث ایجـاد جریـان بهتـر از درون برج می شود را دلیل حساسیت کم آن نسبت به وزش باد دانست. مقادیر افزایش دمای خروجی برج برای چند نیروگاه مختلف در شکل (۱-۳) نشان داده شده است .[17]

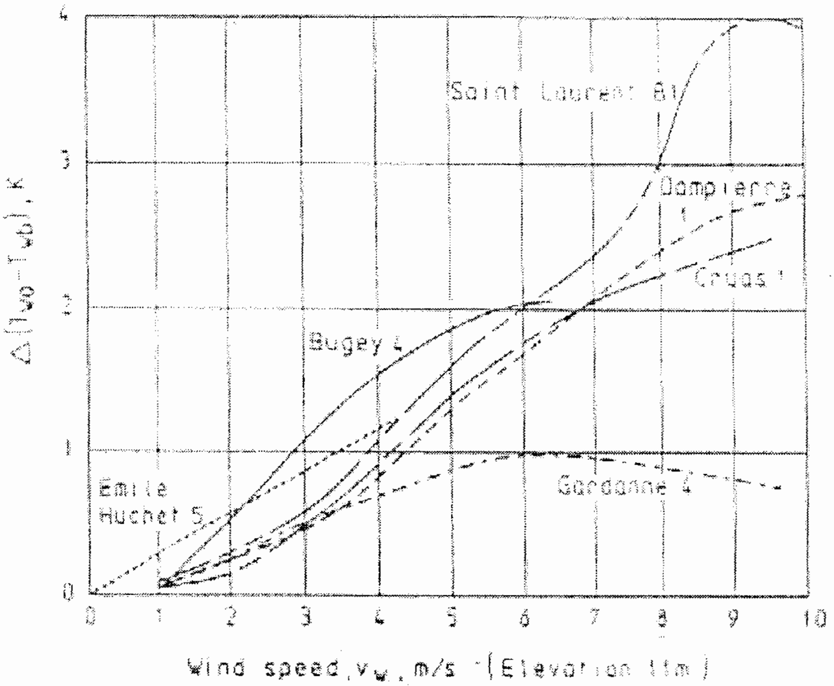


شکل (۱-۳): افزایش دمای آب خروجی از برج برای برجهای مختلف[15],[17]

تستهای عملکردی که روی سه برج خنک کن نیروگاه شهید رجایی انجام شده است ، نشان داد که حساسیت این برجها نسبت به سایر برجهایی که دارای مبدلهای حرارتی عمودی هستند ، کمتر می باشد. دلیل این امر ارتفاع زیاد برج است که باعث ایجاد جریان بهتر و در نتیجه سرعت خروجی زیادتر می شود.

برجهای خنک کن تر نیز در هنگام وزش باد دچار افت عملکرد می شوند . تغییر در اختلاف درجه حرارت بین آب خروجی و دمای حباب تر هوای ورودی ، T_{wb}، برای برجهای تر مختلف در شکل (۱-۴) نشان داده شده است . همه برجها، جریان مخالف هستند به جز Emile Hachet , Saint laurent که جریان عمودی هستند . سرعت باد در ارتفاع ۱۱ متری بالای سطح زمین اندازه گیری شده است .

نتایج بسیار زیادی از تست های مدل برج گزارش شده است اما فقط تعداد محدودی از آنها قادر به پیشگویی اثر باد بر عملکرد برجهای خنک کن بوده اند . معمولاً برای یک مدل در مقیاس کوچک ، ارضا شدن عدد فرود و عدد رینولدز با هم غیر ممکن است ، بنابراین تستهایی که در گذشته انجام شده است یا برپایه تشابه فرود بوده اند و یا تشابه رینولدز که هر یک از روشهای ذکر شده دارای مزایا و معایبی هستند . اندازه گیریهای زیادی بر پایه این دو تشابه صورت گرفته است .



شکل (۱-۴) : افزایش دمای آب در برجهای تر[17]

اگر عدد فرود نادیده گرفته شود ، تستهایی موسوم به تست هم دما (Isothereml) روی مدل انجام می شود . جریان طبیعی در این تستها وجود ندارد و برای ایجاد جریان به طور مصنوعی از یک فن استفاده می شود . عدد رینولدز بدست آمده در این گونه تستها بسیار بزرگتر از مدلهای غیر همدما خواهد بود که عدد رینولدز پایینی دارند .

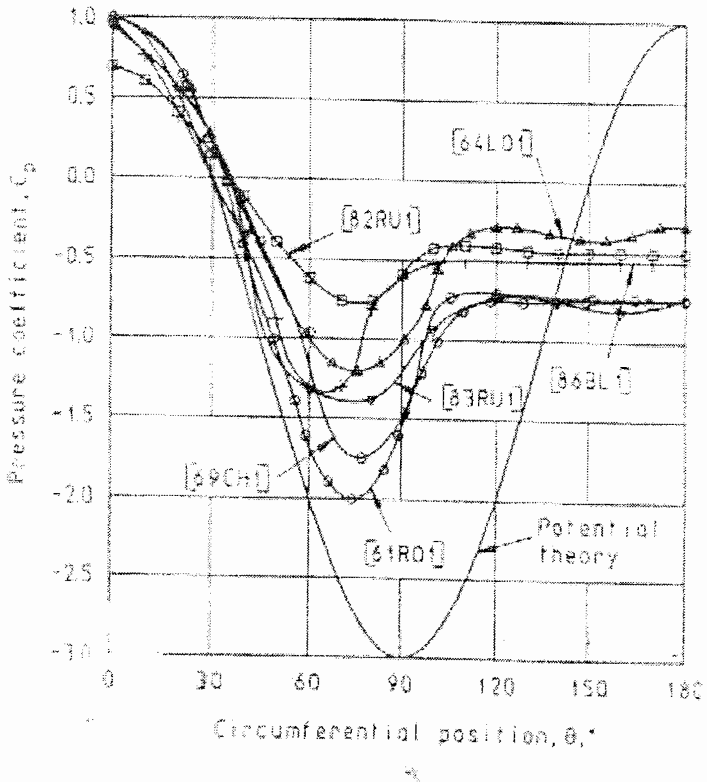
مزیت این روش این است که سرعت سیال داخل برج را بدلیل ایجاد جریان مصنوعی می توان به آسانی اندازه گیری کرد . همچنین نتایج این مدلها دارای دقت و پایداری بسیار خوبی است .

witle پیشنهاد کرد که اثر باد بر عملکرد برج خشک را می توان با بدست آوردن توزیع فشار در جهت پایین برج و در سطح خارجی ، پیش بینی کرد .

وقتی که سیال روی یک استوانه جریان پیدا می کند فشار استاتیکی روی محیط استوانه تغییر می کند . ضریب فشار استاتیک مربوط به صورت زیر است :

$$C_p = (P_\theta - P_\infty)/(\frac{\rho_\infty V_\infty^2}{2}) \qquad (۲-۱)$$

که در آن P_∞ ، فشار استاتیک محلی و P_θمربوط است به شـرایط محـیط در مکـانی دور از اسـتوانه مطابق تئوری جریان پتانسیل می باشند ، توزیع این ضریب در شکل (۱-۵) نشان داده شده است .



شکل (۱-۵): توزیع فشار در ورودی برج[17]

اندازه گیریهای واقعی فشار روی مدلها در قسمت پایین برج نیز در شکل (۱-۵) نشان داده شده اند Ruschewegh , Christopher , Lowe ، توزیع فشار در ورودی یک مدل برج خنک کـن در تونـل باد اندازه گیری کرده اند .

در مدل Ruschewegh ، مبدلهای حرارتی به صورت افقـی قـرار داشـتند و در دو آزمـایش دیگـر آرایش مبدلی به صورت عمودی بوده است . بعضی از اندازه گیریهای انجام شده روی یک بـرج واقعـی در شکل (۱-۵) نشان داده شده است .[17]

برای تحلیل مسأله ، محیط برج به تعداد قسمتهای دلخواه تقسیم می شود . معادلات حرکت و انرژی برای هر قسمت بکار برده می شود و سرانجام سهم همه بخشها در انتقال حرارت برج با هم جمع می شود . در نقطه ای که باد به طور عمود بر مبدلهای حرارتی می وزد (نقطه سکون) ضریب فشار دارای مقدار ماکزیمم است. این مطلب برای برجهایی که مبدلهای حرارتی آنها به صورت عمودی نصب شده صادق می باشد .

Voller , Bunmon تست مدل هم دما را برای مشخص کردن اثرات شکلهای مختلف خروجی برج و آرایش مبدلهای حرارتی بر روی عملکرد برجهای خنک کن که تحت شرایط باد بکار بردند . آنها برای مشاهده اثر باد روی دبی جرمی عبوری از برج ، ضریب فشار برای ورودی و خروجی را بر مبنای اختلاف فشار استاتیک بین گلوگاه برج ومحیط تعریف کردند .[17]

$$C_{pi}, C_{po} = (\Delta P_w - \Delta P)/(\rho_w V_w^2/2) \qquad (۳-۱)$$

که Δp_w، اختلاف فشار استاتیک در حضور باد و Δp اختلاف فشار استاتیک در غیاب باد هستند .

آنها بر پایه این فرض که این پروفیل ، مقدار متوسط یک جریان باد است، یک پروفیل یکنواخت برای باد در نظر گرفتند . آزمایشهای مشابهی توسط Du preez , Kroger برای مقادیر مختلف $\frac{di}{Hi}$ و مقاومتهای مختلف مبدلهای حرارتی ، اثرات پایه های برج و توزیع سرعت باد انجام دادند . آزمایشهای آنها فقط برای برجهای خنک کن با مبدلهای حرارتی افقی انجام شد .

بدلیل اینکه پارامترهای زیادی روی مقدار ضریب فشار ورودی موثر هستند ، پیدا کردن یک رابطه ساده برای C_{P_i} مقدور نمی باشد

همچنین[17] kroger بیان کرده است که با افزایش ارتفاع برجهای خشک کن ، از میزان حساسیت آنها نسبت به سرعت باد کاسته می شود شکل زیر میزان تغییرات دمای approach را برای ارتفاعهای مختلف برج خنک کن نشان می دهد .

فصل دوم

معادلات حاکم در حالت متقارن محوری و انتقال آنها به فضای محاسباتی

در رابطه بالا :

$$n=\begin{cases}0 & ; \quad 2D \;\; Cartesian\\ 1 & ; \quad Axisymmetric\end{cases}$$

۲-۱-۲- بقای مومنتم

شکل معادلات بقای مومنتم در حضور نیروهای سطحی و نیروهای حجمی در حالت دو بعدی و در دستگاه دکارتی در دو جهت $y\,,\,x$ به صورت زیر است :

$$\begin{aligned}&\frac{\partial(\rho u)}{\partial t}+\frac{\partial\left(\rho u^2+p\right)}{\partial x}+\frac{\partial(\rho uv)}{\partial y}=\frac{\partial\tau_{xx}}{\partial x}+\frac{\partial\tau_{xy}}{\partial y}+g_x(\rho-\rho_\infty)\\&\frac{\partial(\rho v)}{\partial t}+\frac{\partial(\rho uv)}{\partial x}+\frac{\partial\left(\rho v^2+p\right)}{\partial y}=\frac{\partial\tau_{yx}}{\partial x}+\frac{\partial\tau_{yy}}{\partial y}+g_y(\rho-\rho_\infty)\end{aligned} \qquad (۴-۲)$$

در حالت تقارن محوری معادلات بقای مومنتم در دستگاه استوانه‌ای به صورت زیر خلاصه می‌شود :

$$\begin{cases}\dfrac{\partial(r\rho v_r)}{\partial t}+\dfrac{\partial\left(r\rho v_r^2+rp\right)}{\partial r}+\dfrac{\partial(r\rho v_r v_z)}{\partial z}=\dfrac{\partial(r\tau_{rr})}{\partial r}+\dfrac{\partial(r\tau_{zr})}{\partial z}+rg_r(\rho-\rho_\infty)+p-\tau_{\theta\theta}\\ \dfrac{\partial(r\rho v_z)}{\partial t}+\dfrac{\partial(r\rho v_r v_z)}{\partial r}+\dfrac{\partial\left(r\rho v_z^2+rp\right)}{\partial z}=\dfrac{\partial(r\tau_{rz})}{\partial r}+\dfrac{\partial(r\tau_{zz})}{\partial z}+rg_z(\rho-\rho_\infty)\end{cases} \qquad (۵-۲)$$

با تبدیل z به x، r به y، v_z به u و v_r به v در روابط (۲-۵) داریم :

$$\begin{cases}\dfrac{\partial(y\rho u)}{\partial t}+\dfrac{\partial(y\rho uv)}{\partial y}+\dfrac{\partial\left(y\rho u^2+yp\right)}{\partial x}=\dfrac{\partial(y\tau_{xy})}{\partial y}+\dfrac{\partial(y\tau_{xx})}{\partial x}+yg_x(\rho-\rho_\infty)\\ \dfrac{\partial(y\rho v)}{\partial t}+\dfrac{\partial\left(y\rho v^2+yp\right)}{\partial y}+\dfrac{\partial(y\rho uv)}{\partial x}=\dfrac{\partial(y\tau_{yy})}{\partial y}+\dfrac{\partial(y\tau_{xy})}{\partial x}+yg_y(\rho-\rho_\infty)+p-\tau_{\theta\theta}\end{cases} \qquad (۶-۲)$$

به طور مشابه می‌توان روابط بقای مومنتم را در دستگاه دکارتی و استوانه‌ای با یک شکل مشترک بیان نمود :

$$\begin{cases}\dfrac{\partial(y^n\rho u)}{\partial t}+\dfrac{\partial(y\rho uv)}{\partial y}+\dfrac{\partial\left(y^n\rho u^2+y^n p\right)}{\partial x}=\dfrac{\partial(y^n\tau_{xy})}{\partial y}+\dfrac{\partial(y^n\tau_{xx})}{\partial x}+y^n g_x(\rho-\rho_\infty)\\ \dfrac{\partial(y^n\rho v)}{\partial t}+\dfrac{\partial\left(y^n\rho v^2+y^n p\right)}{\partial y}+\dfrac{\partial(y^n\rho uv)}{\partial x}=\dfrac{\partial(y^n\tau_{yy})}{\partial y}+\dfrac{\partial(y^n\tau_{xy})}{\partial x}+y^n g_y(\rho-\rho_\infty)+n(p-\tau_{\theta\theta})\end{cases} \qquad (۷-۲)$$

که باز هم در رابطه بالا :

$$n=\begin{cases}0 & ; \quad 2D \;\; Cartesian \\ 1 & ; \quad Axisymmetric\end{cases}$$

۲-۱-۳- بقای انرژی

رابطه بقای انرژی دو بعدی در دستگاه مختصات دکارتی به صورت زیر است :

$$\begin{aligned}&\frac{\partial e}{\partial t}+\frac{\partial}{\partial x}[(e+p)u]+\frac{\partial}{\partial y}[(e+p)v] \\ &=\frac{\partial}{\partial x}\left(u\tau_{xx}+v\tau_{yx}\right)+\frac{\partial}{\partial y}\left(u\tau_{xy}+v\tau_{yy}\right)-\frac{\partial \dot{q}_x}{\partial x}-\frac{\partial \dot{q}_y}{\partial y}+S_h\end{aligned} \tag{۸-۲}$$

رابطه بقای انرژی در حالت تقارن محوری در دستگاه مختصات استوانه‌ای نیز به صورت زیر می‌باشد :

$$\begin{aligned}&\frac{\partial (re)}{\partial t}+\frac{\partial}{\partial r}[r(e+p)v_r]+\frac{\partial}{\partial z}[r(e+p)v_z] \\ &=\frac{\partial}{\partial r}\left(rv_r\tau_{rr}+rv_z\tau_{zr}\right)+\frac{\partial}{\partial z}\left(rv_r\tau_{rz}+rv_z\tau_{zz}\right) \\ &-\frac{\partial (r\dot{q}_r)}{\partial r}-\frac{\partial (r\dot{q}_z)}{\partial z}+S_h\end{aligned} \tag{۹-۲}$$

اگر در معادله بالا x را بجای z، y را بجای r، u را بجای v_z و v را بجای v_r جایگزین می‌توان معادله بقای انرژی را برای دو حالت دو بعدی دکارتی و تقارن محوری استوانه‌ای، به صورت یک شکل واحد بیان نمود :

$$\begin{aligned}&\frac{\partial (y^n e)}{\partial t}+\frac{\partial}{\partial x}\left[y^n(e+p)u\right]+\frac{\partial}{\partial y}\left[y^n(e+p)v\right] \\ &=\frac{\partial}{\partial x}\left[y^n u\tau_{xx}+y^n v\tau_{xy}\right]+\frac{\partial}{\partial y}\left[y^n u\tau_{xy}+y^n v\tau_{yy}\right] \\ &-\frac{\partial}{\partial x}\left(y^n\dot{q}_x\right)-\frac{\partial}{\partial y}\left(y^n\dot{q}_y\right)+S_h\end{aligned} \tag{۱۰-۲}$$

در رابطه بالا :

$$n = \begin{cases} 0 & ; \quad 2D \ \ Cartesian \\ 1 & ; \quad Axisymmetric \end{cases}$$

۲-۲- محاسبه $\dot{\vec{q}}$ با استفاده از قانون هدایت فوریه

برای محاسبه مقادیر $\dot{\vec{q}}$ که در معادلات بقا ظاهر شده اند، می‌توان از قانون هدایت فوریه استفاده کرد.

$$\dot{\vec{q}} = -k\nabla T \tag{۲-۱۱}$$

در دستگاه مختصات دکارتی :

$$\dot{\vec{q}} = -k\left(\hat{i}\,\frac{\partial T}{\partial x} + \hat{j}\,\frac{\partial T}{\partial y} + \hat{k}\,\frac{\partial T}{\partial z} \right) \tag{۲-۱۲}$$

و در دستگاه مختصات استوانه‌ای :

$$\dot{\vec{q}} = -k\left(\hat{i}_r\,\frac{\partial T}{\partial r} + \frac{\hat{i}_\theta}{r}\frac{\partial T}{\partial \theta} + \hat{i}_z\,\frac{\partial T}{\partial z} \right) \tag{۲-۱۳}$$

در حالت دو بعدی در دستگاه دکارتی داریم :

$$\dot{\vec{q}} = -k\left(\hat{i}\,\frac{\partial T}{\partial x} + \hat{j}\,\frac{\partial T}{\partial y} \right) \rightarrow \begin{cases} \dot{q}_x = -k\,\dfrac{\partial T}{\partial x} \\ \dot{q}_y = -k\,\dfrac{\partial T}{\partial y} \end{cases} \tag{۲-۱۴}$$

و برای حالت تقارن محوری($\frac{\partial}{\partial \theta} = 0$) در دستگاه مختصات استوانه‌ای داریم :

$$\dot{\vec{q}} = -k\left(\hat{i}_r\,\frac{\partial T}{\partial r} + \hat{i}_z\,\frac{\partial T}{\partial z} \right) \rightarrow \begin{cases} \dot{q}_r = -k\,\dfrac{\partial T}{\partial r} \\ \dot{q}_z = -k\,\dfrac{\partial T}{\partial z} \end{cases} \tag{۲-۱۵}$$

شکل مشترک برای $\dot{\vec{q}}$ در دستگاه دکارتی در حالت دو بعدی و در دستگاه استوانه‌ای در حالت تقارن محوری، را می‌توان به صورت زیر بیان نمود :

$$\begin{cases} \dot{q}_x = -k\dfrac{\partial T}{\partial x} \\ \dot{q}_y = -k\dfrac{\partial T}{\partial y} \end{cases} \qquad (۱۶-۲)$$

برای مختصات استوانه‌ای x بجای z و y بجای r جایگزین شده است.

۲-۳- تانسور تنش برای سیال نیوتنی

برای یک سیال نیوتنی داریم :

$$\vec{\vec{\tau}} = 2\mu\vec{\vec{d}} + \lambda\left(\nabla.\vec{V}\right)\vec{\vec{I}} \qquad (۱۷-۲)$$

که در رابطه بالا، $\vec{\vec{d}}$ تانسور تغییر شکل و $\vec{\vec{I}}$ تانسور یکه می‌باشد.

طبق فرض استوکس :

$$\begin{aligned} \lambda &= -\frac{2}{3}\mu \\ \vec{\vec{d}} &= \frac{1}{2}\left(\nabla\vec{V} + \nabla\vec{V}^{+}\right) \end{aligned} \qquad (۱۸-۲)$$

که در آن $\nabla\vec{V}^{+}$ ترانسپوز $\nabla\vec{V}$ می‌باشد.

بنا بر روابط بالا تنشهای دو بعدی در دستگاه دکارتی عبارتند از :

$$\tau_{xx} = \left(\lambda + 2\mu\right)\frac{\partial u}{\partial x} + \lambda\frac{\partial v}{\partial y} \qquad (۱۹-۲)$$

$$\tau_{xy} = \tau_{yx} = \mu\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) \qquad (۲۰-۲)$$

$$\tau_{yy} = \left(\lambda + 2\mu\right)\frac{\partial v}{\partial y} + \lambda\frac{\partial u}{\partial x} \qquad (۲۱-۲)$$

$$\tau_{zz} = \lambda\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) \qquad (۲۲-۲)$$

در حالت تقارن محوری در دستگاه استوانه‌ای نیز تنشها عبارتند از :

$$\tau_{rr} = (\lambda + 2\mu)\frac{\partial v_r}{\partial r} + \lambda\left(\frac{\partial v_z}{\partial z} + \frac{v_r}{r}\right) \qquad (۲۳-۲)$$

$$\tau_{rz} = \tau_{zr} = \mu\left(\frac{\partial v_r}{\partial z} + \frac{\partial v_z}{\partial r}\right) \qquad (۲۴-۲)$$

$$\tau_{\theta\theta} = (\lambda + 2\mu)\frac{v_r}{r} + \lambda\left(\frac{\partial v_r}{\partial r} + \frac{\partial v_z}{\partial z}\right) \qquad (۲۵-۲)$$

$$\tau_{zz} = (\lambda + 2\mu)\frac{\partial v_z}{\partial z} + \lambda\left(\frac{\partial v_r}{\partial r} + \frac{v_r}{r}\right) \qquad (۲۶-۲)$$

با نامگذاری y بجای r، x بجای z، v بجای v_r و u بجای v_z در تنشها خواهیم داشت :

$$\tau_{yy} = (\lambda + 2\mu)\frac{\partial v}{\partial y} + \lambda\left(\frac{\partial u}{\partial x} + \frac{v}{y}\right) \qquad (۲۷-۲)$$

$$\tau_{yx} = \tau_{xy} = \mu\left(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}\right) \qquad (۲۸-۲)$$

$$\tau_{\theta\theta} = (\lambda + 2\mu)\frac{v}{y} + \lambda\left(\frac{\partial v}{\partial y} + \frac{\partial u}{\partial x}\right) \qquad (۲۹-۲)$$

$$\tau_{xx} = (\lambda + 2\mu)\frac{\partial u}{\partial x} + \lambda\left(\frac{\partial v}{\partial y} + \frac{v}{y}\right) \qquad (۳۰-۲)$$

از مقایسه تنشهای دو بعدی دکارتی و تقارن محوری می‌توان شکل مشترک تنشها در هر دو دستگاه را به صورت زیر نوشت :

$$\tau_{xx} = (\lambda + 2\mu)\frac{\partial u}{\partial x} + \lambda\left(\frac{\partial v}{\partial y} + n\frac{v}{y}\right) \qquad (۳۱-۲)$$

$$\tau_{xy} = \tau_{yx} = \mu\left(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}\right) \qquad (۳۲-۲)$$

$$\tau_{yy} = (\lambda + 2\mu)\frac{\partial v}{\partial y} + \lambda\left(\frac{\partial u}{\partial x} + n\frac{v}{y}\right) \qquad (۲-۳۳)$$

$$\tau_{\theta\theta} = (\lambda + 2\mu)\frac{v}{y} + \lambda\left(\frac{\partial v}{\partial y} + \frac{\partial u}{\partial x}\right) \qquad (۲-۳۴)$$

$$\tau_{zz} = \lambda\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) \qquad (۲-۳۵)$$

در این روابط :

$$n = \begin{cases} 0 \quad ; \quad 2D \;\; Cartesian \\ 1 \quad ; \quad Axisymmetric \end{cases}$$

در رابطه (۲-۳۴)، تنش $\tau_{\theta\theta}$ مربوط به حالت تقارن محوری می‌باشد.

بجز معادلات بقای جرم، مومنتم و انرژی، معادله دیگری برای گازهای کامل وجود دارد که طبق این معادله فشار p به e, v, u, ρ مربوط می‌شود :

$$p = (\gamma - 1)e - \frac{\gamma - 1}{2}\rho\left(u^2 + v^2\right) \qquad (۲-۳۶)$$

معادلات حاکم بر جریان را که شامل بقای جرم، مومنتم و انرژی می‌باشد را می‌توان به شکل ماتریسی بیان نمود :

$$\frac{\partial \vec{Q}}{\partial t} + \frac{\partial\left(\vec{E}_i - \vec{E}_v\right)}{\partial x} + \frac{\partial\left(\vec{F}_i - \vec{F}_v\right)}{\partial y} = \vec{H} \qquad (۲-۳۷)$$

که در این رابطه :

$$\vec{Q} = \begin{bmatrix} y^n \rho \\ y^n \rho u \\ y^n \rho v \\ y^n e \end{bmatrix} \quad , \quad \vec{E}_i = \begin{bmatrix} y^n \rho u \\ y^n\left(\rho u^2 + p\right) \\ y^n \rho u v \\ y^n (e + p) u \end{bmatrix} \quad , \quad \vec{F} = \begin{bmatrix} y^n \rho v \\ y^n \rho u v \\ y^n\left(\rho v^2 + p\right) \\ y^n v (e + p) v \end{bmatrix}$$

$$\vec{E}_v = \begin{bmatrix} 0 \\ y^n \tau_{xx} \\ y^n \tau_{xy} \\ y^n \left(u\tau_{xx} + v\tau_{xy} - \dot{q}_x \right) \end{bmatrix} \quad , \quad \vec{F}_v = \begin{bmatrix} 0 \\ y^n \tau_{xy} \\ y^n \tau_{yy} \\ y^n \left(u\tau_{xy} + v\tau_{yy} - \dot{q}_y \right) \end{bmatrix}$$

$$\vec{H} = \begin{bmatrix} 0 \\ y^n g_x (\rho - \rho_\infty) \\ y^n g_y (\rho - \rho_\infty) + n(p - \tau_{\theta\theta}) \\ S_h \end{bmatrix}$$

در روابط فوق، اندیس i مربوط به ترم‌های غیر لزج و اندیس v مربوط به ترم‌های لزج می‌باشد.

۲-۴- انتقال معادلات حاکم به فضای محاسباتی

به منظور حل معادلات حاکم بر جریان، مشتقات پاره‌ای موجود در معادلات باید به صورت عبارتهای تفاضل محدود تقریب زده شوند. به این ترتیب معادلات دیفرانسیل پاره‌ای به معادلات جبری تبدیل می‌شوند و این معادلات بر روی شبکه ایجاد شده در قلمرو مورد نظر حل می‌شوند. برای افزایش راندمان و دقت یک روش عددی و اعمال ساده تر شرایط مرزی، تبدیل از فضای فیزیکی به فضای محاسباتی انجام می‌شود. این تبدیل فشرده سازی نقاط شبکه را در مناطقی که گرادیان شدید باشد، فراهم می‌سازد. قلمرو محاسباتی مورد نظر به شکل یک مستطیل با فواصل یکنواخت می‌باشد. برای حل معادلات حاکم بر جریان در فضای محاسباتی، تبدیل این معادلات از فضای فیزیکی به فضای محاسباتی اجتناب ناپذیر است. در این بخش انتقال معادلات حاکم بر جریان سیال (معادلات ناویر - استوکس) در دستگاه مختصات دکارتی یا استوانه‌ای در شکل مشترک آن، از فضای فیزیکی به فضای محاسباتی مورد بررسی قرار می‌گیرد.

۲-۴-۱- متریکها و ژاکوبین‌های تبدیل

اگر مولفه‌های مختصات در فضای فیزیکی را با y, x و مولفه‌های مختصات در فضای محاسباتی را با η, ξ نشان دهیم، آنگاه :

$$\begin{cases}(x, y) = (x(\xi,\eta), y(\xi,\eta)) \\ (\xi,\eta) = (\xi(x, y), \eta(x, y))\end{cases} \tag{۳۸-۲}$$

معادلات حاکم بر جریان در فضای فیزیکی عبارتند از :

$$\vec{Q}_t + \vec{E}_x + \vec{F}_y = \vec{H} \tag{۳۹-۲}$$

که در آن :

$$\vec{E} = \vec{E}_i + \vec{E}_v \quad , \quad \vec{F} = \vec{F}_i + \vec{F}_v \tag{۴۰-۲}$$

طبق تعریف مشتق زنجیره‌ای :

$$\begin{cases}\partial_x = \xi_x \partial_\xi + \eta_x \partial_\eta \\ \partial_y = \xi_y \partial_\xi + \eta_y \partial_\eta\end{cases} \tag{۴۱-۲}$$

در رابطه بالا جمله‌های η_x, ξ_y, ξ_x و η_y متریکهای تبدیل و یا به صورت ساده تر متریک نامیده میشوند. بنا بر این معادلات حاکم به صورت زیر نوشته می‌شوند :

$$\vec{Q}_t + \xi_x \vec{E}_\xi + \eta_x \vec{E}_\eta + \xi_y \vec{F}_\xi + \eta_y \vec{F}_\eta = \vec{H} \tag{۴۲-۲}$$

ماتریس ژاکوبین تبدیل $\widetilde{J}$ و معکوس تبدیل $\widetilde{J}^{-1}$ را به صورت زیر تعریف می‌کنیم :

$$\widetilde{J} \equiv \frac{\partial(x, y)}{\partial(\xi,\eta)} = \begin{bmatrix} x_\xi & x_\eta \\ y_\xi & y_\eta \end{bmatrix} \quad , \quad \widetilde{J}^{-1} \equiv \frac{\partial(\xi,\eta)}{\partial(x, y)} = \begin{bmatrix} \xi_x & \xi_y \\ \eta_x & \eta_y \end{bmatrix} \tag{۴۳-۲}$$

در ریاضیات ثابت می‌شود که معکوس یک ماتریس از رابطه زیر محاسبه می‌شود :

$$\widetilde{A}^{-1} = \frac{1}{\left|\widetilde{A}\right|} adj\left(\widetilde{A}\right) \tag{۴۴-۲}$$

بنابر این می‌توان عناصر $\tilde{J}$ و $\tilde{J}^{-1}$ را به هم مربوط ساخت. دترمینان J را بصورت زیر تعریف می‌کنیم:

$$J \equiv \left|\tilde{J}\right| = \begin{vmatrix} x_\xi & x_\eta \\ y_\xi & y_\eta \end{vmatrix} = x_\xi y_\eta - x_\eta y_\xi \qquad (۲-۴۵)$$

اگر $\tilde{J}^{-1}$ را حساب کنیم داریم :

$$\tilde{J}^{-1} = \frac{1}{J}\begin{bmatrix} y_\eta & -x_\eta \\ -y_\xi & x_\xi \end{bmatrix} = \begin{bmatrix} \xi_x & \xi_y \\ \eta_x & \eta_y \end{bmatrix} \qquad (۲-۴۶)$$

با مقایسه اعضای ماتریسها می‌توان نتیجه گرفت :

$$\xi_x = \frac{y_\eta}{J} \quad , \quad \xi_y = \frac{-x_\eta}{J} \quad , \quad \eta_x = \frac{-y_\xi}{J} \quad , \quad \eta_y = \frac{x_\xi}{J} \qquad (۲-۴۷)$$

۲-۴-۲- انتقال معادلات حاکم بر جریان به فضای محاسباتی

اگر جملات محاسبه شده در رابطه (۲-۱۰) را در رابطه (۲-۵) قرار دهیم خواهیم داشت :

$$\vec{Q}_t + \frac{y_\eta}{J}\vec{E}_\xi - \frac{y_\xi}{J}\vec{E}_\eta - \frac{x_\eta}{J}\vec{F}_\xi + \frac{x_\xi}{J}\vec{F}_\eta = \vec{H} \qquad (۲-۴۸)$$

طرفین رابطه فوق را در J ضرب می‌کنیم. با توجه به مستقل بودن J از t می‌توان نوشت :

$$\left(J\vec{Q}\right)_t + y_\eta\vec{E}_\xi - y_\xi\vec{E}_\eta - x_\eta\vec{F}_\xi + x_\xi\vec{F}_\eta = J\vec{H} \qquad (۲-۴۹)$$

اگر دقت شود، ملاحظه می‌شود که می‌توان معادله فوق را بصورت بقایی نوشت :

$$\left(J\vec{Q}\right)_t + \left(y_\eta\vec{E}\right)_\xi - \left(y_\xi\vec{E}\right)_\eta - \left(x_\eta\vec{F}\right)_\xi + \left(x_\xi\vec{F}\right)_\eta = J\vec{H} \qquad (۲-۵۰)$$

و یا :

$$\left(J\vec{Q}\right)_t + \left(J\xi_x\vec{E}\right)_\xi + \left(J\eta_x\vec{E}\right)_\eta + \left(J\xi_y\vec{F}\right)_\xi + \left(J\eta_y\vec{F}\right)_\eta = J\vec{H} \qquad (۲-۵۱)$$

پس از فاکتور گیری داریم :

$$\left(J\vec{Q}\right)_t + \left[J\left(\xi_x\vec{E}+\xi_y\vec{F}\right)\right]_\xi + \left[J\left(\eta_x\vec{E}+\eta_y\vec{F}\right)\right]_\eta = J\vec{H} \tag{۵۲-۲}$$

حال تعریف می‌کنیم :

$$\hat{Q} \equiv J\vec{Q} \tag{۵۳-۲}$$

$$\hat{E} \equiv J\left(\xi_x\vec{E}+\xi_y\vec{F}\right) \tag{۵۴-۲}$$

$$\hat{F} \equiv J\left(\eta_x\vec{E}+\eta_y\vec{F}\right) \tag{۵۵-۲}$$

$$\hat{H} \equiv J\vec{H} \tag{۵۶-۲}$$

با تعاریف فوق معادلات بقا در فضای محاسباتی به صورت زیر در می‌آید :

$$\hat{Q}_t + \hat{E}_\xi + \hat{F}_\eta = \hat{H} \tag{۵۷-۲}$$

که در این رابطه :

$$\begin{cases} \hat{E} = \hat{E}_i + \hat{E}_v \\ \hat{F} = \hat{F}_i + \hat{F}_v \end{cases} \tag{۵۸-۲}$$

۲-۴-۳- محاسبه جملات معادلات بقا در فضای محاسباتی

۲-۴-۳-۱- محاسبه $\hat{E}_i$ و $\hat{F}_i$

$$\hat{E}_i = J\left(\xi_x\vec{E}_i+\xi_y\vec{F}_i\right) = Jy^n\begin{bmatrix} \xi_x\rho u \\ \xi_x\left(\rho u^2+p\right) \\ \xi_x\rho uv \\ \xi_x(e+p)u \end{bmatrix} + Jy^n\begin{bmatrix} \xi_y\rho v \\ \xi_y\rho uv \\ \xi_y\left(\rho v^2+p\right) \\ \xi_y(e+p)v \end{bmatrix} \tag{۵۹-۲}$$

$$\hat{E}_i = Jy^n\begin{bmatrix} \rho\left(\xi_x u+\xi_y v\right) \\ \rho u\left(\xi_x u+\xi_y v\right)+\xi_x p \\ \rho v\left(\xi_x u+\xi_y v\right)+\xi_y p \\ (e+p)\left(\xi_x u+\xi_y v\right) \end{bmatrix} \tag{۶۰-۲}$$

برای راحتی کار U را به صورت زیر تعریف می‌کنیم :

$$U = \left(\xi_x u + \xi_y v\right) \tag{۶۱-۲}$$

بنا بر این داریم :

$$\hat{E}_i = \begin{bmatrix} \rho U \\ \rho u U + \xi_x p \\ \rho v U + \xi_y p \\ (e+p)U \end{bmatrix} \tag{۶۲-۲}$$

به همین ترتیب به محاسبه $\hat{F}_i$ می‌پردازیم :

$$\hat{F}_i = J\left(\eta_x \vec{E}_i + \eta_y \vec{F}_i\right) = Jy^n \begin{bmatrix} \eta_x \rho u \\ \eta_x\left(\rho u^2 + p\right) \\ \eta_x \rho u v \\ \eta_x (e+p)u \end{bmatrix} + Jy^n \begin{bmatrix} \eta_y \rho v \\ \eta_y \rho u v \\ \eta_y\left(\rho v^2 + p\right) \\ \eta_y (e+p)v \end{bmatrix} \tag{۶۳-۲}$$

$$F_i = Jy^n \begin{bmatrix} \rho\left(\eta_x u + \eta_y v\right) \\ \rho u\left(\eta_x u + \eta_y v\right) + \eta_x p \\ \rho v\left(\eta_x u + \eta_y v\right) + \eta_y p \\ (e+p)\left(\eta_x u + \eta_y v\right) \end{bmatrix} \tag{۶۴-۲}$$

حال V را به صورت زیر تعریف می‌کنیم :

$$V \equiv Jy^n\left(\eta_x u + \eta_y v\right) \tag{۶۵-۲}$$

بنابر این $\hat{F}_i$ خواهد شد :

$$\hat{F}_i = Jy^n \begin{bmatrix} \rho V \\ \rho u V + \eta_x p \\ \rho v V + \eta_y p \\ (e+p)V \end{bmatrix} \tag{۶۶-۲}$$

۲-۴-۳-۲- محاسبه تنشها در فضای محاسباتی

در قسمت قبل شکل مشترک تنشها در مختصات دکارتی و مختصات استوانه‌ای به صورت زیر بدست آمد :

$$\tau_{xx} = \frac{4}{3}\mu u_x - \frac{2}{3}\mu v_y - \frac{2}{3}\mu n\frac{v}{y} \qquad (۶۷-۲)$$

$$\tau_{xy} = \mu u_y + \mu v_x \qquad (۶۸-۲)$$

$$\tau_{yy} = \frac{4}{3}\mu v_y - \frac{2}{3}\mu v_x - \frac{2}{3}\mu n\frac{v}{y} \qquad (۶۹-۲)$$

$$\tau_{\theta\theta} = \frac{4}{3}\mu\frac{v}{y} - \frac{2}{3}\mu\left(u_x + v_y\right) \qquad (۷۰-۲)$$

$$\tau_{zz} = \frac{2}{3}\mu\left(u_x + v_y\right) \qquad (۷۱-۲)$$

با استفاده از قانون مشتق زنجیره‌ای مشتقات سرعتها نسبت به y,x عبارتند از :

$$u_x = \xi_x u_\xi + \eta_x u_\eta \qquad (۷۲-۲)$$

$$v_x = \xi_x v_\xi + \eta_x v_\eta \qquad (۷۳-۲)$$

$$u_y = \xi_y u_\xi + \eta_y u_\eta \qquad (۷۴-۲)$$

$$v_y = \xi_y v_\eta + \eta_y v_\eta \qquad (۷۵-۲)$$

با قرار دادن مقادیر مشتقات سرعت در روابط تنش داریم :

$$\tau_{xx} = \frac{4}{3}\mu\xi_x u_\xi + \frac{4}{3}\mu\eta_x u_\eta - \frac{2}{3}\mu\xi_y v_\xi - \frac{2}{3}\mu\eta_y v_\eta - \frac{2}{3}\mu n\frac{v}{y} \qquad (۷۶-۲)$$

$$\tau_{xy} = \mu\xi_x u_\xi + \mu\eta_y u_\eta + \mu\xi_x v_\xi + \mu\eta_x v_\eta \qquad (۷۷-۲)$$

$$\tau_{yy} = \frac{4}{3}\mu\xi_x v_\xi + \frac{4}{3}\mu\eta_y v_\eta - \frac{2}{3}\mu\xi_x u_\xi - \frac{2}{3}\varpi\eta_x u_\eta - \frac{2}{3}\mu n\frac{v}{y} \qquad (۷۸-۲)$$

$$\tau_{\theta\theta} = \frac{2}{3}\mu\left(2\frac{v}{y} - \xi_x u_\xi - \eta_x u_\eta - \xi_y v_\xi - \eta_y v_\eta\right) \qquad (۷۹-۲)$$

$$\tau_{zz} = \frac{2}{3}\mu\left(\xi_x u_\xi + \eta_x u_\eta + \xi_y v_\xi + \eta_y v_\eta\right) \qquad (۸۰-۲)$$

۲-۴-۳-۳- محاسبه $\dot{q}_x$, $\dot{q}_y$ در فضای محاسباتی

$$\begin{cases} \dot{q}_x = -kT_x = -k\left(\xi_x T_\xi + \eta_x T_\eta\right) = -\left(k\xi_x\right)T_\xi - \left(k\eta_x\right)T_\eta \\ \dot{q}_y = -kT_y = -k\left(\xi_y T_\xi + \eta_y T_\eta\right) = -\left(k\xi_y\right)T_\xi - \left(k\eta_y\right)T_\eta \end{cases} \qquad (۸۱-۲)$$

۲-۴-۳-۴- محاسبه $\hat{E}_v$

$$\hat{E}_v = J\left(\xi_x \vec{E}_v + \eta_x \vec{F}_v\right) \qquad (۸۲-۲)$$

با قرار دادن $\vec{E}_v$ و $\vec{F}_v$ در رابطه اخیر داریم :

$$\hat{E}_v = Jy^n \begin{bmatrix} 0 \\ \xi_x\tau_{xx} \\ \xi_x\tau_{xy} \\ \xi_x\left(u\tau_{xx} + v\tau_{xy} - \dot{q}_x\right) \end{bmatrix} + Jy^n \begin{bmatrix} 0 \\ \xi_y\tau_{xy} \\ \xi_y\tau_{yy} \\ \xi_y\left(u\tau_{xy} + v\tau_{yy} - \dot{q}_y\right) \end{bmatrix} \qquad (۸۳-۲)$$

$\therefore$

$$\hat{E}_v = Jy^n \begin{bmatrix} 0 \\ \xi_x\tau_{xx} + \xi_y\tau_{xy} \\ \xi_x\tau_{xy} + \xi_y\tau_{yy} \\ \xi_x\left(u\tau_{xx} + v\tau_{xy} - \dot{q}_x\right) + \xi_y\left(u\tau_{xy} + v\tau_{yy} - \dot{q}_y\right) \end{bmatrix} \qquad (۸۴-۲)$$

با قرار دادن مقادیر تنشها که قبلاً محاسبه شده‌اند، داریم :

$$\hat{E}_v = Jy^n \begin{bmatrix} row1 \\ row2 \\ row3 \\ row4 \end{bmatrix} \tag{۸۵-۲}$$

که در این رابطه :

$$row1 = 0$$

$$row2 = \left(\frac{4}{3}\xi_x^2 + \xi_y^2\right)\mu u_\xi + \left(\frac{4}{3}\xi_x\eta_x + \xi_y\eta_y\right)\mu u_\eta + \left(\frac{1}{3}\xi_x\xi_y\right)\mu v_\xi + \left(\xi_y\eta_x - \frac{2}{3}\xi_x\eta_y\right)\mu v_\eta - \left(\frac{2}{3}\xi_x\frac{n}{y}\mu v\right)$$

$$row3 = \left(\frac{1}{3}\xi_x\xi_y\right)\mu u_\xi + \left(\xi_x\eta_y - \frac{2}{3}\xi_y\eta_x\right)\mu u_\eta + \left(\xi_x^2 + \frac{4}{3}\xi_y^2\right)\mu v_\xi + \left(\xi_x\eta_x + \frac{4}{3}\xi_y\eta_y\right)\mu v_\eta - \left(\frac{2}{3}\xi_y\frac{n}{y}\mu v\right)$$

$$row4 = k\left(\xi_x^2 + \xi_y^2\right)T_\xi + k\left(\xi_x\eta_x + \xi_y\eta_y\right)T_\eta + u\times(row2) + v\times(row3)$$

برای ساده تر شدن رابطه فوق ضرایب a_1 تا a_{11} را به صورت زیر تعریف می‌کنیم :

$$a_1 = \left(\frac{4}{3}\xi_x^2 + \xi_y^2\right)\mu \tag{۸۶-۲}$$

$$a_2 = \left(\frac{1}{3}\xi_x\xi_y\mu\right) \tag{۸۷-۲}$$

$$a_3 = \left(\xi_x^2 + \frac{4}{3}\xi_y^2\right)\mu \tag{۸۸-۲}$$

$$a_4 = k\left(\xi_x^2 + \xi_y^2\right) \tag{۸۹-۲}$$

$$a_5 = \left(\frac{4}{3}\xi_x\eta_x + \xi_y\eta_y\right)\mu \tag{۹۰-۲}$$

$$a_6 = \left(\xi_y\eta_x - \frac{2}{3}\xi_x\eta_y\right)\mu \tag{۹۱-۲}$$

$$a_7 = -\left(\frac{2}{3}\xi_x \frac{n}{y}\right)\mu \qquad (۹۲-۲)$$

$$a_8 = \left(\xi_x\eta_y - \frac{2}{3}\xi_y\eta_x\right)\mu \qquad (۹۳-۲)$$

$$a_9 = \left(\xi_x\eta_x + \frac{4}{3}\xi_y\eta_y\right)\mu \qquad (۹۴-۲)$$

$$a_{10} = -\frac{2}{3}\xi_y \frac{n}{y}\mu \qquad (۹۵-۲)$$

$$a_{11} = k\left(\xi_x\eta_x + \xi_y\eta_y\right) \qquad (۹۶-۲)$$

حال می‌توان $\hat{E}_v$ را به صورت زیر بازنویسی نمود :

$$\hat{E}_v = Jy^n \begin{bmatrix} 0 \\ \left(a_1u_\xi + a_5u_\eta + a_2v_\xi + a_6v_\xi + a_7v\right) \equiv row2 \\ \left(a_2u_\xi + a_6u_\eta + a_3v_\xi + a_9v_\eta + a_{10}v\right) \equiv row3 \\ a_4T_\xi + a_{11}T_\eta + u\times(row2) + v\times(row3) \end{bmatrix} \qquad (۹۷-۲)$$

۲-۴-۳-۵- محاسبه $\hat{F}_v$

$$\hat{F}_v = J\left(\eta_x\vec{E}_v + \eta_y\vec{F}_v\right) \qquad (۹۸-۲)$$

با قرار دادن مقادیر $\vec{E}_v, \vec{F}_v$ در رابطه اخیر داریم :

$$\hat{F}_v = Jy^n \begin{bmatrix} 0 \\ \eta_x\tau_{xx} \\ \eta_x\tau_{xy} \\ \eta_x\left(u\tau_{xx} + v\tau_{xy} - \dot{q}_x\right) \end{bmatrix} + Jy^n \begin{bmatrix} 0 \\ \eta_y\tau_{xy} \\ \eta_y\tau_{yy} \\ \eta_y\left(u\tau_{xy} + v\tau_{yy} - \dot{q}_y\right) \end{bmatrix} \qquad (۹۹-۲)$$

$$\hat{F}_v = Jy^n \begin{bmatrix} 0 \\ \eta_x \tau_{xx} + \eta_y \tau_{xy} \\ \eta_x \tau_{xy} + \eta_y \tau_{yy} \\ u(\eta_x \tau_{xx} + \eta_y \tau_{xy}) + v(\eta_x \tau_{xy} + \eta_y \tau_{yy}) - \eta_x \dot{q}_x - \eta_y \dot{q}_y \end{bmatrix} \tag{۱۰۰-۲}$$

با قرار دادن مقادیر تنشها که قبلاً محاسبه شده‌اند در رابطه بالا داریم :

$$\hat{F}_v = Jy^n \begin{bmatrix} row1 \\ row1 \\ row3 \\ row4 \end{bmatrix} \tag{۱۰۱-۲}$$

که در رابطه فوق :

$$row1 = 0$$

$$row2 = \left(\frac{4}{3}\xi_x\eta_x + \xi_y\eta_y\right)\mu u_\xi + \left(\frac{4}{3}\eta_x^2 + \eta_y^2\right)\mu u_\eta + \left(\xi_x\eta_y - \frac{2}{3}\xi_y\eta_x\right)\mu v_\xi + \left(\frac{1}{3}\eta_x\eta_y\right)\mu v_\eta - \left(\frac{2}{3}\eta_x\frac{n}{y}\mu v\right)$$

$$row3 = \left(\xi_y\eta_x - \frac{2}{3}\xi_x\eta_y\right)\mu u_\xi + \left(\frac{1}{3}\eta_x\eta_y\right)\mu u_\eta + \left(\xi_x\eta_x + \frac{4}{3}\xi_y\eta_y\right)\mu v_\xi + \left(\eta_x^2 + \frac{4}{3}\eta_y^2\right)\mu v_\eta - \left(\frac{2}{3}\eta_y\frac{n}{y}\mu v\right)$$

$$row4 = k(\xi_x\eta_x + \xi_y\eta_y)T_\xi + k(\eta_x^2 + \eta_y^2)T_\eta + u\times(row2) + v\times(row3)$$

باز برای ساده تر شدن رابطه (۲-۶۳) ضرایب b_1 تا b_{11} را به صورت زیر تعریف می‌کنیم :

$$b_1 = \left(\frac{4}{3}\eta_x^2 + \eta_y^2\right)\mu \tag{۱۰۲-۲}$$

$$b_2 = \left(\frac{1}{3}\eta_x\eta_y\right)\mu \tag{۱۰۳-۲}$$

$$b_3 = \left(\eta_x^2 + \frac{4}{3}\eta_y^2\right)\mu \tag{۱۰۴-۲}$$

$$b_4 = k(\eta_x^2 + \eta_y^2) \tag{۱۰۵-۲}$$

$$b_5 = \left(\frac{4}{3}\xi_x\eta_x + \xi_y\eta_y\right)\mu \quad (۱۰۶-۲)$$

$$b_6 = \left(\xi_x\eta_y - \frac{2}{3}\xi_y\eta_x\right)\mu \quad (۱۰۷-۲)$$

$$b_7 = -\left(\frac{2}{3}\eta_x\frac{n}{y}\right)\mu \quad (۱۰۸-۲)$$

$$b_8 = \left(\xi_y\eta_x - \frac{2}{3}\xi_x\eta_y\right)\mu \quad (۱۰۹-۲)$$

$$b_9 = \left(\xi_x\eta_x + \frac{4}{3}\xi_y\eta_y\right)\mu \quad (۱۱۰-۲)$$

$$b_{10} = -\left(\frac{2}{3}\eta_y\frac{n}{y}\right)\mu \quad (۱۱۱-۲)$$

$$b_{11} = k\left(\xi_x\eta_x + \xi_y\eta_y\right) \quad (۱۱۲-۲)$$

با بازنویسی کردن $\hat{F}_v$ بر حسب ضرایب b_1 تا b_{11} خواهیم داشت :

$$\hat{F}_v = Jy^n\begin{bmatrix} 0 \\ \left(b_5u_\xi + b_1u_\eta + b_6v_\xi + b_2v_\eta + b_7v\right) \equiv row2 \\ \left(b_8u_\xi + b_2u_\eta + b_9v_\xi + b_3v_\eta + b_{10}v\right) \equiv row3 \\ b_{11}T_\xi + b_4T_\eta + u\times(row2) + v\times(row3) \end{bmatrix} \quad (۱۱۳-۲)$$

۲-۴-۳-۶- محاسبه $\hat{H}$

$$\hat{H} = J\begin{bmatrix} 0 \\ y^n g_x(\rho - \rho_\infty) \\ y^n g_y(\rho - \rho_\infty) + n\left(p - \tau_{\theta\theta}\right) \\ S_h \end{bmatrix} \quad (۱۱۴-۲)$$

در رابطه بالا $\tau_{\theta\theta}$ قبلاً محاسبه شده است. همچنین دقت شود که اندیس y,x بر روی g نشان دهنده مولفه‌های شتاب جاذبه می‌باشند نه مشتق.شتاب جاذبه در راستای x یا به عبارتی r صفر در نظر گرفته می شود.

۲-۴-۴- محاسبه متریکهای تبدیل

حل اختلاف محدود برای معادلات انتقال داده شده در فضای محاسباتی معادل است با حل حجم محدود در فضای فیزیکی. متریکهای تبدیل $\eta_y,\eta_x,\xi_y,\xi_x$ نیز از مقایسه دو حالت ذکر شده بدست می‌آیند.

شکل معادلات دیفرانسیل در فضای محاسباتی در حالت کلی به صورت زیر می‌باشد :

$$\left(Jy^n q\right)_t + \left[Jy^n\left(\xi_x E + \xi_y F\right)\right]_\xi + \left[Jy^n\left(\eta_x E + \eta_y F\right)\right]_\eta = Jy^n(S) \qquad (۱۱۵\text{-}۲)$$

اگر معادله بالا، معادله بقای جرم باشد، آنگاه :

$$q=\rho \quad , \quad E=\rho u \quad , \quad F=\rho v \quad , \quad S=0$$

اگر بیانگر بقای مومنتم x باشد :

$$q=\rho u \quad , \quad E=\rho u^2+p-\tau_{xx} \quad , \quad F=\rho uv-\tau_{xy} \quad , \quad S=g_x(\rho-\rho_\infty)$$

اگر نشان دهنده بقای مومنتم y باشد :

$$q=\rho v \quad , \quad E=\rho uv-\tau_{xy} \quad , \quad F=\rho v^2+p-\tau_{yy} \quad , \quad S=g_y(\rho-\rho_\infty)-n\frac{\tau_{\theta\theta}}{y^n}+n\frac{p}{y^n}$$

و بالاخره برای بقای انرژی :

$$q=e \quad , \quad E=(e+p)u-u\tau_{xx}-v\tau_{xy}+\dot{q}_x \quad , \quad F=(e+p)v-u\tau_{xy}-v\tau_{yy}+\dot{q}_y$$
$$S=S_h$$

شکل کلی معادلات بقا در حالت انتگرالی نیز به صورت زیر می‌باشد :

$$\frac{\partial}{\partial t}\oint_{C.\forall} q\,d\forall + \oint_{C.S}\left(n_x E + n_y F\right)dA = \oint_{C.\forall} S\,d\forall \qquad (۱۱۶\text{-}۲)$$

که در این رابطه n_y, n_x، مولفه‌های بردارهای یکه عمود بر هر یک از سطوح نسبت به محورهای y, x می‌باشند.

اگر معادله انتگرالی فوق معادله بقای جرم باشد، داریم :

$$q = \rho, \quad E = \rho u, \quad F = \rho v, \quad S = 0$$

اگر بقای مومنتم x باشد :

$$q = \rho u, \quad E = \rho u^2 + p - \tau_{xx}, \quad F = \rho uv - \tau_{xy}, \quad S = g_x(\rho - \rho_\infty)$$

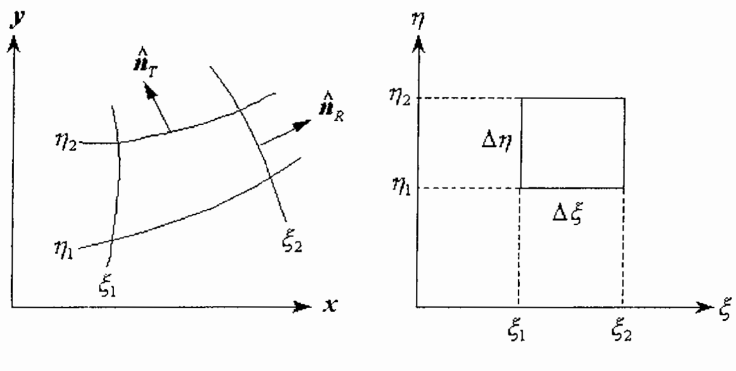
در حالتی که بقای مومنتم y را بیان کند :

$$q = \rho v, \quad E = \rho uv - \tau_{xy}, \quad F = \rho v^2 + p - \tau_{yy}, \quad S = g_y(\rho - \rho_\infty) - n\frac{\tau_{\theta\theta}}{y^n}$$

و برای بقای انرژی :

$$q = e, \quad E = (e+p)u - u\tau_{xx} - v\tau_{xy} + \dot{q}_x, \quad F = (e+p)v - u\tau_{xy} - v\tau_{yy} + \dot{q}_y$$
$$S = S_h$$

برای اعمال روش حجم محدود به معادلات انتگرالی و روش اختلاف محدود به معادلات دیفرانسیلی در فضای محاسباتی شکل (۲-۱) را در نظر بگیرید. شکل سمت چپ یک سلول را در فضای فیزیکی و شکل سمت راست همان سلول را در فضای محاسباتی نشان می‌دهد.



شکل(۲-۱): نمایی شماتیک از یک سلول در فضای فیزیکی و محاسباتی

پس از اعمال روش حجم محدود به معادلات انتگرالی در فضای فیزیکی داریم :

$$\frac{\Delta\forall}{\Delta t}\left(q^{n+1}-q^{n}\right)+\left[\left(n_x E+n_y F\right)dA\right]_L^R+\left[\left(n_x E+n_y F\right)dA\right]_B^T=S\left(\Delta\forall\right) \qquad (۱۱۷-۲)$$

با اعمال روش اختلاف محدود به معادلات دیفرانسیلی در فضای محاسباتی نیز خواهیم داشت :

$$\frac{Jy^{n}}{\Delta t}\left(q^{n+1}-q^{n}\right)+\frac{\left[Jy^{n}\left(\xi_x E+\xi_y F\right)\right]_L^R}{\Delta\xi}+\frac{\left[Jy^{n}\left(\eta_x E+\eta_y F\right)\right]_B^T}{\Delta\eta}=Jy^{n}\left(S\right) \qquad (۱۱۸-۲)$$

طرفین رابطه فوق را در $\Delta\xi.\Delta\eta$ ضرب می‌کنیم تا به صورت شار عبوری از هر سطح دراید :

$$\begin{aligned}&\frac{Jy^{n}\Delta\xi.\Delta\eta}{\Delta t}\left(q^{n+1}-q^{n}\right)+\Delta\eta\left[Jy^{n}\left(\xi_x E+\xi_y F\right)\right]_L^R\\&+\Delta\xi\left[Jy^{n}\left(\eta_x E+\eta_y F\right)\right]_B^T=Jy^{n}\Delta\xi.\Delta\eta\left(S\right)\end{aligned} \qquad (۱۱۹-۲)$$

با معادل گذاشتن روابط در فضای فیزیکی و محاسباتی داریم :

$$\frac{Jy^{n}\Delta\xi.\Delta\eta}{\Delta t}=\frac{\Delta\forall}{\Delta t} \qquad (۱۲۰-۲)$$

$$Jy^{n}\Delta\eta.\xi_{x,R}=n_{x,R}dA_R \qquad (۱۲۱-۲)$$

$$Jy^{n}\Delta\eta.\xi_{y,R}=n_{y,R}dA_R \qquad (۱۲۲-۲)$$

$$Jy^{n}\Delta\xi.\eta_{x,R}=n_{x,T}dA_T \qquad (۱۲۳-۲)$$

$$Jy^{n}\Delta\xi.\eta_{y,T}=n_{y,T}dA_T \qquad (۱۲۴-۲)$$

$$Jy^{n}\Delta\xi.\Delta\eta=\Delta\forall \qquad (۱۲۵-۲)$$

از روابط به دست آمده می‌توان نتیجه گرفت :

$$Jy^{n}=\frac{\Delta\forall}{\Delta\xi.\Delta\eta} \qquad (۱۲۶-۲)$$

$$Jy^{n}\xi_{x,R}=\frac{n_{x,R}}{\Delta\eta}dA_R \qquad (۱۲۷-۲)$$

$$Jy^{n}\xi_{y,R}=\frac{n_{y,R}}{\Delta\eta}dA_R \qquad (۱۲۸-۲)$$

$$Jy^{n}\eta_{x,T}=\frac{n_{x,T}}{\Delta\xi}dA_T \qquad (۱۲۹-۲)$$

$$Jy^n\eta_{y,T} = \frac{n_{y,T}}{\Delta\xi} dA_T \qquad (۱۳۰-۲)$$

اگر در ر ابطه (۲-۱۲۶)، $\Delta\xi = \Delta\eta = 1$ در نظر گرفته شود، آنگاه ژاکوبین تبدیل برابر حجم سلول می‌باشد:

$$Jy^n = \Delta\forall \qquad (۱۳۱-۲)$$

با توجه به روابط به دست آمده ملاحظه می‌شود که می‌توان با محاسبه مولفه‌های بردارهای یکه عمود بر هر یک از سطوح، متریکهای تبدیل محاسبه کرد.

۲-۴-۵- محاسبه مولفه‌های بردارهای عمود بر سطوح یک سلول

ثابت می‌شود مولفه‌های بردارهای یکه عمود بر سطح را می‌توان از رابطه زیر حساب کرد :

$$\begin{cases} n_x = (sign)\dfrac{\Delta y}{A} \\ n_y = -(sign)\dfrac{\Delta x}{A} \end{cases} \qquad (۱۳۲-۲)$$

که در رابطه بالا :

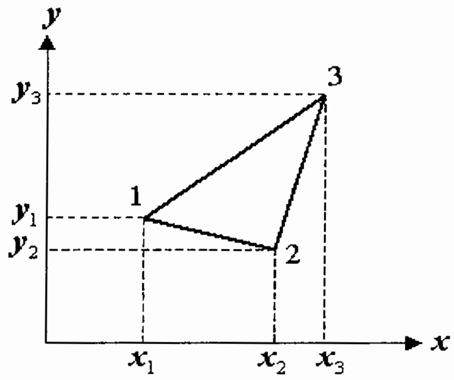
$$\Delta x = x_2 - x_1 \quad , \quad \Delta y = y_2 - y_1 \quad , \quad A = \sqrt{(\Delta x)^2 + (\Delta y)^2}$$

و مقدار $sign$ با توجه به نوع سطح، یکی از مقادیر درج شده در جدول زیر است :

Face	*Right*	*Left*	*Bottom*	*Top*
sign	+۱	-۱	-۱	+۱

۲-۴-۶- محاسبه حجم مثلثهای تشکیل دهنده سلول

یک سلول با چهار ضلع را مانند شکل(۲-۱) در نظر بگیرید.این سلول را می‌توان متشکل از دو مثلث دانست. اگر مختصات رئوس هر یک از مثلثها معلوم باشد، میتوان مساحت آنها را تعیین نمود. حال یک مثلث مطابق شکل (۲-۲) را در نظر بگیرید :



شکل(۲-۲):شماره گذاری رئوس هر مثلث به منظور محاسبه مساحت

ثابت می‌شود که می‌توان مساحت مثلث 123 را از رابطه زیر محاسبه نمود :

$$A_{123} = \frac{1}{2}\left(\Delta x_{12}\Delta y_{13} - \Delta x_{13}\Delta y_{12}\right) \qquad (۲-۱۳۳)$$

که دراین رابطه :

$$\begin{cases} \Delta x_{ij} = x_j - x_i \\ \Delta y_{ij} = y_j - y_i \end{cases}$$

با استفاده از روابط بدست آمده می‌توان V,U را تصحیح نمود. قبلاً تعریف کردیم :

$$\begin{cases} U = \xi_x u + \xi_y v \\ V = \eta_x u + \eta_y v \end{cases} \qquad (۲-۱۳۴)$$

اگر بجای $\xi_x, \xi_y, \eta_x, \eta_y$ مقادیر بدست آمده را جایگزین کنیم، داریم :

$$\begin{cases} U = n_{x,R}\dfrac{dA_R}{Jy^n\Delta\eta}u + n_{y,R}\dfrac{dA_R}{Jy^n\Delta\eta}v \\ V = n_{x,T}\dfrac{dA_T}{Jy^n\Delta\xi}u + n_{y,T}\dfrac{dA_T}{Jy^n\Delta\xi}v \end{cases} \qquad (۲-۱۳۵)$$

حال تعریف می‌کنیم :

$$\overline{u} \equiv \frac{U}{\left(dA_R/Jy^n\Delta\eta\right)} = n_{x,R}u + n_{y,R}v \qquad (۲-۱۳۶)$$

همچنین :

$$\overline{v} \equiv \frac{V}{\left(dA_T/Jy^n\Delta\xi\right)} = n_{x,T}u + n_{y,T}v \qquad (۲-۱۳۷)$$

همانطور که ملاحظه می‌شود، مقادیر $\overline{v},\overline{u}$ به راحتی قابل محاسبه مـی‌باشـند. از روابـط (۲-۱۳۶) و (۲-۱۳۷) داریم :

$$\begin{cases} U = \dfrac{dA_R}{Jy^n\Delta\eta}\overline{u} \\ V = \dfrac{dA_T}{Jy^n\Delta\xi}\overline{v} \end{cases} \qquad (۲-۱۳۸)$$

با قرار دادن V,U در روابط مربوط به $\hat{F}_i\,,\hat{E}_i$ می‌توان تصحیح زیر را انجام داد :

$$\hat{E}_i = \frac{dA_R}{\Delta\eta}\begin{bmatrix} \rho\overline{u} \\ \rho u\overline{u} + n_{x,R}p \\ \rho v\overline{u} + n_{y,R}p \\ (e+p)\overline{u} \end{bmatrix} \quad , \quad \hat{F}_i = \frac{dA_T}{\Delta\xi}\begin{bmatrix} \rho\overline{v} \\ \rho u\overline{v} + n_{x,T}p \\ \rho v\overline{v} + n_{y,T}p \\ (e+p)\overline{v} \end{bmatrix} \qquad (۲-۱۳۹)$$

اگر مقادیر $\xi_{x,T}$، $\xi_{y,T}$، $\eta_{x,T}$، $\eta_{y,T}$ را در ضرایب b_i,a_i جایگزین نماییم، می‌توان تصحیح زیـر را نیـز برای این ضرایب انجام داد :

$$a_1 = \left(\frac{1}{3}n_{x,R}^2 + 1\right)\left(\frac{dA_R}{Jy^n\Delta\eta}\right)^2\mu \qquad (۲-۱۴۰)$$

$$a_2 = \frac{1}{3}n_{x,R}n_{y,R}\left(\frac{dA_R}{Jy^n\Delta\eta}\right)^2\mu \qquad (۲-۱۴۱)$$

$$a_3 = \left(1 + \frac{1}{3} n_{y,R}{}^2\right)\left(\frac{dA_R}{Jy^n \Delta \eta}\right)^2 \mu \qquad (۱۴۲-۲)$$

$$a_4 = k\left(\frac{dA_R}{Jy^n \Delta \eta}\right)^2 \qquad (۱۴۳-۲)$$

$$a_5 = \left(\frac{4}{3} n_{x,R}\eta_{x,R} - \frac{2}{3} n_{y,R}\eta_{y,R}\right)\left(\frac{dA_R}{Jy^n \Delta \eta}\right)\mu \qquad (۱۴۴-۲)$$

$$a_6 = \left(n_{y,R}\eta_{x,R} - \frac{2}{3} n_{x,R}\eta_{y,R}\right)\left(\frac{dA_R}{Jy^n \Delta \eta}\right)\mu \qquad (۱۴۵-۲)$$

$$a_7 = -\frac{2}{3}\left(n_{x,R}\frac{n}{y_R}\right)\left(\frac{dA_R}{Jy^n \Delta \eta}\right)\mu \qquad (۱۴۶-۲)$$

$$a_8 = \left(n_{x,R}\eta_{y,R} - \frac{2}{3} n_{y,R}\eta_{x,R}\right)\left(\frac{dA_R}{Jy^n \Delta \eta}\right)\mu \qquad (۱۴۷-۲)$$

$$a_9 = \left(n_{x,R}\eta_{x,R} + \frac{4}{3} n_{y,R}\eta_{y,R}\right)\left(\frac{dA_R}{Jy^n \Delta \eta}\right)\mu \qquad (۱۴۸-۲)$$

$$a_{10} = -\frac{2}{3}\left(n_{y,R}\frac{n}{y_R}\right)\left(\frac{dA_R}{Jy^n \Delta \eta}\right)\mu \qquad (۱۴۹-۲)$$

$$a_{11} = k\left(n_{x,R}\eta_{x,R} + n_{y,R}\eta_{y,R}\right)\left(\frac{dA_R}{Jy^n \Delta \eta}\right) \qquad (۱۵۰-۲)$$

برای ضرایب b_i نیز داریم :

$$b_1 = \left(\frac{1}{3} n_{x,T}{}^2 + 1\right)\left(\frac{dA_T}{Jy^n \Delta \xi}\right)^2 \mu \qquad (۱۵۱-۲)$$

$$b_2 = \frac{1}{3} n_{x,T} n_{y,T}\left(\frac{dA_T}{Jy^n \Delta \xi}\right)^2 \mu \qquad (۱۵۲-۲)$$

$$b_3 = \left(1 + \frac{1}{3} n_{y,T}{}^2\right)\left(\frac{dA_T}{Jy^n \Delta \xi}\right)^2 \mu \qquad (۱۵۳-۲)$$

$$b_4 = k\left(\frac{dA_T}{Jy^n \Delta \xi}\right)^2 \qquad (۱۵۴-۲)$$

$$b_5 = \left(\frac{4}{3}\xi_{x,T}n_{x,T} + \xi_{y,T}n_{y,T}\right)\left(\frac{dA_T}{Jy^n\Delta\xi}\right)\mu \tag{۱۵۵-۲}$$

$$b_6 = \left(\xi_{x,T}n_{y,T} - \frac{2}{3}\xi_{y,T}n_{x,T}\right)\left(\frac{dA_T}{Jy^n\Delta\xi}\right)\mu \tag{۱۵۶-۲}$$

$$b_7 = -\frac{2}{3}\left(n_{x,T}\frac{n}{y_T}\right)\left(\frac{dA_T}{Jy^n\Delta\xi}\right)\mu \tag{۱۵۷-۲}$$

$$b_8 = \left(\xi_{y,T}n_{x,T} - \frac{2}{3}\xi_{x,T}n_{y,T}\right)\left(\frac{dA_T}{Jy^n\Delta\xi}\right)\mu \tag{۱۵۸-۲}$$

$$b_9 = \left(\xi_{x,T}n_{x,T} + \frac{4}{3}\xi_{y,T}n_{y,T}\right)\left(\frac{dA_T}{Jy^n\Delta\xi}\right)\mu \tag{۱۵۹-۲}$$

$$b_{10} = -\frac{2}{3}\left(n_{y,T}\frac{n}{y_T}\right)\left(\frac{dA_T}{Jy^n\Delta\xi}\right)\mu \tag{۱۶۰-۲}$$

$$b_{11} = k\left(\xi_{x,T}n_{x,T} + \xi_{y,T}n_{y,T}\right)\left(\frac{dA_T}{Jy^n\Delta\xi}\right) \tag{۱۶۱-۲}$$

۲-۵- تولید شبکه به روش حل معادلات دیفرانسیل (از نوع بیضوی)

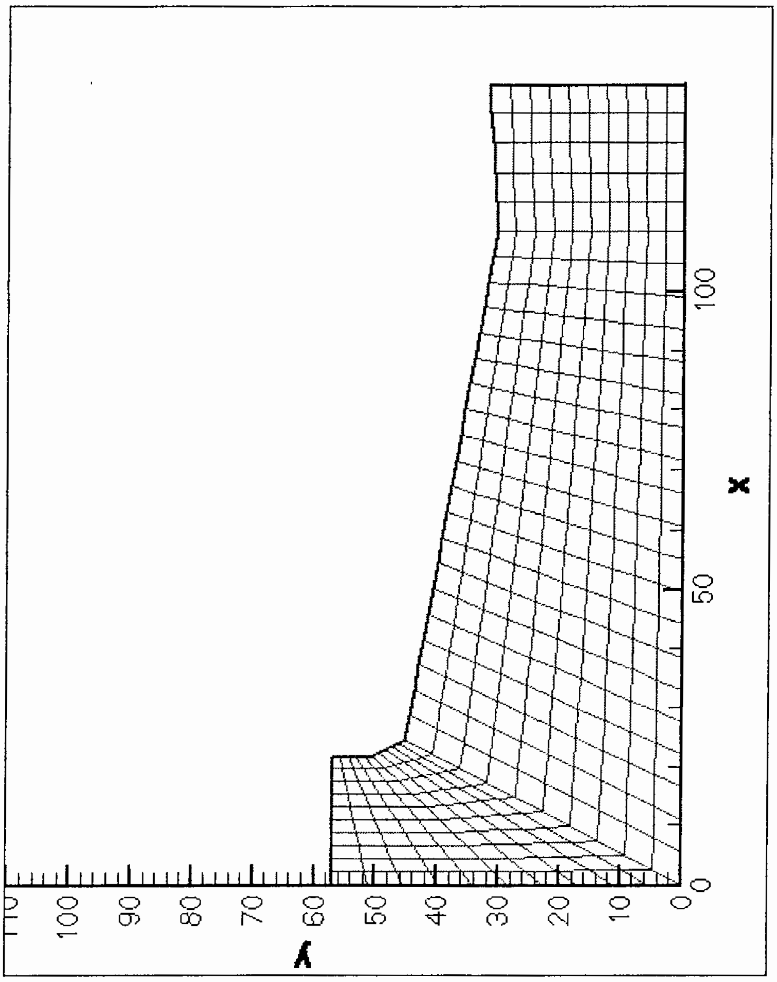
در این روش یک دستگاه معادلات دیفرانسیل پاره‌ای را حل می‌کنیم تا نقاط شبکه در فضای فیزیکی بدست آید. برای درک بهتر این موضوع، معادله هدایت حرارتی دو بعدی دائم را در نظر بگیرید. اگر توزیع دما بر روی مرزها معلوم باشد، توزیع دما در نقاط داخلی به دست می‌آید.

حال دستگاه معادلات دیفرانسیل پاره‌ای زیر را در نظر بگیرید :

$$\begin{cases} \dfrac{\partial^2\xi}{\partial x^2} + \dfrac{\partial^2\xi}{\partial y^2} = 0 \\ \dfrac{\partial^2\eta}{\partial x^2} + \dfrac{\partial^2\eta}{\partial y^2} = 0 \end{cases} \tag{۱۶۲-۲}$$

در دستگاه معادلات بالا، متغیرهای مستقل y,x می‌باشند. حل دستگاه بالا با داشتن نقاط مرزی در فضای y,x، منجر به پیدا شدن نقاط داخلی در فضای محاسباتی η,ξ خواهد شد.

اما به دلیل اینکه فضای محاسباتی یک مستطیل می‌باشد و توزیع نقاط داخلی و مرزی آن مشخص است، می‌توان مساله را معکوس کرد و با جابجا کردن متغیرهای وابسته و مستقل و حل معادلات تبدیل شده به نقاط داخلی در فضای فیزیکی دست یافت. شبکه تولید شده توسط این روش در شکل (۲-۳) ملاحظه می گردد.



شکل(۲-۳): شبکه تولید شده

فصل سوم

روش حل معادلات در حالت متقارن محوری

۳-۱- روش حل معادلات

در این فصل به بررسی روشهای بالا دستی برای جملات غیر لزج پرداخته می‌شود. جملات لـزج نیـز جدای از مرتبه گسسته سازی جملات غیر لزج، همواره به صورت مرکزی[1] تقریب زده می‌شوند.

عبارت بالا دستی برای ترم‌های غیر لزج بدین معنی است که گسسته سازی بر اساس جهـت پخـش امواج به صورت پیشرو و یا پسرو انجام می‌شود. این امواج می‌توانند امواج مشخصه قـوانین بقـا و یـا یـک آشفتگی کوچک مانند سرعت صوت باشد.

ایده اصلی روشهای بالا دستی، مشاهده سیستمهای معادلات هذلولوی مانند معادلات اویلر مـی‌باشـد که در این معادلات، موج، اطلاعات فیزیکی را در طول مسیرهای خاصی منتشر می‌کند. ایـن مسـیرها در حقیقت جهت‌های مشخصه نامیده می‌شوند.

روشهای بالا دستی به دو دسته اصلی تقسیم می‌شوند [۶] :

روشهای تجزیه بردار شار[2]

روشهای تجزیه اختلاف شار[3]

1- Centeral

2- Flux Vector Splitting

3- Flux Difference Splitting

4- Steger Warming

5- Van Leer

در روش تجزیه بردار شار، جملات شار بر اساس علامت مقادیر ویژه ماتریس ژاکوبین بردارهـای شـار تجزیه می‌شوند و گسسته سازی بر طبق علامت سرعت انتشار تعیین می‌شوند.

در روش تجزیه اختلاف شار، حل مساله توسط حـل دقیـق مسـاله ریمـان در سـطح بـین سـلولهای محاسباتی بدست می‌آید.

۳-۱-۱- خطی سازی معادلات حاکم

به منظور تحقیق بر روی خواص جملات غیر لزج و نیز حل عددی معادلات به روی تجزیه بردار شار، اولین قدم خطی کردن معادلات حاکم می‌باشد.

معادلات حاکم در فضای محاسباتی به صورت زیر می‌باشند :

$$\frac{\partial \hat{Q}}{\partial t}+\frac{\partial \hat{E}}{\partial \xi}+\frac{\partial \hat{F}}{\partial \eta}=\hat{H} \tag{۱-۳}$$

که در روابط بالا :

$$\begin{cases}\hat{E}=\hat{E}_i-\hat{E}_v\\ \hat{F}=\hat{F}_i-\hat{F}_v\end{cases} \tag{۲-۳}$$

با قرار دادن روابط (۳-۱۶۴) در معادله (۳-۱۶۳) داریم :

$$\frac{\partial \hat{Q}}{\partial t}+\frac{\partial\left[\hat{E}_i-\hat{E}_v\right]}{\partial \xi}+\frac{\partial\left[\hat{F}_i-\hat{F}_v\right]}{\partial \eta}=\hat{H} \tag{۳-۳}$$

و یا :

$$\frac{\partial \hat{Q}}{\partial t}+\frac{\partial \hat{E}_i}{\partial \xi}-\frac{\partial \hat{E}_v}{\partial \xi}+\frac{\partial \hat{F}_i}{\partial \eta}-\frac{\partial \hat{F}_v}{\partial \eta}=\hat{H} \tag{۴-۳}$$

اگر برای جملات غیر لزج داشته باشیم :

$$\begin{cases}\hat{E}_i=\hat{E}_i\left(\hat{Q}\right)\\ \hat{F}_i=\hat{F}_i\left(\hat{Q}\right)\end{cases} \tag{۵-۳}$$

که در روابط بالا :

$$\hat{Q}=\hat{Q}(\xi,\eta,t) \tag{۶-۳}$$

آنگاه با استفاده از قانون مشتق زنجیره‌ای داریم :

$$\begin{cases} \dfrac{\partial \hat{E}_i}{\partial \xi} = \dfrac{\partial \hat{E}_i}{\partial \hat{Q}} \dfrac{\partial \hat{Q}}{\partial \xi} \\ \dfrac{\partial \hat{F}_i}{\partial \eta} = \dfrac{\partial \hat{F}_i}{\partial \hat{Q}} \dfrac{\partial \hat{Q}}{\partial \eta} \end{cases} \qquad (۷-۳)$$

طبق تعریف :

$$\begin{cases} \hat{A}_i \equiv \dfrac{\partial \hat{E}_i}{\partial \hat{Q}} \\ \hat{B}_i \equiv \dfrac{\partial \hat{F}_i}{\partial \hat{Q}} \end{cases} \qquad (۸-۳)$$

بنا بر این معادلات (۳-۷) به صورت زیر در می‌آیند :

$$\begin{cases} \dfrac{\partial \hat{E}_i}{\partial \xi} = \hat{A}_i \dfrac{\partial \hat{Q}}{\partial \xi} \\ \dfrac{\partial \hat{F}_i}{\partial \eta} = \hat{B}_i \dfrac{\partial \hat{Q}}{\partial \eta} \end{cases} \qquad (۹-۳)$$

عبارتهای $\hat{B}_i$, $\hat{A}_i$ ماتریسهای ژاکوبین تبدیل شارهای غیر لزج نام دارند.

۳-۱-۱-۱- محاسبه ژاکوبین تبدیل شارهای غیر لزج

الف - محاسبه $\hat{A}_i$

برای محاسبه این جمله به ترتیب زیر عمل می‌کنیم :

$$\hat{A}_i = \frac{\partial \hat{E}_i}{\partial \hat{Q}} = \frac{\partial\left[J\left(\xi_x E_i + \xi_y F_i\right)\right]}{\partial (JQ)} = \xi_x \frac{\partial E_i}{\partial Q} + \xi_y \frac{\partial F_i}{\partial Q} \qquad (۱۰-۳)$$

ماتریسهای B_i , A_i را به صورت زیر تعریف می‌کنیم :

$$\begin{cases} A_i = \dfrac{\partial E_i}{\partial Q} \\ B_i = \dfrac{\partial F_i}{\partial Q} \end{cases} \qquad (۱۱\text{-}۳)$$

مقادیر این ماتریسها پس از محاسبه عبارتند از :

$$A_i = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \dfrac{\gamma-1}{2}\left(u^2+v^2\right)-u^2 & (3-\gamma)u & -(\gamma-1)v & (\gamma-1) \\ -uv & v & u & 0 \\ \left[(\gamma-1)\left(u^2+v^2\right)-\dfrac{\gamma e}{\rho}\right]u & \dfrac{\gamma e}{\rho}-\dfrac{\gamma-1}{2}\left(3u^2+v^2\right) & -(\gamma-1)uv & \gamma u \end{bmatrix}$$

و

$$B_i = \begin{bmatrix} 0 & 0 & 1 & 0 \\ -uv & v & u & 0 \\ \dfrac{\gamma-1}{2}\left(u^2+v^2\right)-v^2 & -(\gamma-1)u & (3-\gamma)v & \gamma-1 \\ \left[(\gamma-1)\left(u^2+v^2\right)-\dfrac{\gamma e}{\rho}\right]v & -(\gamma-1)uv & \dfrac{\gamma e}{\rho}-\dfrac{\gamma-1}{2}\left(u^2+3v^2\right) & \gamma v \end{bmatrix} \qquad (۱۲\text{-}۳)$$

با تعریف h به صورت :

$$h \equiv \frac{e+p}{\rho} = \frac{\gamma e}{\rho} - \frac{\gamma-1}{2}\left(u^2+v^2\right) \qquad (۱۳\text{-}۳)$$

و با جاگذاری مقادیر A_i , B_i در $\hat{A}_i$ و ساده سازی نتیجه می‌شود :

$$\hat{A}_i = \begin{bmatrix} \hat{a}_{11,i} & \hat{a}_{12,i} & \hat{a}_{13,i} & \hat{a}_{14,i} \\ \hat{a}_{21,i} & \hat{a}_{22,i} & \hat{a}_{23,i} & \hat{a}_{24,i} \\ \hat{a}_{31,i} & \hat{a}_{32,i} & \hat{a}_{33,i} & \hat{a}_{34,i} \\ \hat{a}_{41,i} & \hat{a}_{42,i} & \hat{a}_{43,i} & \hat{a}_{44,i} \end{bmatrix} \qquad (۱۴\text{-}۳)$$

که در این رابطه :

$$\hat{a}_{11,i} = 0$$

$$\hat{a}_{12,i} = \xi_x$$

$$\hat{a}_{13,i} = \xi_y$$

$$\hat{a}_{14,i} = 0$$

$$\hat{a}_{21,i} = \xi_x \frac{\gamma - 1}{2}\left(u^2 + v^2\right) - uU$$

$$\hat{a}_{22,i} = \xi_x (2 - \gamma)u + U$$

$$\hat{a}_{23,i} = \xi_y u - \xi_x (\gamma - 1)v$$

$$\hat{a}_{24,i} = \xi_x (\gamma - 1)$$

$$\hat{a}_{31,i} = \xi_y \frac{\gamma - 1}{2}\left(u^2 + v^2\right) - vU$$

$$\hat{a}_{32,i} = \xi_x v - \xi_y (\gamma - 1)u$$

$$\hat{a}_{33,i} = U + \xi_y (2 - \gamma)v$$

$$\hat{a}_{34,i} = \xi_y (\gamma - 1)$$

$$\hat{a}_{41,i} = \left[\frac{\gamma - 1}{2}\left(u^2 + v^2\right) - h\right]U$$

$$\hat{a}_{42,i} = \xi_x h - (\gamma - 1)uU$$

$$\hat{a}_{43,i} = \xi_x h - (\gamma - 1)vU$$

$$\hat{a}_{44,i} = \gamma U$$

لازم به ذکر است که U عبارت است از : $U = \xi_x u + \xi_y v$

ب - محاسبه $\hat{B}_i$

اگر مراحلی را که برای محاسبه $\hat{A}_i$ طی کردیم، برای $\hat{B}_i$ نیز طی کنیم، خواهیم داشت :

$$\hat{B}_i = \begin{bmatrix} \hat{b}_{11,i} & \hat{b}_{12,i} & \hat{b}_{13,i} & \hat{b}_{14,i} \\ \hat{b}_{21,i} & \hat{b}_{22,i} & \hat{b}_{23,i} & \hat{b}_{24,i} \\ \hat{b}_{31,i} & \hat{b}_{32,i} & \hat{b}_{33,i} & \hat{b}_{34,i} \\ \hat{b}_{41,i} & \hat{b}_{42,i} & \hat{b}_{43,i} & \hat{b}_{44,i} \end{bmatrix} \qquad (۱۵\text{-}۳)$$

در رابطه (۳-۱۵) نیز :

$$\hat{b}_{11,i} = 0$$

$$\hat{b}_{12,i} = \eta_x$$

$$\hat{b}_{13,i} = \eta_y$$

$$\hat{b}_{14,i} = 0$$

$$\hat{b}_{21,i} = \eta_x \frac{\gamma - 1}{2}\left(u^2 + v^2\right) - uV$$

$$\hat{b}_{22,i} = \eta_x (2 - \gamma) u + V$$

$$\hat{b}_{23,i} = \eta_y u - \eta_x (\gamma - 1) v$$

$$\hat{b}_{24,i} = \eta_x (\gamma - 1)$$

$$\hat{b}_{31,i} = \eta_y \frac{\gamma - 1}{2}\left(u^2 + v^2\right) - vV$$

$$\hat{b}_{32,i} = \eta_x v - \eta_y (\gamma - 1) u$$

$$\hat{b}_{33,i} = V + \eta_y (2 - \gamma) v$$

$$\hat{b}_{34,i} = \eta_y (\gamma - 1)$$

$$\hat{b}_{41,i} = \left[\frac{\gamma - 1}{2}\left(u^2 + v^2\right) - h\right] V$$

$$\hat{b}_{42,i} = \eta_x h - (\gamma - 1) uV$$

$$\hat{b}_{43,i} = \eta_y h - (\gamma - 1) vV$$

$$\hat{b}_{44,i} = \gamma V$$

در روابط بالا نیز، V عبارت است از : $V = \eta_x u + \eta_y v$

۳-۱-۲- روشهای تجزیه بردار شارهای غیر لزج

یکی از زیر گروه‌های روشهای بالا دستی، روشهای تجزیه بردار شار است. اساس این روشـها در سـال ۱۹۵۲ توسط کورانت[1] برای روش بالا دست مرتبه اول بیان گردید و در سال ۱۹۸۱ و۱۹۸۲ توسط استگر و وارمینگ کامل گردید [۷].

برای بررسی این روش ماتریس دلخواه $A_{3\times3}$ را در نظر بگیرید. حال فـرض کنیـد مقـادیر ویـژه ایـن ماتریس به ترتیب $\lambda_3,\lambda_2,\lambda_1$ باشد. در این صورت اگر $\Lambda(A)$ ماتریس مقادیر ویژه ماتریس A باشد و S ماتریس بردارهای ویژه باشد، به لحاظ ریاضی می‌توان ثابت نمود :

$$A = S\Lambda S^{-1} \tag{۱۶-۳}$$

حال ماتریس مقادیر ویژه Λ را می‌توان به صورت جمع دو مقدار Λ^- , Λ^+ در نظر گرفت :

$$\Lambda = \Lambda^+ + \Lambda^- \tag{۱۷-۳}$$

$$A = S\left(\Lambda^+ + \Lambda^-\right)S^{-1} \tag{۱۸-۳}$$

بنابر این ماتریس A را می‌توان به صورت زیر نوشت :

$$A = S\left(\Lambda^+ + \Lambda^-\right)S^{-1} = S\Lambda^+S^{-1} + S\Lambda^-S^{-1} = A^+ + A^- \tag{۱۹-۳}$$

طبق رابطه (۳-۱۹) ماتریس A به دو قسمت مثبت و منفی A^-, A^+ تجزیه شود به طوری که :

$$\begin{cases} A^+ = S\Lambda^+S^{-1} \\ A^{-1} = S\Lambda^{-1}S^{-1} \end{cases} \tag{۲۰-۳}$$

در صورتی که مقادیر ویژه ماتریسهای A^-, A^+ محاسبه شود، ملاحظه خواهد شد که :

$$\begin{cases} \Lambda\left(A^+\right) = \Lambda^+ \geq 0 \\ \Lambda\left(A^-\right) = \Lambda^- \leq 0 \end{cases} \tag{۲۱-۳}$$

حال فرض کنید $F_{3\times1}$ حاصل ضرب یک ماتریس $A_{3\times3}$ در بردار $Q_{3\times1}$ باشد :

1- Courant

$$F = AQ \qquad (۳-۲۲)$$

در روش تجزیه شار استگر - وارمینگ یک ماتریس شار عمومی مانند F_G به صورت زیر تعریف می‌شود:

$$F_G(D) = SDS^{-1}Q \qquad (۳-۲۳)$$

که D یک ماتریس قطری است و عناصر قطر آن مقادیر ویژه ماتریس A می‌باشند.

حال با توجه به D، F_G می‌تواند به صورتهای زیر باشد :

$$\begin{cases} if \quad D = \Lambda^+ \quad \Rightarrow \quad F_G(\Lambda^+) = F^+ \\ if \quad D = \Lambda^- \quad \Rightarrow \quad F_G(\Lambda^-) = F^- \end{cases} \qquad (۳-۲۴)$$

و در حالت کلی نیز :

$$F_G(\Lambda) = F_G(\Lambda^+ + \Lambda^-) = F \qquad (۳-۲۵)$$

برای روشن تر شدن این موضوع معادله اویلر یک بعدی زیر را در نظر بگیرید :

$$\frac{\partial Q}{\partial t} + \frac{\partial F}{\partial x} = 0, \quad Q = \begin{Bmatrix} \rho \\ \rho u \\ e \end{Bmatrix} \quad , \quad F = \begin{Bmatrix} \rho u \\ \rho u^2 + p \\ (e+p)u \end{Bmatrix} \qquad (۳-۲۶)$$

این معادله به صورت زیر خطی می‌شود :

$$\frac{\partial Q}{\partial t} + A\frac{\partial Q}{\partial x} = 0, \quad A = \frac{\partial F}{\partial Q} \qquad (۳-۲۷)$$

مقادیر ویژه ماتریس A عبارتند از :

$$\begin{cases} \lambda_1 = u = Mc \\ \lambda_2 = u + c = (M+1)c \\ \lambda_3 = u - c = (M-1)c \end{cases} \qquad (۳-۲۸)$$

اگر F_G را بر حسب Q, D محاسبه کنیم خواهیم داشت :

$$F_G = \frac{\rho}{2\gamma}\begin{Bmatrix} 2(\gamma-1)\lambda_1 + \lambda_2 + \lambda_3 \\ [2(\gamma-1)\lambda_1 M + \lambda_2(M+1) + \lambda_3(M-1)]c \\ \left[(\gamma-1)\lambda_1 M^2 + \frac{\lambda_2}{2}(M+1)^2 + \frac{\lambda_3}{2}(M-1)^2 + \frac{3-\gamma}{2(\gamma-1)}(\lambda_2+\lambda_3)\right]c^2 \end{Bmatrix} \qquad (۳-۲۹)$$

در صورتی که شار F_G بر حسب ریشه‌های λ_1 , λ_2 , λ_3 تعیین علامت شود، چهار ناحیه ظاهر خواهد شد که در جدول (۳-۱) مشخص شده است.

$$\begin{cases}\lambda_1 = u = Mc \\ \lambda_2 = u + c = (M+1)c \\ \lambda_3 = u - c = (M-1)c\end{cases} \Rightarrow \begin{cases} M = 0 \rightarrow \lambda_{11} = 0 \\ M = -1 \rightarrow \lambda_2 = 0 \\ M = 1 \rightarrow \lambda_3 = 0\end{cases}$$

جدول (۳-۱) مقادیر F_G بر حسب λ

$F^+ = F_G(0,0,0) = 0$ $F^- = F_G(\lambda_1,\lambda_2,\lambda_3) = F$	$F^+ = F_G(0,\lambda_2,0) = F^+{}_{\mathrm{II}}$ $F^+ = F_G(\lambda_1,0,\lambda_3) = F^-{}_{\mathrm{II}}$	$F^+ = F_G(\lambda_1,\lambda_2,0) = F^+{}_{\mathrm{III}}$ $F^- = F_G(0,0,\lambda_3) = F^-{}_{\mathrm{III}}$	$F^+ = F_G(\lambda_1,\lambda_2,\lambda_3) = F$ $F^- = F_G(0,0,0) = 0$
$M \le -1$	$-1 \le M \le 0$	$0 \le M \le 1$	$M \ge 1$

با محاسبه F داریم :

$$F = \begin{Bmatrix} f_1 \\ f_2 \\ f_3 \end{Bmatrix} = \begin{Bmatrix} mass \\ momentum \\ energy \end{Bmatrix} \Rightarrow \begin{cases} f_1 = \rho c M \\ f_2 = \rho c^2 \cdot \left(M^2 + \dfrac{1}{\gamma} \right) \\ f_3 = \rho c^3 \cdot \left(\dfrac{M^2}{2} + \dfrac{1}{\gamma - 1} \right) \cdot M \end{cases} \qquad (۳۰-۳)$$

حال می‌توان با توجه به رابطه (۳-۳۰) و جدول (۳-۱) به عنوان مثال شارهای جرمی‌تجزیه شده توسط روش استگر – وارمینگ[6] را به صورت زیر به دست آورد :

$$f^+{}_{1,SW} = \begin{cases} 0 & -\infty \le M \le -1 \\ \dfrac{\rho c}{2\gamma} & -1 \le M \le 0 \\ \dfrac{\rho c}{2\gamma}\left[2(\gamma - 1)M + (M+1)\right] & 0 \le \mathrm{M} \le 1 \\ \rho c M & 1 \le M \le +\infty \end{cases} \qquad (۳۱-۳)$$

$$f^{-}{}_{1,SW}=\begin{cases}\rho c M & -\infty\le M\le -1\\ \dfrac{\rho c}{2\gamma}\left[2(\gamma-1)M+(M-1)\right] & -1\le M\le 0\\ \dfrac{\rho c}{2\gamma}(M-1) & 0\le M\le 1\\ 0 & 1\le M\le +\infty\end{cases}\qquad (۳۲-۳)$$

همین کار را می‌توان برای شارهای مومنتم و انرژی نیز انجام داد که در اینجا از انجام آن صرف نظر می‌شود.

همانطور که از روابط (۳-۳۱) و (۳-۳۲) ملاحظه می‌شود، شار جرمی‌تجزیه شده تابع درجه اول از u می‌باشد. با نوشتن شارهای مومنتم و انرژی مشخص خواهد شد که شار مومنتم تابع درجه دوم از u و شار انرژی تابع درجه سوم از u می‌باشد. البته باید توجه داشت که روش شکافت شارها منحصر به فرد نمی‌باشد و به طرق مختلف می‌توان این کار را انجام داد.

۳-۱-۲-۱- تجزیه شارهای غیر لزج $\hat{F}_i, \hat{E}_i$ بر مبنای روش استگر- وارمینگ

الف -تجزیه بردار شار $\hat{E}_i$

اگر شار عمومی $\hat{E}_i$ بر حسب مقادیر ویژه ماتریس ژاکوبین تبدیل $\hat{A}_i=\dfrac{\partial \hat{E}_i}{\partial \hat{Q}}$، مرتب شود می‌توان شارهای مثبت و منفی را بر اساس علامت مقادیر ویژه ماتریس مذکور تعیین کرد. توجه شود که در فضای محاسباتی مقادیر ویژه ماتریس ژاکوبین تبدیل $\hat{A}_i$ عبارتند از :

$$\begin{cases}\hat{\lambda}_1=\overline{u}\\ \overline{\lambda}_2=\overline{u}\\ \overline{\lambda}_3=\overline{u}+c\\ \overline{\lambda}_4=\overline{u}-c\end{cases}\qquad (۳۳-۳)$$

که در رابطه بالا چنانچه توضیح داده شد :

$\overline{u}=n_{x,R}u+n_{y,R}v$

اگر :

$$\hat{E}^{+} = \begin{bmatrix} \hat{e}^{+}{}_{1} \\ \hat{e}^{+}{}_{2} \\ \hat{e}^{+}{}_{3} \\ \hat{e}^{+}{}_{4} \end{bmatrix} \qquad (۳۴\text{-}۳)$$

باشد، پس از محاسبه، مقادیر شارهای مثبت و منفی به ازای مقادیر مختلف M_{ξ} به شرح زیر می‌باشد:

$$if \quad -\infty \le M_{\xi} \le -1 \quad then:$$

$$\begin{cases} \hat{e}^{+}{}_{1} = 0 \\ \hat{e}^{+}{}_{2} = 0 \\ \hat{e}^{+}{}_{3} = 0 \\ \hat{e}^{+}{}_{4} = 0 \end{cases} \qquad (۳۵\text{-}۳)$$

$$if \quad -1 \le M_{\xi} \le 0 \quad then:$$

$$\begin{cases} \hat{e}^{+}{}_{1} = \dfrac{\rho}{2\gamma}(\bar{u}+c) \\ \hat{e}^{+}{}_{2} = \dfrac{\rho}{2\gamma}(\bar{u}+c)(u+n_{x,R}c) \\ \hat{e}^{+}{}_{3} = \dfrac{\rho}{2\gamma}(\bar{u}+c)(u+n_{y,R}c) \\ \hat{e}^{+}{}_{4} = \dfrac{\rho}{2\gamma}(\bar{u}+c)\left(\bar{u}c+\dfrac{u^{2}+v^{2}}{2}+\dfrac{c^{2}}{\gamma-1}\right) \end{cases} \qquad (۳۶\text{-}۳)$$

$$if \quad 0 \le M_\xi \le 1 \quad then:$$

$$\begin{cases} \hat{e}^+{}_1 = \rho\bar{u} + \dfrac{\rho}{2\gamma}(c-\bar{u}) \\ \hat{e}^+{}_2 = \rho u\bar{u} + \dfrac{\rho}{2\gamma}(\bar{u}-c)(-u+n_{x,R}c)+n_{x,R}p \\ \hat{e}^+{}_3 = \rho v\bar{u} + n_{y,R}p + \dfrac{\rho}{2\gamma}(-\bar{u}+c)(v-n_{y,R}c) \\ \hat{e}^+{}_4 = \bar{u}(e+p) + \dfrac{\rho}{2\gamma}(c-\bar{u})\left(\dfrac{u^2+v^2}{2} - \bar{u}c + \dfrac{c^2}{\gamma-1}\right) \end{cases} \qquad (۳۷-۳)$$

$$if \quad 1 \le M_\xi \le +\infty \quad then:$$

$$\begin{cases} \hat{e}^+{}_1 = \rho\bar{u} \\ \hat{e}^+{}_2 = \rho u\bar{u} + n_{x,R}p \\ \hat{e}^+{}_3 = \rho v\bar{u} + n_{y,R}p \\ \hat{e}^+{}_4 = \bar{u}(e+p) \end{cases} \qquad (۳۸-۳)$$

و اگر :

$$\hat{E}^-{}_i = \begin{bmatrix} \hat{e}^-{}_1 \\ \hat{e}^-{}_2 \\ \hat{e}^-{}_3 \\ \hat{e}^-{}_4 \end{bmatrix} \qquad (۳۹-۳)$$

باشد، به ازای مقادیر مختلف M_ξ داریم :

$$if \quad -\infty \le M_\xi \le -1 \quad then:$$

$$\begin{cases} \hat{e}^-{}_1 = \rho\bar{u} \\ \hat{e}^-{}_2 = \rho u\bar{u} + n_{x,R}p \\ \hat{e}^-{}_3 = \rho v\bar{u} + n_{y,R}p \\ \hat{e}^-{}_4 = \bar{u}(e+p) \end{cases} \qquad (۴۰-۳)$$

$$if \quad -1 \le M_{\xi} \le 0 \quad then:$$

$$\begin{cases} \hat{e}^{-}{}_1 = (\overline{u}+c)\dfrac{-\rho}{2\gamma} + \rho\overline{u} \\ \hat{e}^{-}{}_2 = \rho u\overline{u} + n_{x,R}p - \dfrac{\rho}{2\gamma}(\overline{u}+c)(u+n_{x,R}c) \\ \hat{e}^{-}{}_3 = \rho v\overline{u} + n_{y,R}p - \dfrac{\rho}{2\gamma}(\overline{u}+c)(v+n_{y,R}c) \\ \hat{e}^{-}{}_4 = \overline{u}(e+p) - \dfrac{\rho}{2\gamma}(\overline{u}+c)\left[\overline{u}c + \dfrac{u^2+v^2}{2} + \dfrac{c^2}{\gamma-1}\right] \end{cases} \qquad (۴۱-۳)$$

$$if \quad 0 \le M_{\xi} \le 1 \quad then:$$

$$\begin{cases} \hat{e}^{-}{}_1 = \dfrac{\rho}{2\gamma}(\overline{u}-c) \\ \hat{e}^{-}{}_2 = \dfrac{\rho}{2\gamma}(\overline{u}-c)(u-n_{x,R}c) \\ \hat{e}^{-}{}_3 = \dfrac{\rho}{2\gamma}(\overline{u}-c)(v-n_{y,R}c) \\ \hat{e}^{-}{}_4 = \dfrac{\rho}{2\gamma}(\overline{u}-c)\left(-\overline{u}c + \dfrac{u^2+v^2}{2} + \dfrac{c^2}{\gamma-1}\right) \end{cases} \qquad (۴۲-۳)$$

$$if \quad 1 \le M_{\xi} \le +\infty \quad then:$$

$$\begin{cases} \hat{e}^{-}{}_1 = 0 \\ \hat{e}^{-}{}_2 = 0 \\ \hat{e}^{-}{}_3 = 0 \\ \hat{e}^{-}{}_4 = 0 \end{cases} \qquad (۴۳-۳)$$

ب - تجزیه بردار شار $\hat{F}_i$

اگر شار $\hat{F}_i$ را نیز بر حسب مقادیر ویژه ماتریس ژاکوبین تبدیل $\hat{B}_i = \dfrac{\partial \hat{F}_i}{\partial \hat{Q}}$ مرتب کنیم آنگاه می‌توان آن را بر حسب علامت مقادیر ویژه ماتریس $\hat{B}_i$ به شارهای مثبت و منفی تجزیه نمود. در این حالت مقادیر ویژه ماتریس $\hat{B}_i$ عبارتند از :

$$\begin{cases} \hat{\lambda}_1 = \bar{v} \\ \hat{\lambda}_2 = \bar{v} \\ \hat{\lambda}_3 = \bar{v} + c \\ \hat{\lambda}_4 = \bar{v} - c \end{cases} \qquad (۳-۴۴)$$

اگر :

$$\hat{F}^{+}{}_i = \begin{bmatrix} \hat{f}^{+}{}_1 \\ \hat{f}^{+}{}_2 \\ \hat{f}^{+}{}_3 \\ \hat{f}^{+}{}_4 \end{bmatrix} \qquad (۳-۴۵)$$

باشد، آنگاه به ازای مقادیر مختلف M_η خواهیم داشت :

$$if \quad -\infty \le M_\eta \le -1 \quad then:$$

$$\begin{cases} \hat{f}^{+}{}_1 = 0 \\ \hat{f}^{+}{}_2 = 0 \\ \hat{f}^{+}{}_3 = 0 \\ \hat{f}^{+} \quad = 0 \end{cases} \qquad (۳-۴۶)$$

$$if \quad -1 \le M_\eta \le 0 \quad then:$$

$$\begin{cases} \hat{f}^{+}{}_1 = \dfrac{\rho}{2\gamma}(\bar{v} + c) \\ \hat{f}^{+}{}_2 = \dfrac{\rho}{2\gamma}(\bar{v} + c)(u + n_{x,T}c) \\ \hat{f}^{+}{}_3 = \dfrac{\rho}{2\gamma}(\bar{v} + c)(u + n_{y,T}c) \\ \hat{f}^{+}{}_4 = \dfrac{\rho}{2\gamma}(\bar{v} + c)\left(\bar{v}c + \dfrac{u^2 + v^2}{2} + \dfrac{c^2}{\gamma - 1}\right) \end{cases} \qquad (۳-۴۷)$$

$$if \quad 0 \le M_\eta \le 1 \quad then:$$

$$\begin{cases} \hat{f}^+{}_1 = \rho\bar{v} - \dfrac{\rho}{2\gamma}(\bar{v}-c) \\ \hat{f}^+{}_2 = \rho u\bar{v} + n_{x,T}p - \dfrac{\rho}{2\gamma}(\bar{v}-c)(u - n_{x,T}c) \\ \hat{f}^+{}_3 = \rho v\bar{v} + n_{y,T}p - \dfrac{\rho}{2\gamma}(\bar{v}-c)(v - n_{y,T}c) \\ \hat{f}^+{}_4 = (e+p)\bar{v} - \dfrac{\rho}{2\gamma}(\bar{v}-c)\left(-\bar{v}c + \dfrac{u^2+v^2}{2} + \dfrac{c^2}{\gamma-1}\right) \end{cases} \tag{۴۸-۳}$$

$$if \quad 1 \le M_\eta \le +\infty \quad then:$$

$$\begin{cases} \hat{f}^+{}_1 = \rho\bar{v} \\ \hat{f}^+{}_2 = \rho u\bar{v} + n_{x,T}p \\ \hat{f}^+{}_3 = \rho v\bar{v} + n_{y,T}p \\ \hat{f}^+{}_4 = (e+p)\bar{v} \end{cases} \tag{۴۹-۳}$$

و اگر :

$$\hat{F}^-{}_i = \begin{bmatrix} \hat{f}^-{}_1 \\ \hat{f}^-{}_2 \\ \hat{f}^-{}_3 \\ \hat{f}^-{}_4 \end{bmatrix} \tag{۵۰-۳}$$

باشد، به ازای مقادیر مختلف M_η، اجزای $\hat{F}_i^-$ از روابط زیر محاسبه می‌شوند :

$$if \quad -\infty \le M_\eta \le -1 \quad then:$$

$$\begin{cases} \hat{f}^-{}_1 = \rho\bar{v} \\ \hat{f}^-{}_2 = \rho u\bar{v} + n_{x,T}p \\ \hat{f}^-{}_3 = \rho v\bar{v} + n_{y,T}p \\ \hat{f}^-{}_4 = (e+p)\bar{v} \end{cases} \tag{۵۱-۳}$$

$$if \quad -1 \le M_\eta \le 0 \quad then:$$

$$\begin{cases} \hat{f}^-_1 = \rho\bar{v} - \dfrac{\rho}{2\gamma}(\bar{v}+c) \\ \hat{f}^-_2 = \rho u\bar{v} + n_{x,T}p - \dfrac{\rho}{2\gamma}(\bar{v}+c)(u+n_{x,T}c) \\ \hat{f}^-_3 = \rho v\bar{v} + n_{y,T}p - \dfrac{\rho}{2\gamma}(\bar{v}+c)(v+n_{y,T}c) \\ \hat{f}^-_4 = \bar{v}(e+p) - \dfrac{\rho}{2\gamma}(\bar{v}+c)\left(\bar{v}c + \dfrac{u^2+v^2}{2} + \dfrac{c^2}{\gamma-1}\right) \end{cases} \tag{۵۲-۳}$$

$$if \quad 0 \le M_\eta \le 1 \quad then:$$

$$\begin{cases} \hat{f}^-_1 = \dfrac{\rho}{2\gamma} \\ \hat{f}^-_2 = \dfrac{\rho}{2\gamma}(\bar{v}-c)(u-n_{x,T}c) \\ \hat{f}^-_3 = \dfrac{\rho}{2\gamma}(\bar{v}-c)(v-n_{y,T}c) \\ \hat{f}^-_4 = \dfrac{\rho}{2\gamma}(\bar{v}-c)\left(-\bar{v}c + \dfrac{u^2+v^2}{2} + \dfrac{c^2}{\gamma-1}\right) \end{cases} \tag{۵۳-۳}$$

$$if \quad 1 \le M_\eta \le +\infty \quad then:$$

$$\begin{cases} \hat{f}^-_1 = 0 \\ \hat{f}^-_2 = 0 \\ \hat{f}^-_3 = 0 \\ \hat{f}^-_4 = 0 \end{cases} \tag{۵۴-۳}$$

۳-۱-۲-۲- تجزیه ماتریس ژاکوبین‌های تبدیل شارهای غیر لزج

این ماتریسها را نیز می‌توان به روشهای استگر - وارمینگ به اجزای مثبت و منفی تجزیه کرد به طوری که داشته باشیم :

$$\begin{cases} \hat{A}_i = \hat{A}_i^{\,+} + \hat{A}_i^{\,-} \\ \hat{B}_i = \hat{B}_i^{\,+} + \hat{B}_i^{\,-} \end{cases} \qquad (۵۵\text{-}۳)$$

اگر فرض شود :

$$\hat{A}_i^{\,+} = \begin{bmatrix} a_{11}^+ & a_{12}^+ & a_{13}^+ & a_{14}^+ \\ a_{21}^+ & a_{22}^+ & a_{23}^+ & a_{24}^+ \\ a_{31}^+ & a_{32}^+ & a_{33}^+ & a_{34}^+ \\ a_{41}^+ & a_{42}^+ & a_{43}^+ & a_{44}^+ \end{bmatrix} \quad , \quad \hat{A}_i^{\,-} = \begin{bmatrix} a_{11}^- & a_{12}^- & a_{13}^- & a_{14}^- \\ a_{21}^- & a_{22}^- & a_{23}^- & a_{24}^- \\ a_{31}^- & a_{32}^- & a_{33}^- & a_{34}^- \\ a_{41}^- & a_{42}^- & a_{43}^- & a_{44}^- \end{bmatrix} \qquad (۵۶\text{-}۳)$$

و نیز :

$$\hat{B}_i^{\,+} = \begin{bmatrix} b_{11}^+ & b_{12}^+ & b_{13}^+ & b_{14}^+ \\ b_{21}^+ & b_{22}^+ & b_{23}^+ & b_{24}^+ \\ b_{31}^+ & b_{32}^+ & b_{33}^+ & b_{34}^+ \\ b_{41}^+ & b_{42}^+ & b_{43}^+ & b_{44}^+ \end{bmatrix} \quad , \quad \hat{B}_i^{\,-} = \begin{bmatrix} b_{11}^- & b_{12}^- & b_{13}^- & b_{14}^- \\ b_{21}^- & b_{22}^- & b_{23}^- & b_{24}^- \\ b_{31}^- & b_{32}^- & b_{33}^- & b_{34}^- \\ b_{41}^- & b_{42}^- & b_{43}^- & b_{44}^- \end{bmatrix} \qquad (۵۷\text{-}۳)$$

آنگاه عضوهای این ماتریسها عبارتند از :

$$a_{ij}^+ = \frac{\partial \hat{e}_i^+}{\partial q_j} \quad , \quad a_{ij}^- = \frac{\partial \hat{e}_i^-}{\partial q_j} \quad , \quad b_{ij}^+ = \frac{\partial \hat{f}_i^+}{\partial q_j} \quad , \quad b_{ij}^- = \frac{\partial \hat{f}_i^-}{\partial q_j} \quad , \quad \begin{cases} i = 1 \; to \; 4 \\ j = 1 \; to \; 4 \end{cases} \qquad (۵۸\text{-}۳)$$

و مقادیر $\hat{f}_i^- , \hat{f}_i^+ , \hat{e}_i^- , \hat{e}_i^+$ نیز اجزای مثبت و منفی شارهای غیر لزج هستند که توسط روشهای استگر- وارمینگ در همین فصل محاسبه شده اند.

با محاسبه مقادیر $b_{ij}^{\pm} , a_{ij}^{\pm}$ و قرار دادن آنها در روابط (۳-۵۶) و (۳-۵۷) می‌توان اجزای تجزیه شده ماتریسهای ژاکوبین‌های تبدیل را محاسبه نمود که در اینجا به علت بالا رفتن حجم محاسبات از انجام این کار صرف نظر می‌شود.

۳-۲- شرایط مرزی

انواع شرایط مرزی در جریان سیال را می‌توان به صورت زیر دسته بندی نمود :

۱- دیواره ها (*Walls*)

۲- مرز ورود جریان (*Inflow Boundary*)

۳- مرز خروج جریان (*Outflow Boundary*)

۴- محور تقارن (*Symmetry Line*)

در حالت کلی تعداد متغیرهای برای جریان وجود دارد به عنوان مثال :

$$p_0 , T_0 , p , u , v , T , e , M , c , h , \rho , \rho u , \rho v , \ldots$$

در هر صورت روی مرزها تعداد کافی از متغیر ها باید مشخص شوند و این متغیرها باید مستقل خطی باشند در غیر این صورت شرایط مرزی به تعداد کافی مشخص نخواهد شد. معلوم شدن تعداد شرایط مرزی مورد نیاز بسته به نوع مرز است و یا با برون یابی بدست می‌آید.

از لحاظ تحلیلی با رعایت اصل استقلال خطی هیچ فرقی بین انتخاب متغیرها وجود ندارد ولی به لحاظ عددی انتخاب نامطلوب متغیرهای جریان باعث بوجود آمدن موجهای عددی بر روی مرزها شده که اثر آن ممکن است به داخل میدان حل نفوذ کرده و باعث مخدوش شدن دقت حل و یا همگرایی آن شود.

روش کلی اعمال شرایط مرزی چنین است که با در نظر گرفتن سلول مجازی[۱] در مجاورت مرز حقیقی، متغیرهای جریان بر سلول مجازی را چنان تعیین می‌کنیم که شرط مرزی بر روی مرز فیزیکی به درستی اعمال شود.

۳-۲-۱- مرز جامد

برای مرز جامد می‌توان دو نوع دیواره به صورت زیر تعریف نمود :

۱- دیواره با برش کامل[۲]

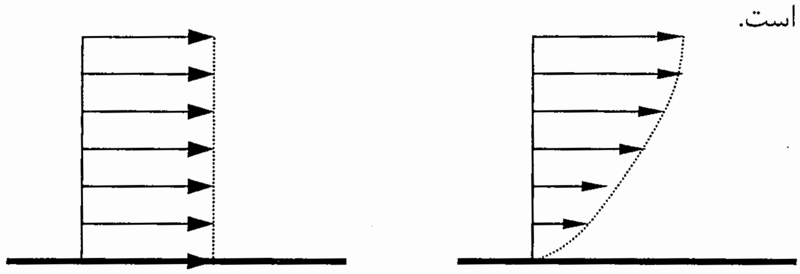
1- Fictitious Cell

2- Full Shear

3- No shear

۲- دیواره بدون برش[۳]

نمایی از دیواره با برش کامل و دیواره بدون برش در شکل (۳-۱) به ترتیب در سمت چپ و راست نشان داده شده است.

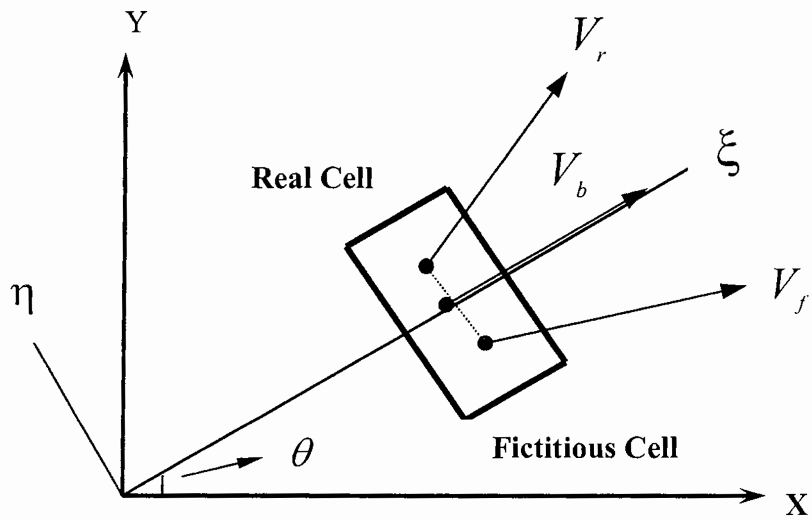


Free Slip (Full Shear) B.C.   No Slip (No Shear) B.C.

شکل (۳-۱): نمایی از دیواره‌های با برش و بدون برش روی مرز

در هر یک از دیواره‌های فوق مولفه قائم سرعت صفر می‌باشد : $V_n = 0$

برای بدست آوردن سرعت مماسی در سلول مجازی به منظور اعمال درست شرط مرزی، دو دستگاه مختصات (x, y) و (ξ, η) را مطابق شکل (۳-۲) نسبت به هم در نظر بگیرید :



شکل(۳-۲): بدست آوردن سرعت مماسی در سلول مجازی

مطابق این شکل می‌توان روابط زیر را برای مولفه‌های بردارهای عمود و مماس بر مرز نوشت :

$$\hat{t}=\begin{Bmatrix}t_x\\t_y\end{Bmatrix}=\begin{Bmatrix}cos\theta\\sin\theta\end{Bmatrix} \qquad ,\hat{n}=\begin{Bmatrix}n_x\\n_y\end{Bmatrix}=\begin{Bmatrix}-sin\theta\\cos\theta\end{Bmatrix}$$

بنابر این :

$$\begin{cases}\hat{n}=\hat{i}n_x+\hat{j}n_y=-\hat{i}\,sin\theta+\hat{j}\,cos\theta\\ \hat{t}=\hat{i}t_x+\hat{j}t_y=\hat{i}\,cos\theta+\hat{j}\,sin\theta\end{cases} \tag{۵۹-۳}$$

با استفاده از دو رابطه بالا مولفه‌های عمودی و مماسی سرعتهای سلول حقیقی، سلول مجازی و گره مرزی به صورت زیر محاسبه می‌شوند :

$$\begin{cases}V_{r,n}=\vec{V}_r.\hat{n}=\left(\hat{i}u_r+\hat{j}v_r\right)\!.\!\left(\hat{i}n_x+\hat{j}n_y\right)=u_rn_x+v_rn_y\\ V_{r,t}=\vec{V}_r.\hat{t}=\left(\hat{i}u_r+\hat{j}v_r\right)\!.\!\left(\hat{i}t_x+\hat{j}t_y\right)=u_rt_x+v_rt_y\end{cases} \tag{۶۰-۳}$$

$$\begin{cases}V_{f,n}=\vec{V}_f.\hat{n}=\left(\hat{i}u_f+\hat{j}v_f\right)\!.\!\left(\hat{i}n_x+\hat{j}n_y\right)=u_fn_x+v_fn_y\\ V_{f,t}=\vec{V}_f.\hat{t}=\left(\hat{i}u_f+\hat{j}v_f\right)\!.\!\left(\hat{i}t_x+\hat{j}t_y\right)=u_ft_x+v_ft_y\end{cases} \tag{۶۱-۳}$$

$$\begin{cases}V_{b,n}=\vec{V}_b.\hat{n}=\left(\hat{i}u_b+\hat{j}v_b\right)\!.\!\left(\hat{i}n_x+\hat{j}n_y\right)=u_bn_x+v_bn_y\\ V_{b,t}=\vec{V}_b.\hat{t}=\left(\hat{i}u_b+\hat{j}v_b\right)\!.\!\left(\hat{i}t_x+\hat{j}t_y\right)=u_bt_x+v_bt_y\end{cases} \tag{۶۲-۳}$$

حال سرعتهای عمودی و مماسی سلول مجازی را طوری تعیین می‌کنیم که شرط مرزی مورد نظر روی دیواره ارضا شود.

۳-۲-۱-۱- مرز جامد بدون برش

روی این مرز نیز هر دو مولفه سرعت دیواره برابر صفر می‌باشد.

$$V_{b,t}=V_{b,n}=0$$

بنابر این شرط مرزی چنین اعمال می‌شود :

$$\vec{V}_f=-\vec{V}_r$$

که از این رابطه نتیجه می‌شود :

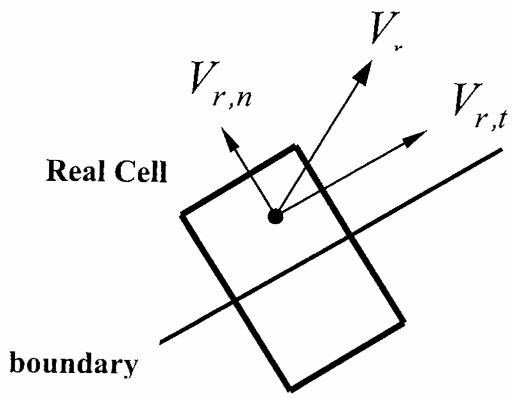
$$\begin{cases} u_f = -u_r \\ v_f = -v_r \end{cases} \qquad (۶۳-۳)$$

برای T, p نیز با برون یابی خواهیم داشت :

$$\begin{cases} p_f = p_r \\ T_f = T_r \end{cases} \qquad (۶۴-۳)$$

۳-۲-۲- مرز ورود جریان

مطابق شکل (۳-۳) برای این مرز سرعت روی مرز دارای دو مولفه می‌باشد.



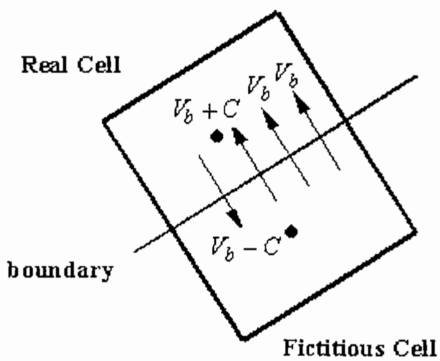
شکل (۳-۳):مرز ورود جریان

اهیت این مرز به گونه‌ای است که مشخصه‌های سیستم از آن قابل عبور می‌باشند و مقدار $V_{b,n}$ تعیین می‌کند که چند مشخصه به داخل ناحیه وارد و چند مشخصه از آن خارج می‌شود. این مشخصه‌ها عبارتند از $V_{b,n}+c$ ، $V_{b,n}-c$ ، $V_{b,n}$ ، $V_{b,n}$.

به تعداد مشخصه‌هایی که از مرز خارج می‌شود متغیر جریان باید برونیابی شود و به تعداد مشخصه‌هایی که به مرز وارد می‌شود باید متغیر جریان در سلول مجازی از قبل تعیین شود. بنابر این دو عامل ورود و یا خروج مشخصه ها و نیز علامت مشخصه ها نحوه اعمال شرط مرزی را تعیین می‌کند.

۳-۲-۲-۱- مرز ورود جریان زیر صوتی[1]

در جریان زیر صوتی $V_{b,n} < c$ می‌باشد بنابر این $V_{b,n} - c < 0$ می‌باشد. لذا از این مرز سه مشخصه وارد و یک مشخصه خارج می‌شود (مطابق شکل (۳-۴)).



شکل(۳-۴): مرز ورود جریان زیر صوتی

حال می‌توان سه متغیر جریان را که مستقل خطی هستند را به صوت دلخواه انتخاب کرد. به عنوان مثال می‌توان p, u, v را انتخاب نمود. با انتخاب این متغیر ها خواهیم داشت :

1- Subsonic Inflow Boundary

$$u_f = u_b$$
$$v_f = v_b \qquad (۶۵-۳)$$
$$p_f = p_b(static\ pressure)$$

متغیر چهارم با برونیابی انجام می‌شود :

$$T_f = T_r \qquad (۶۶-۳)$$

می‌توان P_0 ,T_0 و α (زاویه جریان) را به عنوان سه متغیر مستقل خطی انتخاب کرد. با انتخاب α, T_0, P_0 به عنوان سه متغیر مستقل، این مقادیر را از قبل تعیین کرده و سرعت در سلول مجازی را نیز با برونیابی تعیین می‌کنیم. بنابر این :

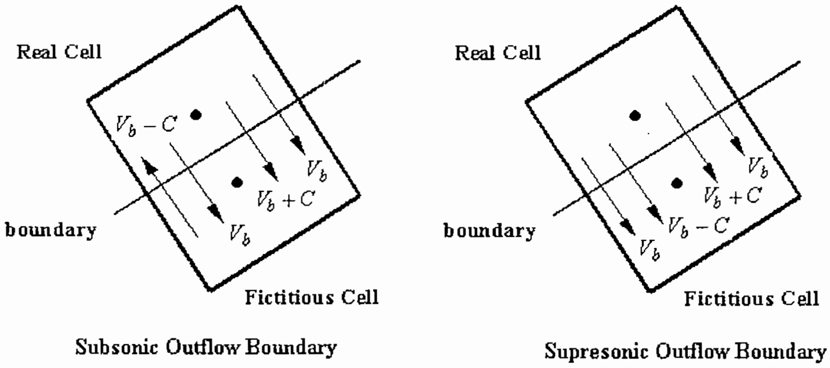
$$\begin{cases} u_f = \left|\vec{V}_r\right| cos\theta \\ v_f = \left|\vec{V}_r\right| sin\theta \end{cases} \qquad (۶۷-۳)$$

که در این رابطه $\left|\vec{V}_r\right| = \sqrt{\left(u_r^2 + v_r^2\right)}$ می‌باشد.

۳-۲-۳- مرز خروج جریان[1]

بر روی این مرز نیز در جریان زیر صوتی سه مشخصه خارج و یک مشخصه به مرز وارد می‌شود. بنابر این باید سه متغیر جریان به صورت برونیابی از داخل ناحیه تعیین شده و یک متغیر از قبل تعیین شده باشد. در جریان ما فوق صوت هر چهار مشخصه از مرز خارج می‌شوند لذا باید چهار متغیر مستقل از داخل ناحیه برونیابی می‌شوند. شکل (۳-۵) مرز خروج جریان را در دو حالت زیر صوت و مافوق صوت نشان می‌دهد.

1- Outflow boundary



شکل(۳-۵): شرط مرزی خروج جریان در حالت زیر صوت و مافوق صوت

۳-۲-۴- محور تقارن

بر روی این مرز نیز مولفه قائم سرعت و نیز گرادیان متغیرها در جهت نرمال برابر صفر می‌باشد. بنابر این شرط نیز همانند مرز جامد با برش کامل می‌باشد.

۳-۳- الگوریتم حل

روشهای متعددی برای حل معادلات ناویر - استوکس وجود دارد که در بین آنها مـی‌تـوان بـه روش صریح مک کورمک و روشهای تجزیه بردار شار با فرمولبندی صریح و یا ضمنی اشاره کـرد [۱۰و۱۷] . در این کار از روش تجزیه بردار شار با فرمول بندی ضمنی استفاده شده است.

در فصل دوم معادلات حاکم بر حرکت سیال در فضای محاسباتی به صورت زیر استخراج شد :

$$\hat{Q}_t + \hat{E}_\xi + \hat{F}_\eta = \hat{H} \qquad (۶۸\text{-}۳)$$

با تقریب مشتقات زمانی به صورت پیشرو خواهیم داشت :

$$\frac{\hat{Q}^{n+1} - \hat{Q}^n}{\Delta t} + \left[\hat{E}_\xi\right]^{n+1} + \left[\hat{F}_\eta\right]^{n+1} = \left[\hat{H}\right]^{n+1} \qquad (۶۹\text{-}۳)$$

معادله (۳-۶۹) غیر خطی است و از این رو از یک روش خطی سازی باید اسـتفاده کـرد. در اینجـا از بسط سری تیلور برای خطی سازی استفاده می‌شود که در این صورت روابط زیر استخراج می‌شوند :

$$\hat{E}^{n+1} = \hat{E}^n + \frac{\partial \hat{E}^n}{\partial \hat{Q}} \Delta \hat{Q} = \hat{E}^n + A \Delta \hat{Q} \tag{۷۰-۳}$$

$$\hat{F}^{n+1} = \hat{F}^n + \frac{\partial \hat{F}^n}{\partial \hat{Q}} \Delta \hat{Q} = \hat{F}^n + B \Delta \hat{Q} \tag{۷۱-۳}$$

$$\hat{H}^{n+1} = \hat{H}^n + \frac{\partial \hat{H}^n}{\partial \hat{Q}} \Delta \hat{Q} = \hat{H}^n + C \Delta \hat{Q} \tag{۷۲-۳}$$

با قرار دادن روابط فوق در رابطه (۳-۶۹) داریم :

$$\frac{\Delta \hat{Q}}{\Delta t} + \frac{\partial}{\partial \xi}\left[\hat{E}^n + A \Delta \hat{Q}\right] + \frac{\partial}{\partial \eta}\left[\hat{F}^n + B \Delta \hat{Q}\right] = \left[\hat{H}^n + C \Delta \hat{Q}\right] \tag{۷۳-۳}$$

حال این معادله را به صورت زیر باز نویسی می‌کنیم :

$$\left\{ I + \Delta t \left[\frac{\partial}{\partial \xi}(A) + \frac{\partial}{\partial \eta}(B) - C \right] \right\} \Delta \hat{Q} = -\Delta t \left[\frac{\partial \hat{E}^n}{\partial \xi} + \frac{\partial \hat{F}^n}{\partial \eta} + \hat{H}^n \right] \tag{۷۴-۳}$$

$$A = A_i - A_v$$

$$B = B_i - B_v$$

$$C = C_i - C_v$$

ماتریس ضرایب دستگاه فوق یک بلوک پنج قطری است و حل آن دشوار می‌باشد. بنابر این برای راحتی کار و افزایش راندمان، می‌توان دستگاه را به دو دستگاه معادلات بلوکی سه قطری که به دنبال هم حل می‌شوند، تقریب زد.

یکی از این روشها روش فاکتور گیری تقریبی[1] است. در مسائل دو بعدی روش فاکتور گیری تقریبی جواب پایداری ایجاد می‌کند ولی در مسائل سه بعدی این روش ناپایدار می‌باشد و برای غلبه بر ناپایداری باید از لزجت مصنوعی استفاده کرد ولی در حالت کلی افزودن لزجت مصنوعی نامطلوب است و در صورت امکان باید از آن پرهیز کرد [۳و۱۴] .

1- Approximation Factorization

$$\left\{I+\Delta t\left[\frac{\partial A_i}{\partial \xi}-\frac{\partial A_v}{\partial \xi}\right]\right\}\left\{I+\Delta t\left[\frac{\partial B_i}{\partial \eta}-\frac{\partial B_v}{\partial \eta}+\Delta t(C_i-C_v)\right]\right\}\Delta\hat{Q}= \tag{۷۵-۳}$$
$$-\Delta t\left[\frac{\partial \hat{E}_i''}{\partial \xi}+\frac{\partial \hat{F}_i''}{\partial \eta}+\frac{\partial \hat{E}_v''}{\partial \xi}+\frac{\partial \hat{F}_v''}{\partial \eta}+(\hat{H}_i''-\hat{H}_v'')\right]$$

پس از تجزیه متریک های ژاکوبین و بردارهای شار داریم:

$$\left[I+\Delta t\frac{\partial}{\partial \xi}\left[A_i^+ + A_i^-\right]-\Delta t\frac{\partial A_v}{\partial \xi}\right]\left[I+\Delta t\frac{\partial}{\partial \xi}\left[A_i^+ + A_i^-\right]-\Delta t\frac{\partial A_v}{\partial \xi}+\Delta t(C_i-C_v)\right]\Delta\hat{Q}= \tag{۷۶-۳}$$
$$-\Delta t\left[\frac{\partial}{\partial \xi}\left[\hat{E}_i^+ - \hat{E}_v^-\right]+\frac{\partial}{\partial \eta}\left[\hat{F}_i^+ - \hat{F}_v^-\right]-\frac{\partial \hat{E}_v}{\partial \xi}-\frac{\partial \hat{F}_v}{\partial \eta}+(\hat{H}_i-\hat{H}_v)\right]$$

معادله فوق از مراحل پی در پی زیر حل می شود:

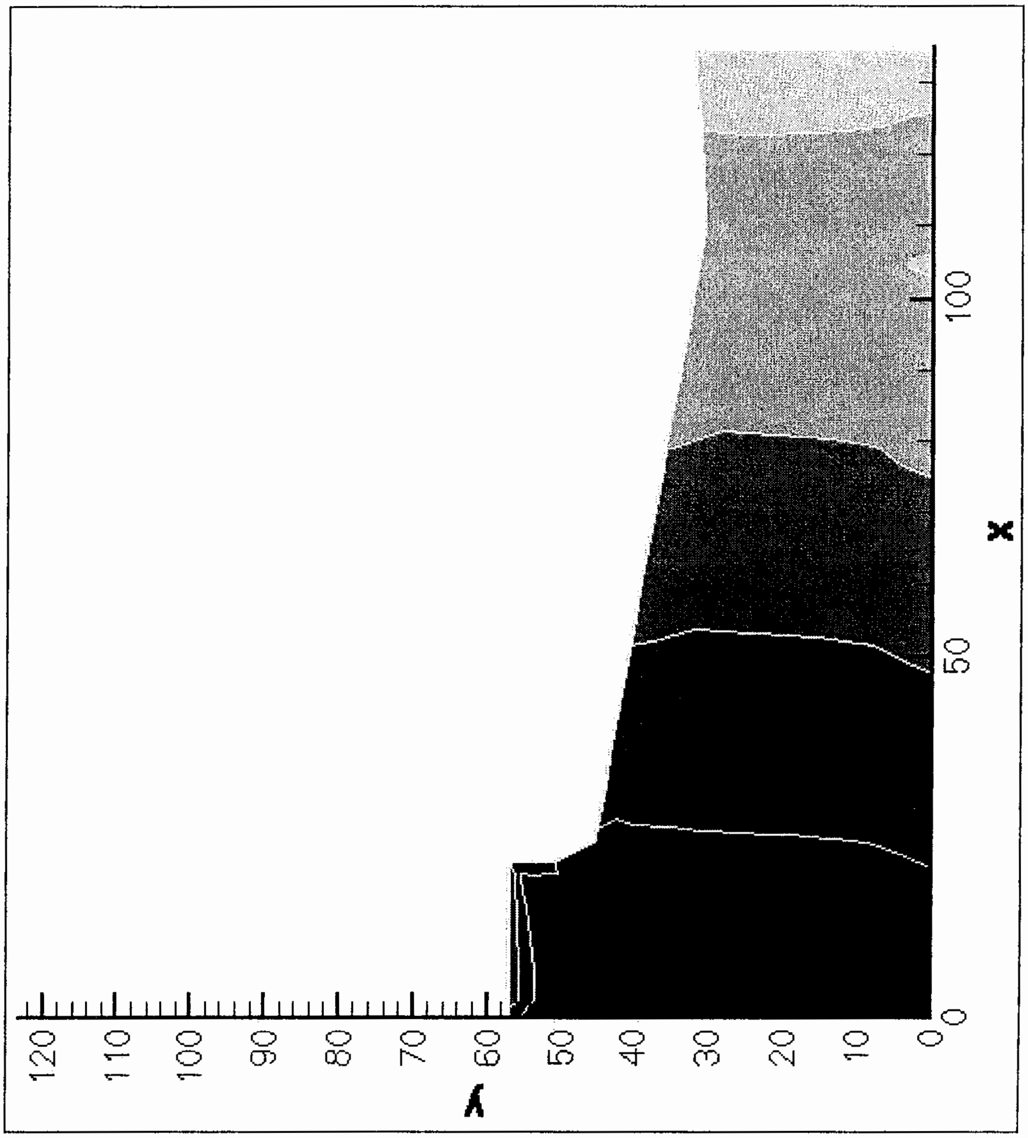
$$\left[I+\Delta t\frac{\partial}{\partial \xi}\left[A_i^+ + A_i^-\right]-\Delta t\frac{\partial A_v}{\partial \xi}\right]\Delta\hat{Q}^{*}= \tag{۷۷-۳}$$
$$-\Delta t\left[\frac{\partial}{\partial \xi}\left[\hat{E}_i^+ - \hat{E}_v^-\right]+\frac{\partial}{\partial \eta}\left[\hat{F}_i^+ - \hat{F}_v^-\right]-\frac{\partial \hat{E}_v}{\partial \xi}-\frac{\partial \hat{F}_v}{\partial \eta}+(\hat{H}_i-\hat{H}_v)\right]$$

و

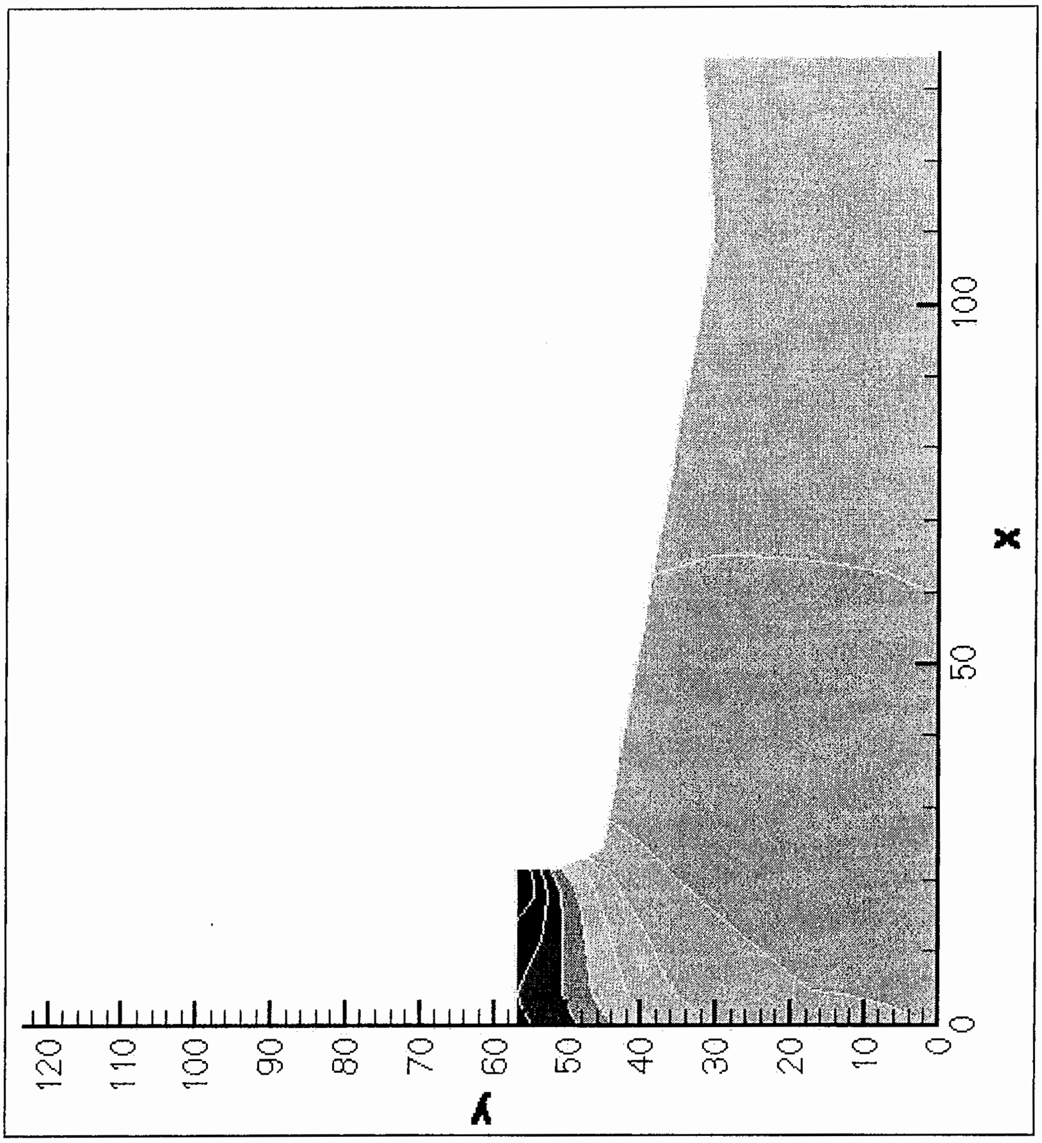
$$\left[I+\Delta t\frac{\partial}{\partial \xi}\left[A_i^+ + A_i^-\right]-\Delta t\frac{\partial A_v}{\partial \xi}+\Delta t(C_i-C_v)\right]\Delta\hat{Q}=\Delta\hat{Q}^{*} \tag{۷۸-۳}$$

۳-۴- ارائه نتایج

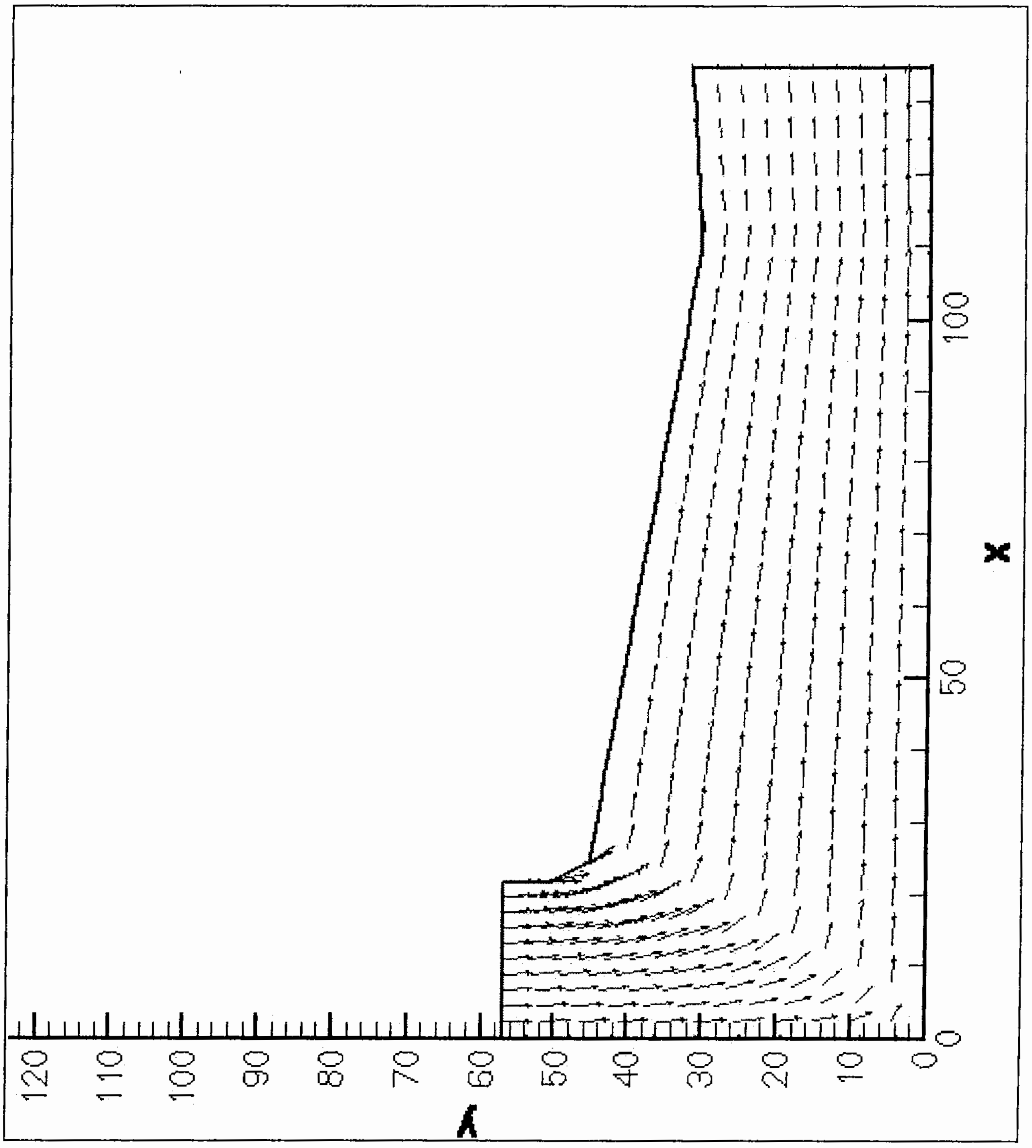
پس از حل معادلات نتایج زیر برای یک برج خنک کن بدست آمده است که کانتورهای فشار، دما و بردارهای سرعت در شکلهای (۳-۶)،(۳-۷) و (۳-۸) ملاحظه می شوند. با توجه به کانتورهای دما ملاحظه می شود که تغییرات دما فقط در ناحیه ورودی برج که مبدلهای حرارتی قرار دارند رخ می دهد.



شکل(۳-۶): کانتورهای فشار



شکل(۳-۷): کانتورهای دما



شکل(۳-۸): بردارهای سرعت

فصل چهارم

حل معادلات سه بعدی با استفاده از FLUENT

۴-۱- فرضیات

برای حل معادلات با استفاده از نرم افزار Fluent فرضیات زیر انجام گرفته است :

۱ـ مدل جریان در هنگام وزش باد ، سه بعدی می باشد .

۲ـ بدلیل وجود جابجایی طبیعی می توان از فرض Boussinesq و یا گازایده آل غیر قابل تراکم استفاده کرد .

۳ـ بدلیل عدم وجود جریان چرخشی بالا از مدل توربولانس k-ε استفاده شده است .

۴ـ پروفیل سرعت باد به صورت یکنواخت در نظر گرفته شده است .

۵ـ از تغییرات لحظه ای سرعت باد صرف نظر شده است .

۶ـ مبدلهای حرارتی به صورت یک محیط متخلخل (porous) به همراه یک منبع حرارتی (Source Term) در نظر گرفته شده است .

۷ـ مدل مورد استفاده ، مدل یک برج واقعی در مقیاس صنعتی یعنی برج خنک کن نیروگاه شازند اراک می باشد .

۴-۲- معادلات حاکم

با توجه به فرضیات ذکر شده معادلات حاکم بر جریان داخل و اطراف برج خنک کن خشک تحت شرایط باد متقاطع به شکل برداری زیر خواهند بود .

$$\nabla \cdot V = 0 \qquad (۱-۴)$$

$$(V \cdot \nabla)V = -\frac{1}{\rho}\nabla P + \nabla \cdot (\frac{\sigma}{\rho}) - \beta(T - T_a)g + F \qquad (۲-۴)$$

$$\rho(V\cdot\nabla)T=-\nabla\cdot\left[(\Gamma+\Gamma_t)\nabla T\right]+Q \qquad (۳-۴)$$

که V نمایشگر بردار سرعت ، T دما و σ تا نسور تنش است که به وسیله فرمول زیر بیان می‌شود

:

$$\sigma_{ij}=(\mu+\mu_t)\cdot S_{ij} \qquad (۴-۴)$$

$$S_{ij}=\frac{1}{2}\left(\frac{\partial V_i}{\partial x_j}+\frac{\partial V_i}{\partial x_i}\right)$$

μ و μ_t به ترتیب لزجت مولکولی و توربولانس ، β ضریب انبساط حجمی هوا و T_a دمای هوای محیط است .

ترم F در معادله ممنتوم در برگیرنده افت فشار هوا در هنگام عبور از رادیاتورها بوده که این ترم فقط در ورودی برج ظاهر می شود .

ترم Q در معادله انرژی ، مقدار حرارت منتقل شده به سیال از رادیاتورهاست . میزان Q در حالت جابجایی طبیعی (بدون وزش باد) دارای مقداری ثابت بوده و به طور یکنواخت در ناحیه ورودی برج توزیع می شود . اما در هنگام وزش باد (جابجایی اجباری) بدلیل اختلال در جریان ورودی برج و عواملی که در این فصل ذکر خواهند شد ، توزیع Q در ورودی برج دیگر یکنواخت نبوده و حرارت داده شده به سیال در نقاط مختلف در ورودی برج متفاوت است .

اما به منظور مدل سازی صرفاً جریان داخل برج و نه از نقطه نظر انتقال حرارت و اینکه با بررسی های به عمل آمده ، آنچه باعث توزیع یکنواخت Q می شود ،اختلال در جریان ورودی برج است ، از اثر توزیع غیر یکنواخت Q در ورودی برج صرف نظر شده و یک مقدار ثابت و یکنواخت Q منظور می شود .

همچنین به دلیل مخلوط شدن هوا بلافاصله پس از ناحیه ورودی (رادیاتورها) وضعیت جریان داخل برج تا حوالی ناحیه خروجی آن تقریباً به صورت متقارن محوری در می آید . بعلاوه اگر مشکل اختلال در جریان به نحوی حل شود ، دیگر توزیع Q غیر یکنواخت نخواهد بود . Γ و Γ_t در معادله انرژی بیانگر ضریب انتقال حرارت هدایتی مولکولی و توربولانس هستند .

$$\Gamma = \frac{\mu}{\Pr} \qquad \Gamma_t = \frac{\mu_t}{\Pr_t}, \mu_t = \rho c_\mu \frac{k^2}{\varepsilon}$$

جریان داخل و اطراف برج خنک کن در هنگام وزش باد ،یک جریان مغشوش است و با استفاده از مدل k-ε ، معادلات توربولانس به صورت زیر بیان می شوند :

$$(V \cdot \nabla)k = \nabla.[(\upsilon + \upsilon_t/\sigma_k)\nabla k] + P + G - \varepsilon \tag{۵-۴}$$

$$(V \cdot \nabla)\varepsilon = \nabla.[(\upsilon + \upsilon_t/\sigma_\varepsilon)\nabla k] + c_{1\varepsilon}\frac{\varepsilon}{k}(P+G) - c_{2\varepsilon}\frac{\varepsilon^2}{k} \tag{۶-۴}$$

$$\upsilon_t = \frac{1}{\rho}\mu_t = c_\mu \frac{k^2}{\varepsilon} \tag{۷-۴}$$

که p انرژی سینتیک تولید شده به وسیله توربولانس و G به وسیله نیروی غوطه وری هستند و توسط روابط زیر محاسبه می شوند :

$$P = \nu_t S_{ij} S_{ij} \tag{۸-۴}$$

$$G = -g\beta \frac{\nu_t \partial T}{\sigma_t \partial z} \tag{۹-۴}$$

ضرایب ثابت در مدل توربولانس به شرح زیر هستند :

$c_{1\varepsilon} = 1.44$	$c_\mu = 0.09$
$\sigma_k = 1$	$c_{2\varepsilon} = 1.92$
$\sigma_t = 1$	$\sigma_\varepsilon = 1.3$

۴-۳- مدلسازی جریان

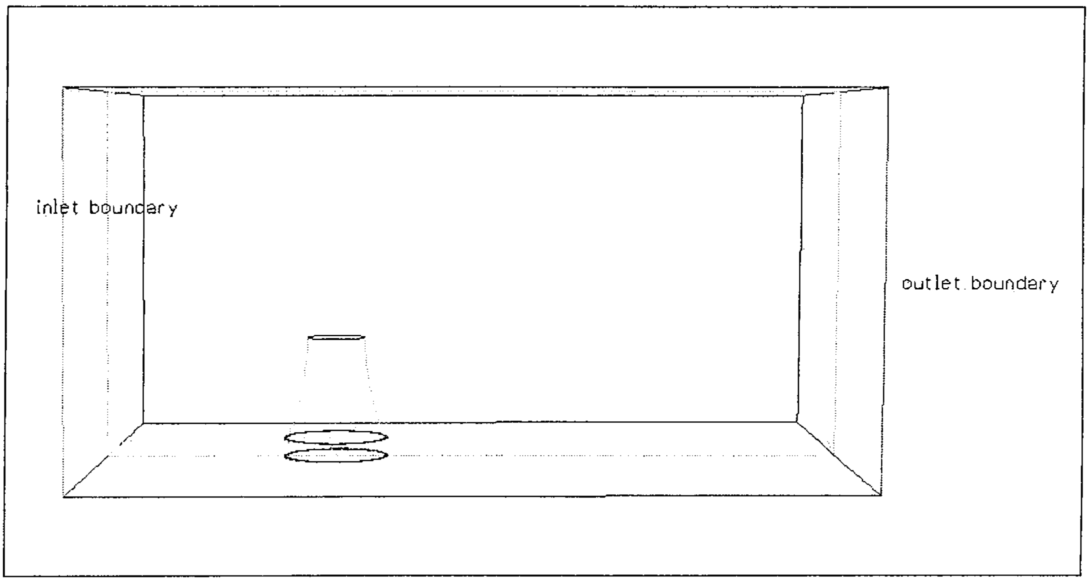
در این حالت از روش حجم محدود برای گسسته سازی معادلات و روش ضمنی برای حل معادلات گسسته شده جبری استفاده شده است . از الگوریتم SIMPLE برای محاسبه فشار و میدان جریان استفاده می شود . دامنه محاسباتی شامل ۳۰۷۷۰۰ مش بی سازمان (unstructured) بوده و روش upwind مرتبه یک برای مجزاسازی معادلات حاکم بکار گرفته شده است . شبکه تولید شده

۴-۳-۱- فضای فیزیکی و محاسباتی

مدل مورد استفاده در این تحقیق ، مدل یک برج واقعی در مقیاس صنعتی با ابعاد زیر است که مربوط به نیروگاه حرارتی شازند اراک می باشد :

حرارت دفع شده	۴۰۴ مگاوات
قطر پایین برج	۱۱۰ متر
ارتفاع برج	۱۳۰ متر
قطر گلوگاه	۶۲ متر
ارتفاع مبدلها	۲۰ متر
درجه حرارت طراحی محیط	۱۵ درجه سانتی گراد

فضای اطراف برج خنک کن همانطور که در شکل (۴-۲) ملاحظه می شود به صورت یک مکعب مستطیل با ابعاد m ۴۰۰×۴۰۰×۷۰۰ در نظر گرفته شده است که دارای یک مرز ورودی و یک خروجی است .



شکل(۴-۲): فضای محاسباتی

۴-۳-۲- مدلسازی مبدلهای حرارتی

مبدلهای حرارتی در ناحیه ورودی برج به صورت یک محیط متخلخل (porous) در نظر گرفته شده اند .

برای تعریف ناحیه متخلخل در نرم افزار Fluent از مدل Power Law در محیطهای متخلخل استفاده شده است که بر طبق آن

$$\frac{dp}{dx} = C_0 \times V^{C_1} \qquad (۱۰-۴)$$

کهdp : افت فشار هوا از روی رادیاتورها

dx: اندازه عرض مبدل حرارتی در جهت جریان هوای خنک کن

V :سرعت هوای عبوری

C_0, C_1 :مقادیر ثابت

برای مدل کردن ناحیه مبدل حرارتی نیاز به یافتن مقادیر C_0, C_1 و دادن آنها به عنوان ورودی به نرم افزار داریم ، برای این منظور با توجه به روابط ارائه شده از سوی شرکت EGI (کمپانی سازنده مبدلهای حرارتی برجهای خنک کن خشک) برای افت فشار در یک مبدل حرارتی داریم:

$$\Delta p_a = 0.158(L_1.C_k^{\ 0.5})^{1.76} \qquad (۱۱-۴)$$

$$L_3 = L_1.C_k^{0.5} \qquad (۱۲-۴)$$

$$\Delta p_a = 0.158 L_3^{\ 1.76} \qquad (۱۳-۴)$$

که در این روابط :

Δp_a :افت فشار طرف هوا بر حسب $\frac{kg}{m^2}$

L_1 :گذر هوای عبوری از واحد سطح حرارتی بر حسب $\frac{ton}{m^2.hr}$

C_k :ضریب تصحیح

L_3 گذر تصحیح شده هوای عبوری از روی لوله های مبدل حرارتی بر حسب $\frac{ton}{m^2.hr}$

بنابر توصیه شرکت EGI مقدار ۱۰٪ به مقدار افت فشار در رابطه قبل افزوده می شود بنابراین داریم :

$$\Delta p_a = 0.174 L_3^{1.76} \qquad (۱۴-۴)$$

حال برای محاسبه ضرایب C_0, C_1 داریم :

$$L_3 = \rho \times V \times 3.6 \qquad (۱۵-۴)$$

$$\frac{\Delta p}{\Delta x} = \frac{g}{\Delta x} \times 0.174 \times L_3{}^{1.76} \qquad (۱۶-۴)$$

$$\frac{\Delta p}{\Delta x} = \frac{g}{\Delta x} \times 0.174 \times (3.6 \times \rho)^{1.76} \times V^{1.76} = C_0 \times V^{C_1} \qquad (۱۷-۴)$$

بنا بر این خواهیم داشت:

$$C_0 = \frac{0.174 g}{\Delta x} (3.6 \times \rho)^{1.76}$$

$$C_1 = 1.76$$

در روابط فوق:

g :شتاب گرانش

Δx: عمق مبدل حرارتی در جهت جریان

ρ: چگالی هوا

همچنین یک منبع حرارتی (Source Term) نیز در ناحیه ورودی برج در نظر گرفته شده است.

۴-۴-شرایط مرزی

شرایط مرزی طبق شکل (۴-۲) که نمایشگر دامنه استفاده شده در این تحقیق می باشد عبارتند از:

۴-۴-۱- شرط مرزی سرعت ورودی (velocity inlet)

برای بیان سرعت و سایر خواص جریان در مرز ورودی استفاده شده است . مولفه سرعت در جهت جریان برابر با سرعت باد در نظر گرفته شده است :

$$u = u_w$$

و دیگر مولفه های سرعت یعنی w,v برابر صفر در نظر گرفته شده اند .

دمای هوا برابر درجه حرارت طراحی یعنی 285k منظور شده است . برای تعیین پارامترهای اغتشاش از مدل شدت اغتشاش Turbulence Intensity و نسبت ویسکوزیته اغتشاش (Turbulence viscosity Ratio) استفاده شده است که شدت توربولانس از رابطه زیر قابل محاسبه است ؛

$$I = \frac{u'}{u_{ave}} = 0.16(\mathrm{Re}_{Dh})^{-\frac{1}{8}} \qquad (۱۸-۴)$$

نسبت وزیسکوزیته اغتشاش $(\frac{\mu}{\mu_t})$ به طور مستقیم با عدد رینولدز اغتشاش $(\mathrm{Re}_t = \frac{k^2}{\varepsilon \nu})$ متناسب است و به صورت نمونه پارامترهای اغتشاش در حوزه مقادیر $10 \prec \frac{\mu}{\mu_t} \prec 1$ تنظیم می شود .

۴-۴-۲- شرط مرزی فشار خروجی (Pressure outlet)

این شرط مرزی برای تعریف فشار استاتیکی هوا در خروجی جریان بکار می رود .در این نوع شرط مرزی فشار نسبی خروجی در مرز به عنوان ورودی به نرم افزار داده می شود . همچنین دمای هوا و پارامترهای اغتشاش که قبلاً توضیح داده شد .

۴-۴-۳-شرط مرزی دیوار(wall)

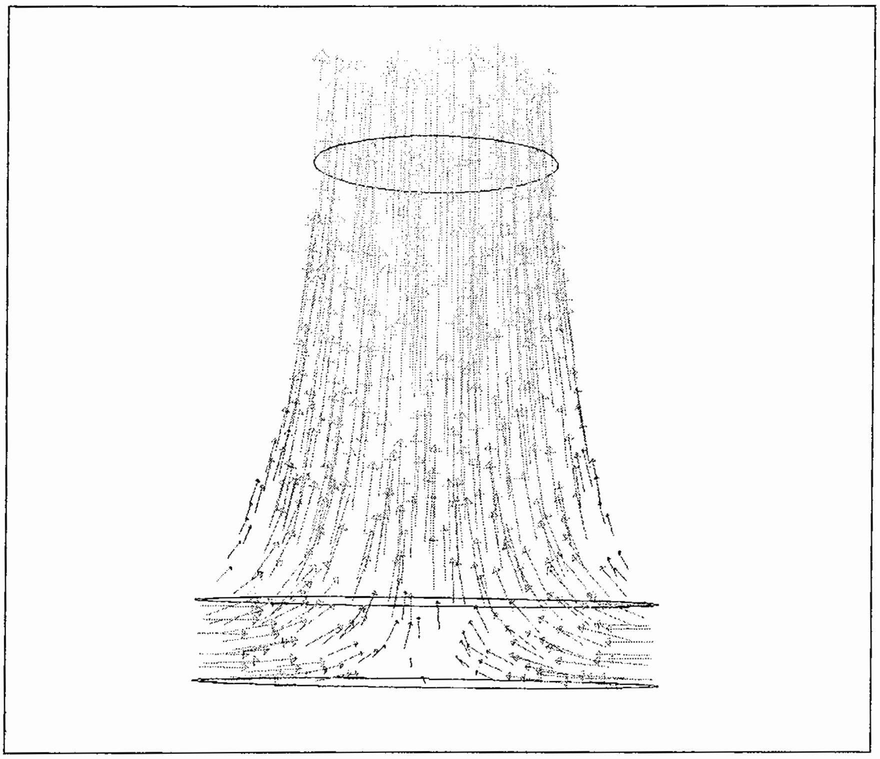
این شرط مرزی یعنی شرط عدم لغزش در دیواره ها برای بدنه برج خنک کن و دیوارهای بادشکن (wind break walls) استفاده شده است .

۴-۵-ارائه نتایج

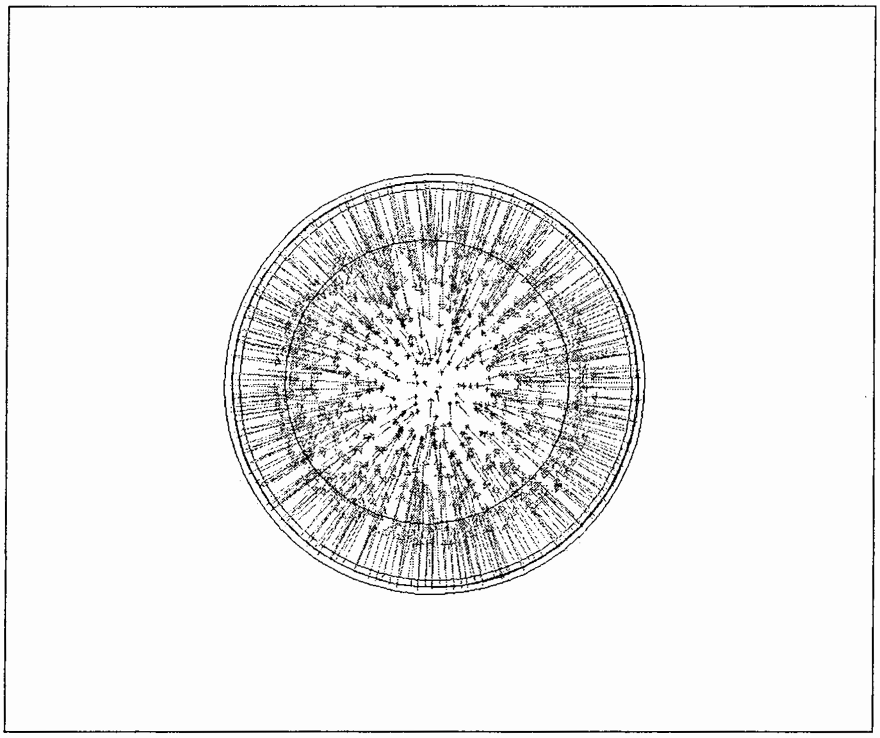
با توجه به مطالبی که تاکنون ذکر شده ، شبیه سازی برای سرعتهای مختلف باد انجام شده و نتایج حاصل از آن به صورت زیر ارائه خواهند شد .

ابتدا نتایج حاصل از حل برای سرعت باد صفر یعنی حالت جابجایی طبیعی ارائه می شوند. بردارهای سرعت در صفحه تقارن و صفحه افقی در ارتفاع ۱۰ متری در شکلهای (۴-۳) و (۴-۴) نشان داده شده اند. ملاحظه می شود که جریان کاملا متقارن محوری است.

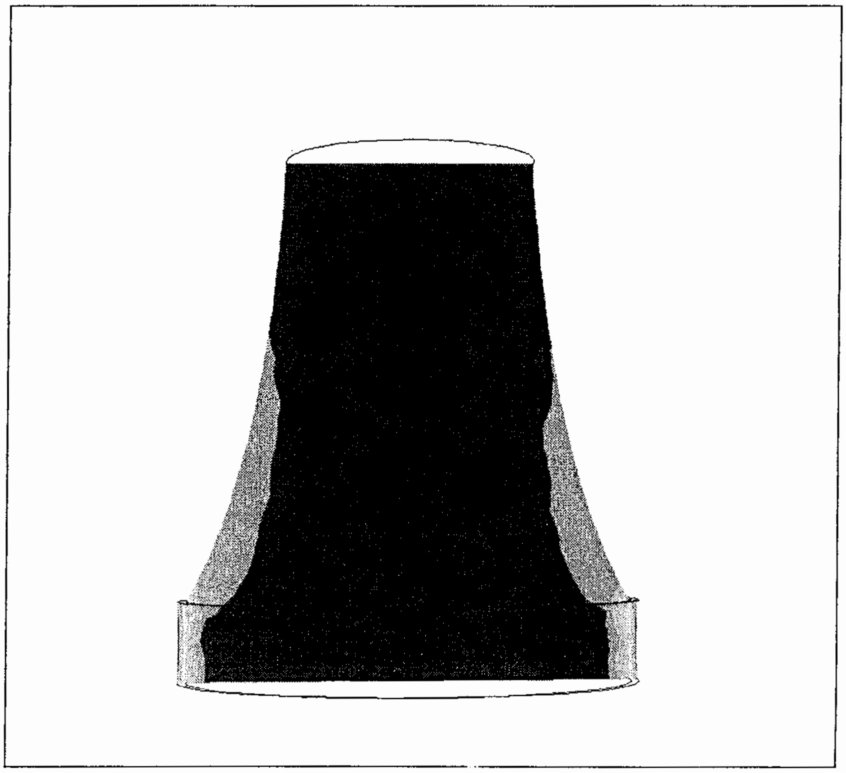
کانتورهای دما در صفحه تقارن نیز در شکل(۴-۵) ارائه شده است و همانطور که در فصل قبل اشاره شد، تغییرات دما فقط در ناحیه ورودی برج اتفاق می افتد.



شکل(۴-۳): بردارهای سرعت در صفحه تقارن برای حالت جابجایی طبیعی



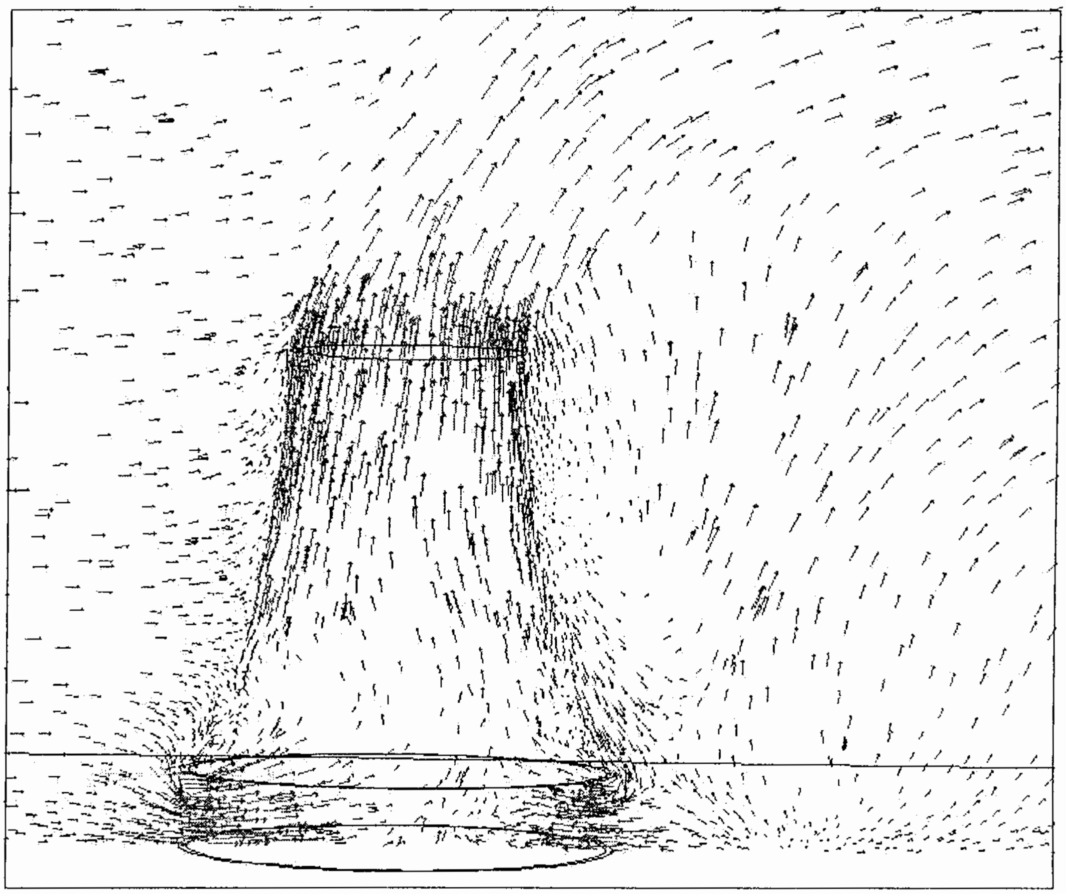
شکل(۴-۴): بردارهای سرعت در صفحه افقی برای حالت جابجایی طبیعی



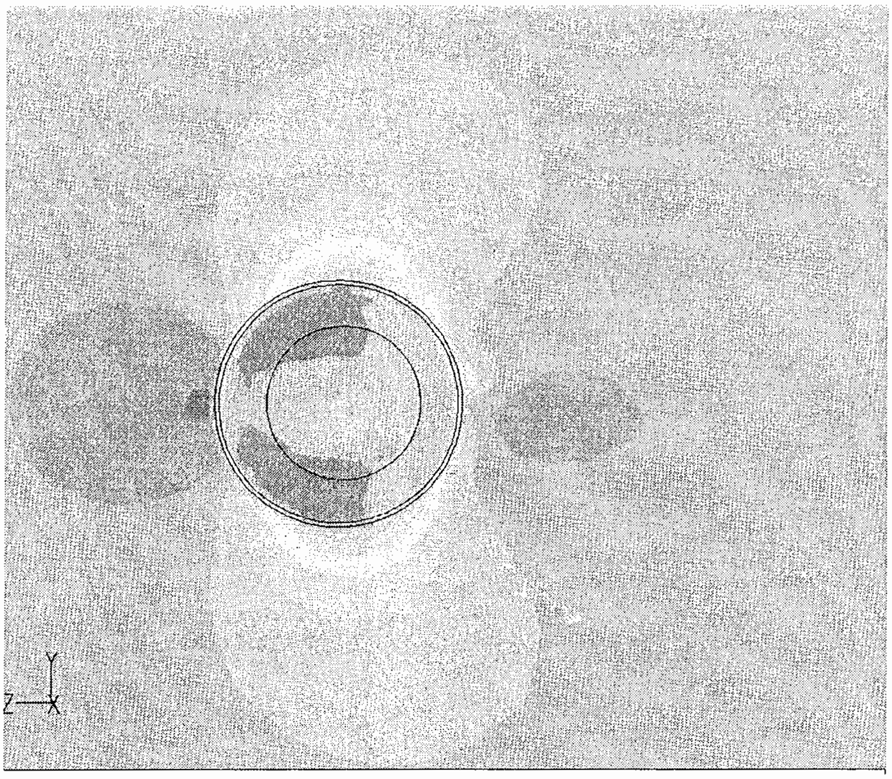
شکل(۴-۵): کانتورهای دما در صفحه تقارن برای حالت جابجایی طبیعی

بردارهای سرعت را در صفحه تقارن و صفحه افقی در ارتفاع ۱۰ متری برای سرعت باد 5 m/s در شکلهای (۴-۶) و (۴-۷) نشان داده شده است. کانتورهای دما در صفحه تقارن نیز در شکل (۴-۸) ملاحظه می شوند .

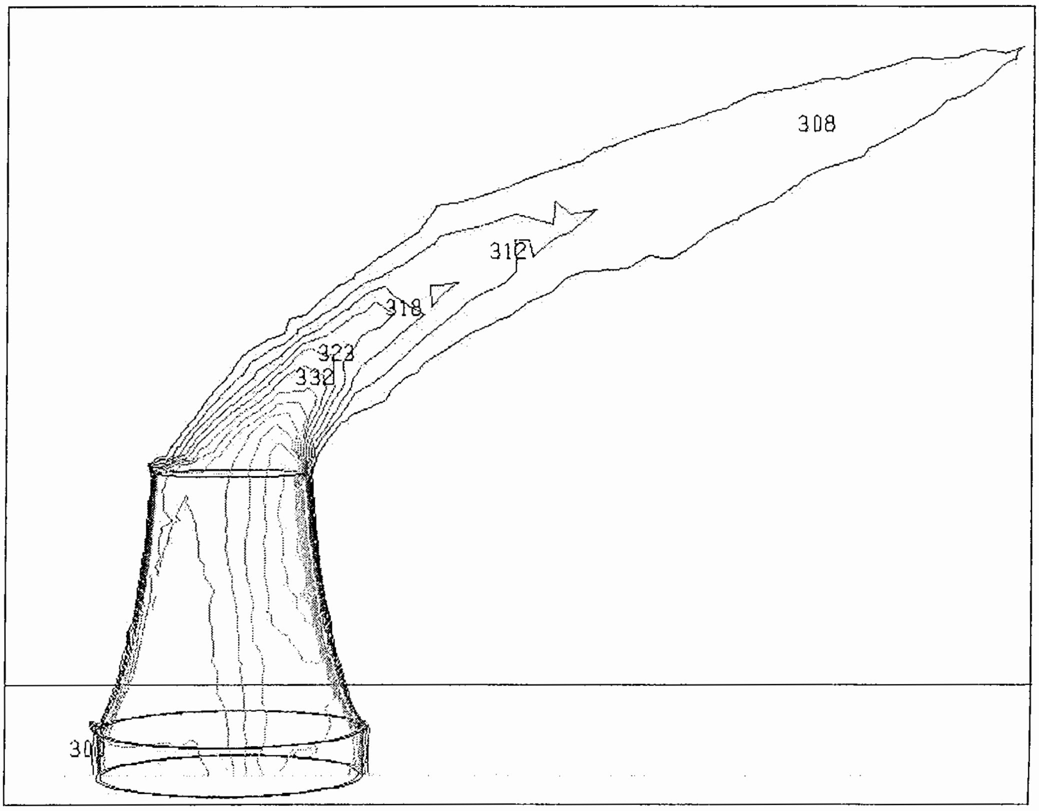
با توجه به شکلهای ارائه شده دبی جرمی ورودی در قسمتهای کناری و پشت برج کاهش در قسمتی از برج که مقابل جهت وزش باد قرار دارد افزایش می یابد . همچنین باد یک شکل در پوش مانند در بالای برج ایجاد می کند. این پدیده (Wind Cover or Cap Effect) بدلیل تفاوت اندازه حرکت و جهت سیال خروجی از برج و سیالی است که به طور موازی با افق حرکت می کند می باشد ، این پدیده نیز باعث کاهش مکش برج و نهایتاً کاهش دبی هوای خنک کن عبوری از برج می شود .



شکل (۴-۶) : بردارهای سرعت در صفحه تقارن برای سرعت 5m/s



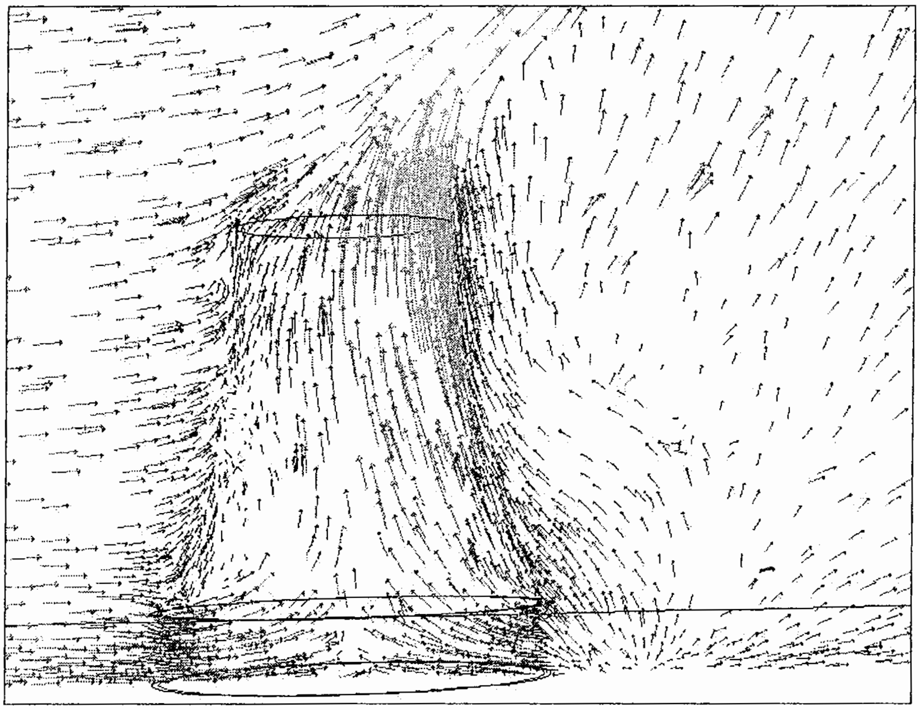
شکل (۴-۱۰): کانتورهای فشار در صفحه افقی برای سرعت باد ۱۰m/s



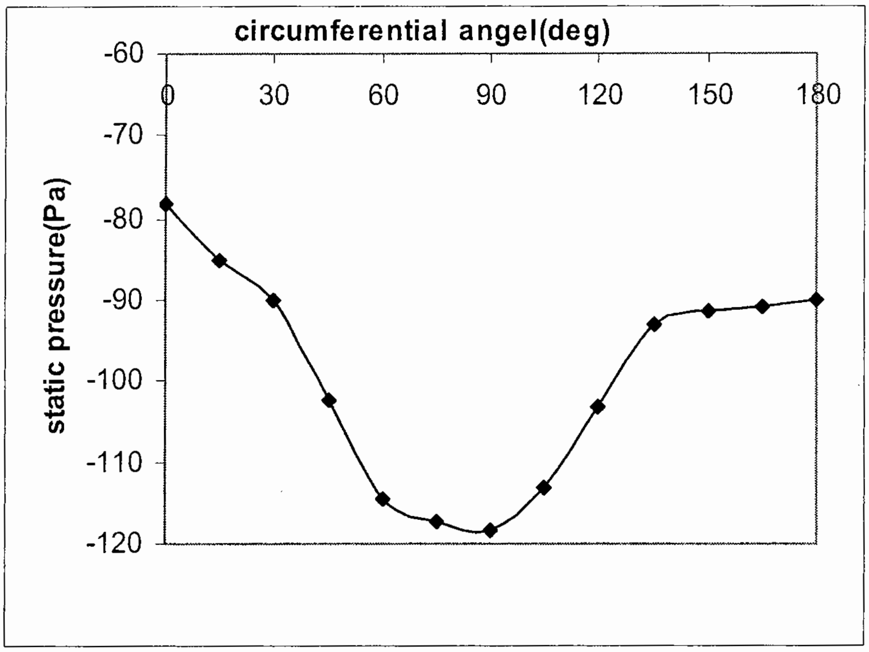
شکل (۴-۱۱): کانتورهای دما در صفحه تقارن (u=10 m/s)

بردارهای سرعت در صفحه تقارن برای سرعت باد ۱۰ m/s در شکل (۴-۱۲) نشان داده شده اند .با توجه به این شکل واضح است که پدیده wind – cover در بالای برج ایجاد شده است . تغییرات فشار در سرعت باد ۱۰m/s در شکل (۴-۱۳) نشان داده شده است که با توجه به شکل (۱-۴) در فصل یک که مقادیر اندازه گیری شده ضریب فشار را نشان می دهد رفتار مشابهی دارد .

با افزایش سرعت باد ، سرعت هوای ورودی در قسمت جلویی افزایش یافته و هوای وارد شده از قسمت پشت برج را به عقب رانده و بنابراین به مقدار زیادی از دبی هوای خنک کن کاسته شده و عملکرد برج دچار افت شدید می شود . همچنین با توجه به این مطالب با افزایش سرعت باد پدیده wind cover و همچنین گردابه ها نیز تشدید می شوند .



شکل(۴-۱۲): بردارهای سرعت در صفحه تقارن در سرعت باد 10 m/s



شکل (۴-۱۳) : تغییرات فشار ورودی برج

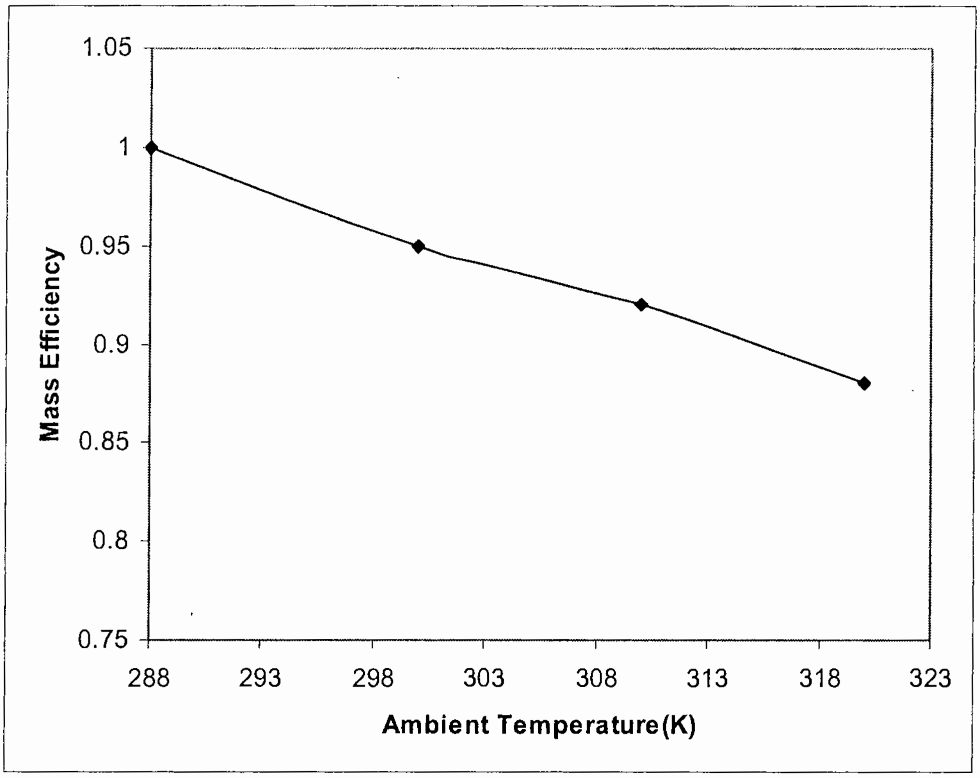
۴-۵-۱- راندمان جرمی

برای مقایسه عملکرد برج در حالتهای مختلف ، میزان دبی جریان هوایی که از داخل رادیاتورها و در نتیجه از داخل برج عبور می کند می تواند به عنوان شاخصی مناسب از انتقال حرارت و در نتیجه عملکرد برج باشد . با معرفی یک ضریب تأثیر η که برابر است با نسبت دبی جرمی هوای عبوری ازداخل برج در حالتهای مختلف به دبی جرمی هوای عبوری از برج تحت شرایط جابجایی طبیعی (بدون وزش باد) ودمای محیط 285k ، حالتهای مختلف مورد بررسی قرار می‌گیرند .

$$\eta = \frac{\dot{m}}{\dot{m}_{nominal}}$$

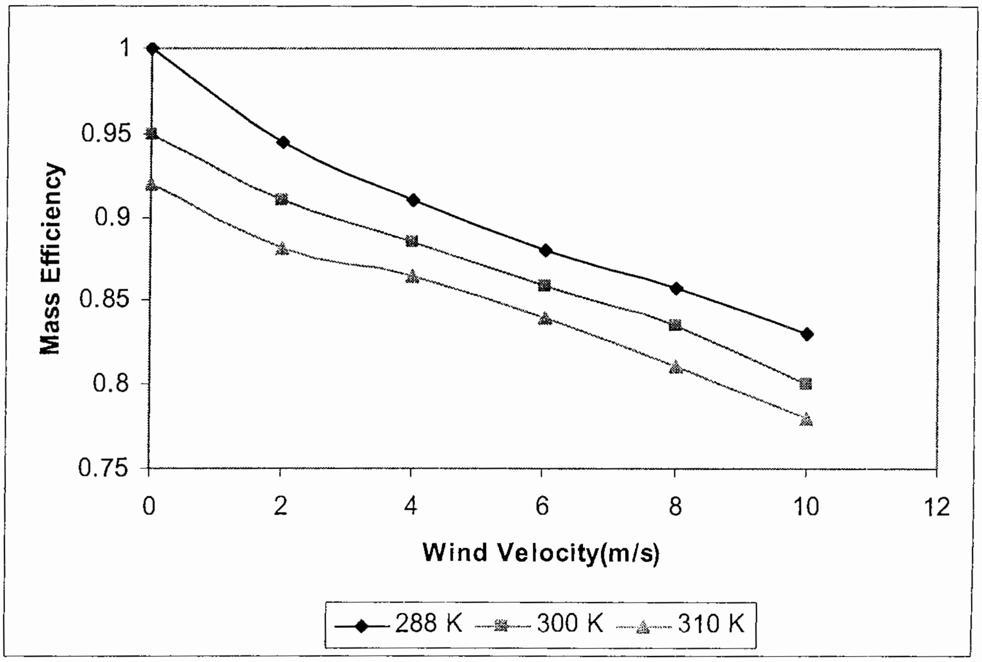
شکل (۴-۱۴) تغییرات راندمان جرمی را برای سرعتهای مختلف باد نشان می دهد . ملاحظه می شود که با افزایش سرعت باد راندمان جرمی نهایتاً عملکرد برج خنک کن دچار افت می شود با توجه به نتایجی که تاکنون ذکر شد ، افت عملکرد برج ناشی از عوامل زیر می تواند باشد:

الف ـ ایجاد ورتکس در پایین برج بدلیل تفاوت سرعت های جریان ورودی از سمت مقابل باد و پشت آن.



شکل (۴-۱۵) : تغییرات راندمان جرمی در دماهای محیط مختلف

اثر دمای محیط در سرعتهای مختلف باد نیز اندازه گیری شـده کـه مقـادیر رانـدمان جرمـی در سرعتهای مختلف و سه دمای محیط در شکل (۴-۱۶) نشان داده شده است .



شکل (۴-۱۶) : تغییرات راندمان جرمی در سرعتهای مختلف باد و دماهای مختلف محیط

۵ـ۱ـ ارائه راه حل

در فعل گذشته عوامل افت عملکرد برجهای خنک کن خشک تحت شرایط باد متقاطع بیان شد . جهت بهبود عملکرد برجهای خنک کن تحت شرایط ذکر شده ، ابتدا kroeger [14] دیوارهای بادشکن را برای برجهای خنک کن Homon-Type معرفی کرد .

در برجهای Homon-Type ، مبدلهای حرارتی یا به عبارتی رادیاتورها به صورت افقی در سطح مقطع برج قرار گرفته اند در صورتی که در برجهای Heller-Type که موضوع مورد بحث در این تحقیق می باشند ، مبدلهای حرارتی به صورت عمودی در ورودی برج نصب شده‌اند .

تحقیقات kroeger نشان داد که برای برجهای مذکور ، استفاده از دیوار باد شکن به شکل یک دیوار متخلخل (porous) که در وسط برج و به صورت عمود بر جریان باد قرار می گیرد ، نقش مهمی در بهبود عملکرد برجهای Homon-Type تحت شرایط باد متقاطع دارد .

همزمان با تحقیق حاضر و هنگامی که نتایج ، نقش ایجاد تغییر در شکل خارجی برج و ایجاد دیوارهایی جهت هدایت باد به داخل برجهای خنک کن را در بهبود راندمان این برجها با اثر باد ، مثبت نشان می داد، مقاله ای توسط AłW aked & Behnia [16] منتشر شد که در این مقاله دیوارهای بادشکن به شکل دیوارهای شعاعی خارجی در برجهای Homon Type به عنوان راه حل مشکل مورد بحث ارائه شده است . در بخش بعدی دیوارهای بادشکن مورد استفاده و نتایج حاصله از آنها معرفی می شوند .

۵-۲- دیوارهای بادشکن در پایین برج

تاکنون اثرات مخرب باد بر کارایی برجهای خنک کن و عوامل آنها مورد بحث قرار گرفت و راههایی که تاکنون برای حل مشکل ارائه شده است معرفی شدند . پیشنهادی که برای کاهش اثرات باد ارائه شده است ، استفاده از خود باد به عنوان عاملی جهت کاهش اثرات مذکور می‌باشد . بدین ترتیب که توسط وسایلی باد متقاطع به صورت یکنواخت در ورودی برج تقسیم شود به نحوی که سرعت ورودی شعاعی در همه نواحی سه گانه ذکر شده (Front , side & back part) برابر سرعت باد شود که این حالت یک حالت ایده ال می باشد که عملاً رسیدن به چنین شرایطی غیر ممکن به نظر می رسد .

برای نزدیک شدن به این هدف نصب دیوارهایی در مکانهایی که باعث افت عملکرد برج می شوند یعنی ناحیه side part و به مقدار کمی back part و هدایت باد به این نواحی که در نهایت باعث افزایش دبی جرمی هوای خشک کن عبوری از روی مبدلهای حرارتی می شود ضروری به نظر می رسد . به طور کلی چهار نوع دیوار باد شکن در این تحقیق مورد بررسی قرار می گیرند .

۱- دو دیوار منحنی شکل در زاویه ۱۲۰ درجه.

۲- چهار دیوار منحنی شکل در زوایای ۱۲۰ و ۱۸۰ درجه.

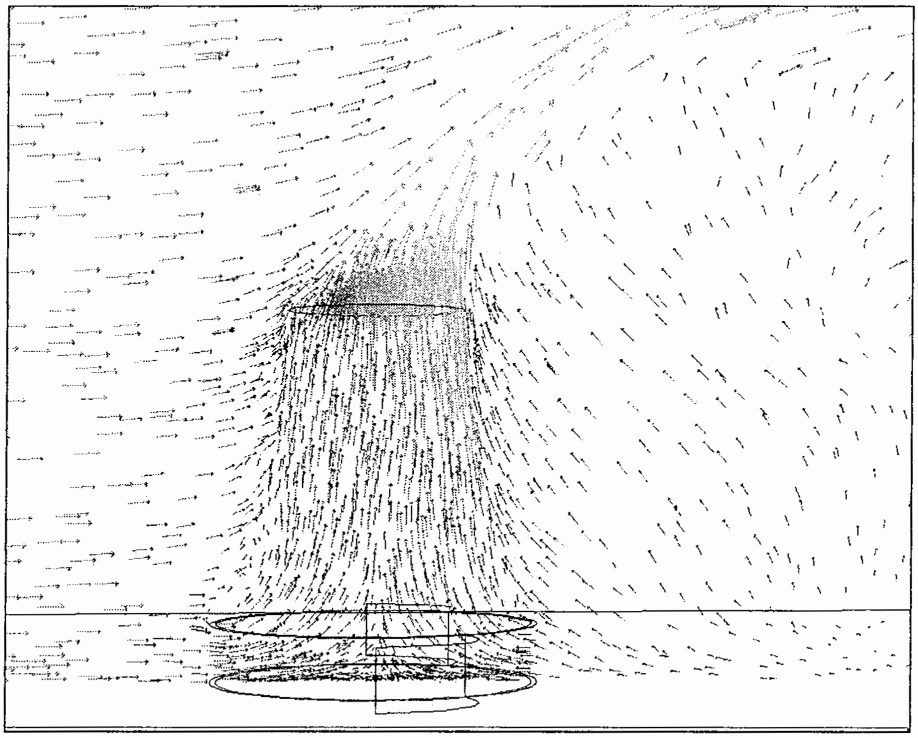
۳- دو دیوار شعاعی در زاویه ۹۰ درجه.

۴- استفاده از هشت دیوار شعاعی.

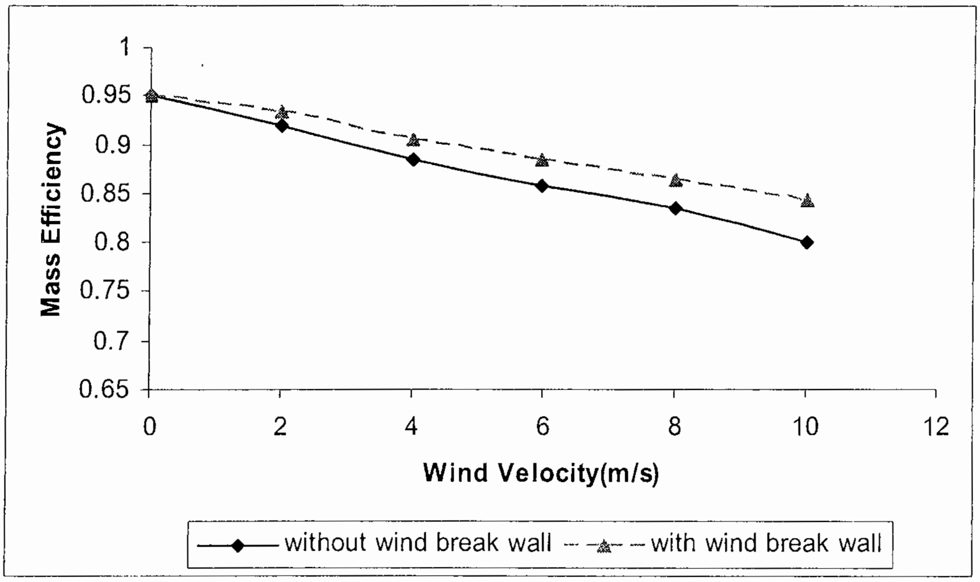
۵-۲-۱- دو دیوار منحنی شکل در زاویه ۱۲۰ درجه

اولین طرح پیشنهادی جهت نصب دیوارهای بادشکن ، استفاده از دیوار منحنی شکل به نحوی در شکل (۵-۱) ملاحظه می شود در زاویه ۱۲۰ درجه بود. نصب این دیوارها دبی جرمی خنک کن را افزایش داده و باعث افزایش راندمان برج می شود .

بردارهای سرعت در صفحه تقارن و نیز تغییرات راندمان جرمی در شکلهای (۵-۳)و (۵-۴) مشاهده می شوند . ضمناً باید با توجه به نمودار راندمان جرمی مشخص می شود که نصب این دیوارها برای حالت سرعت صفر باد یعنی شرایط جابجایی طبیعی ،اثر منفی بر عملکرد ندارد .



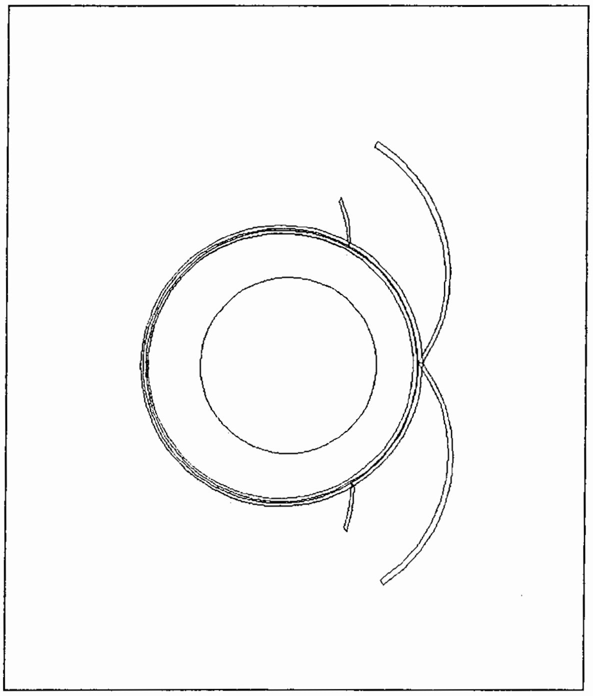
شکل(۵-۳): بردارهای سرعت در صفحه تقارن برای دو دیوار منحنی شکل



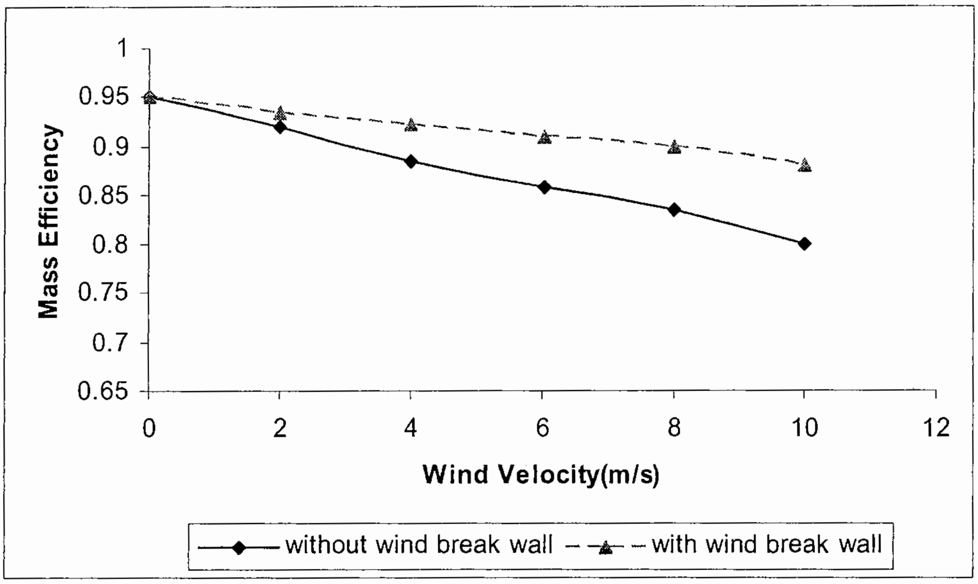
شکل(۵-۴):تغییرات راندمان جرمی برای دو دیوار منحنی شکل

۵-۲-۲- چهار دیوار منحنی شکل در زوایای ۱۲۰ و ۱۸۰ درجه

دومین پیشنهاد برای نصب دیوارهای بادشکن ، استفاده از دیوارهای کـوچکتر از حالـت قبـل در زاویه ۱۲۰ درجه و افزودن دو دیوار دیگر به نحو نشان داده شده در شکل (۵-۵) در زاویه ۱۸۰ درجه می باشد . در این حالت نیز شاهد افزایش دبی جرمی ، راندمان جرمی و نهایتاً راندمان خنک کاری به میزانی بیشتر از حالت قبل هستیم بدلیل اینکه ناحیه پشت برج نیـز تقویـت شـده اسـت . بردارهـای سرعت در صفحه افقی در ارتفاع ۱۰ متر ، بردارهای سرعت در صفحه تقارن و تغییرات راندمان جرمی در سرعتهای مختلف باد به ترتیب در شکل های (۵-۶)و (۵-۷)و (۵-۸)مشـاهده مـی شـود . در ایـن حالت نیز همانطور که مشاهده می شود ، گردابه ها و پدیده درپوشی نیز ضعیف تر شده اند .



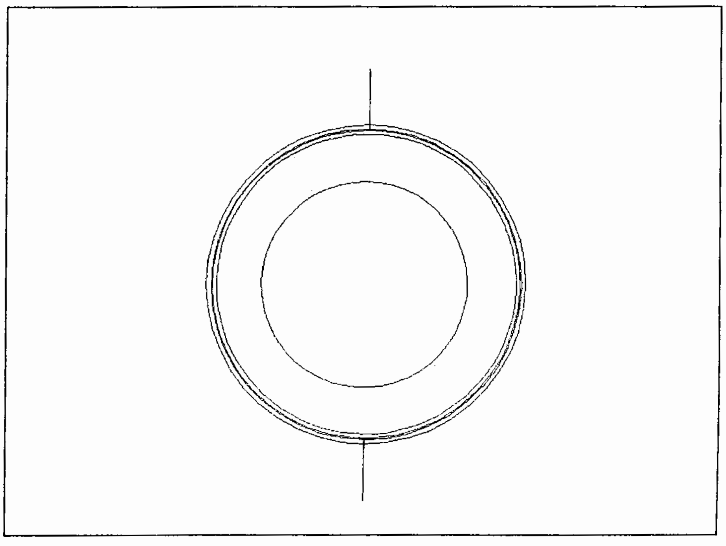
شکل(۵-۵): استفاده از چهار دیوار منحنی شکل



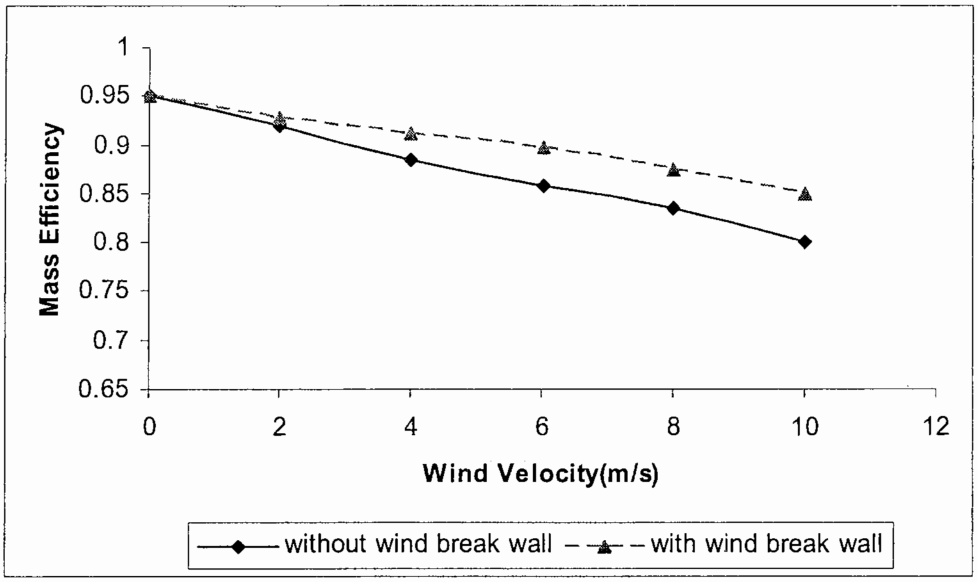
شکل(۵-۸): تغییرات راندمان جرمی برای چهار دیوار منحنی شکل

۵ـ۲ـ۳ـ دو دیوار شعاعی در زاویه ۹۰ درجه

نوع سوم از دیوارهای بادشکن ، نصب دو دیوار شعاعی مانند آنچه در شکل (۵-۹) دیده می‌شود در زاویه ۹۰ درجه می باشد . استفاده از این دیوارها نیز سبب بهبود عملکرد برجهای خنک کن می شود . بردارهای سرعت در صفحه افقی در ارتفاع ۱۰ متری ، بردارهای سرعت در صفحه تقارن و تغییرات ارندمان جرمی در سرعتهای مختلف در شکلهای (۵-۱۰) و (۵-۱۱) و (۵-۱۲) مشاهده می شوند .



شکل(۵-۹): استفاده از دو دیوار شعاعی

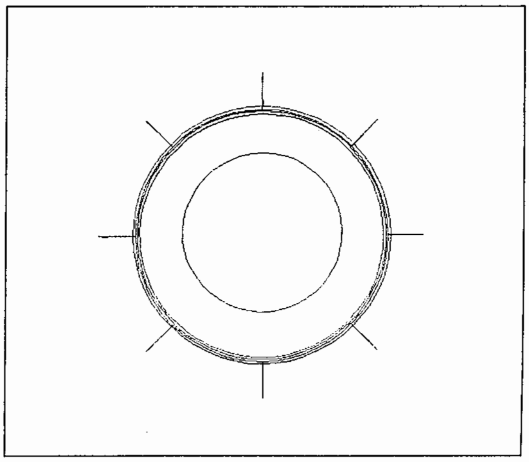


شکل(۵-۱۲): تغییرات راندمان جرمی برای دو دیوار شعاعی

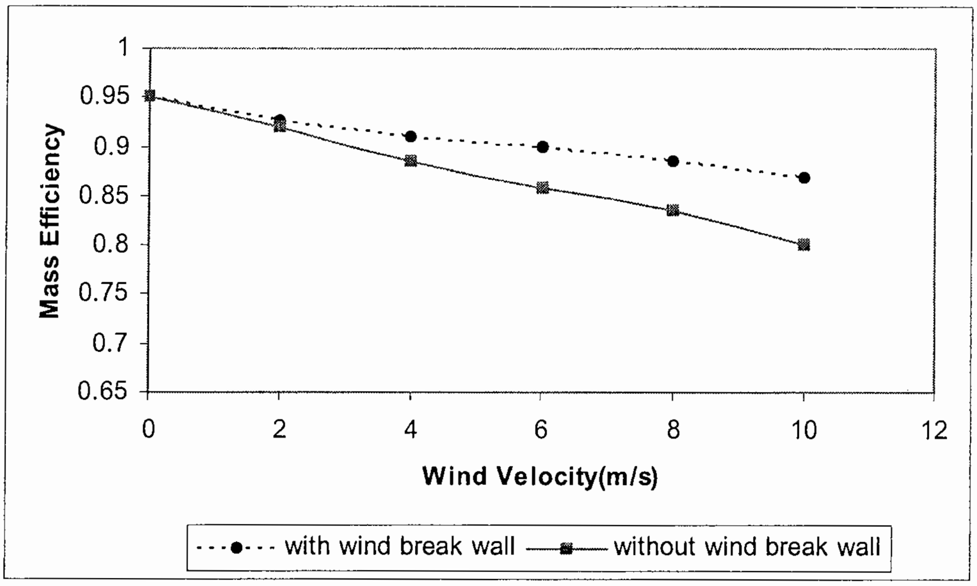
۵ـ۲ـ۴ـ استفاده از هشت دیوار شعاعی

آخرین نوع دیوارهای مورد استفاده ، استفاده از ۸ دیوار به ظور شعاعی و بـه فواصـل مسـاوی در ورودی و در قسمت خارجی برج می باشد(شکل (۵-۱۳)) . بردارهای سرعت در صفحه افقی در ارتفاع ۱۰ متری و بردارهای سرعت در صفحه تقارن در شکل های (۵-۱۴) و (۵-۱۵) ملاحظه می شوند .

با توجه به شکلهای اشاره شده ، گردابه های ایجـاد شـده در پـایین بـرج در هنگـام وزش بـاد و همچنین اثر درپوشی (Wind Cover) به مقدار قابل ملاحظه ای تضعیف شده اند که این امر باعث افزایش دبی جرمی هوا می شود .



شکل(۵-۱۳): استفاده از ۸ دیوار شعاعی

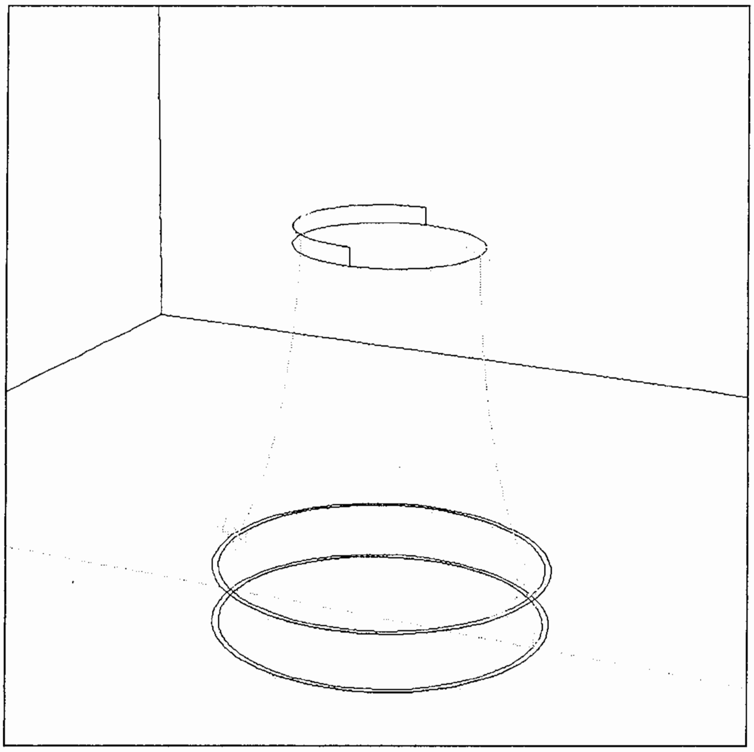


شکل(۵-۱۶): تغییرات راندمان جرمی برای ۸ دیوار شعاعی

بنابراین تغییرات راندمان جرمی در سرعتهای مختلف بادشکن (۵-۱۶) ارائه شده اند . با توجه به انواع دیوارهای بادشکن ذکر شده ، هر چند که جهت باد غالب در هر منطقه مشخص باشد اما بـدلیل اینکه این مدل به جهت باد وابستگی ندارد برای مناطقی که جهت باد در آنها متغیر است بیشتر مفید می باشد .

۵ـ۳ـ استفاده از دیوار بادشکن در بالای برج

تاکنون کارهایی که جهت افزایش راندمان برجهای خنک کن خشک تحت اثر باد معرفی شدند ، مربوط به پایین برج و از بین بردن گردابه های تشکیل شده در آنجا بود. اما تاکنون راه حلی برای حل مشکل پدیده درپوشی ارائه نشده است . تحقیقات اولیه انجام شده نشان داد کـه ایجـاد تغییراتـی در شکل خارجی برج در بالای آن نیز می تواند باعث کاهش اثر پدیده درپوشی شـود . لـذا بـا توجـه بـه شکل (۵-۱۷) یک نوع دیوار بادشکن در بالای برج در نظر گرفته شده است .



شکل(۵-۱۷): استفاده از دیوار بادشکن در بالای برج

این دیوار در دو ارتفاع ۲ و۵ متر به صورت نیم دایره در مقابل جهت باد در بالای برج منظور شده است . بردارهای سرعت در صفحه افقی و صفحه تقارن به ترتیب در شکلهای (۵-۱۸) و (۵-۱۹) برای دیوار با ارتفاع ۵ متر مشاهده می شوند . با توجه به این شکلها ملاحظه می شود که علاوه بـر اینکـه پدیده درپوشی (wind cover) بهبود پیدا کرده است ، گردابه های موجود در پایین برج نیز از بـین رفته اند .

علاوه بر دیوارهای به ارتفاع ۵ متر ، دیوارهایی با ارتفاع ۲ متر نیز در نظـر گرفتـه شـده انـد کـه بردارهای سرعت در صفحه تقارن برای این مورد نیز در شکل (۵-۲۰) ارائه شده است.

با توجه به نتایجی که تاکنون ذکر شد ،مشخص مـی شـود کـه پدیـده درپوشـی و گردابـه هـای تشکیل شده در پایین برج در پدیده مستقل از هم نیستند و به نحوی که اشاره شد راه حل هایی کـه برای از بین بردن هر یک پیشنهاد شد باعث تضعیف پدیده دیگر نیز می شوند . لذا اسـتفاده از دیـوار بادشکن در بالای برج با از بین بردن پدیده درپوشی ، مکش در برج را افزایش داده و بـه ایـن ترتیـب باعث افزایش دبی جرمی هوای خنک کن و راندمان جرمی شده و گردابه ها را نیز از بین می برد .

مراجع و مآخذ

[1] Charles Hirsch , Numerical Computation of Internal and External Flows , Vol. 2 , JOHN WILY & SONS , 1990.

[2] V. Venkatakrishnan , A Perspective on Unstructured Grid Flow Solver , ICASE Reports , 1991.

[3] Hoffmann, K.A. and Chiang, S.T., 1989, "Computational Fluid Dynamics for Engineers," First Edition, two volumes, Austin, Texas: EES.

[4] Neal T. Frink , Paresh Parikh , Shahyar Pirzadeh , A Fast Upwind Solver for the Euler Equations on Three – Dimensional Unstructured Meshes , AIAA Paper 91-0102 , 1991.

[5] John C. Tennhill , Daol A. Anderson , Richard H. Pletcher , Computational Fluids Mechanics and Heat Transfer , Taylor & Francis , 2nd Edition , 1997.

[6] Josep L. Steger and R. F. Warming , Flux Vector Splitting of the Inviscid Gasdynamic Equations with Application to Finite Difference Methods, J. of Comp. Phy. , 1981

[7] Brame Van Leer , Flux Vector Splitting for the Euler Equations , ICASE Reports,

[8] A Comparison of Finite Volume Flux Vector Splitting for the Euler Equations , AIAA Paper 85-0122 , 1985.

[9] J.C.Mandal and S.M.Deshpande, " Kinetic Flux Vector Splitting For Euler Equations", Computers Fluids, Vol.23, No.2, 1994.

[10] Neal T. Frink , Paresh Parikh , Shahyar Pirzadeh , A Fast Upwind Solver for the Euler Equations on Three – Dimensional Unstructured Meshes , AIAA Paper 91-0102 , 1991

[11] D. Bergetrom, D. Derkson and K.Rezkallah, Numerical Study of Wind Flow Over a Cooling Tower, Journal of Wind Engineering and Industrial Aerodynamics, Vol. 46-47, pp. 657-664, (1993).

[12] D. Demoren and W. Rodi, Three Dimensional Numerical Calculations of Flow and Plume Spreading Past Cooling Towers, Journal of Heat Transfer, Vol. 109, pp. 113-119, (1987).

[13] FLUENT, User's Guide, FLUENT Incorporated, Lebanon, NH, (1999).

[14] A.F. Du Preez and D. Kroger, the Effect of The Heat Exchanger Arrangement and Wind Break Walls on The Performance of Natural Draft Dry-Cooling Towers Subjected to Cross-Winds, Journal of Wind Engineering and Industrial aerodynamics, Vol. 58, pp. 293-303, (1995).

[15] M. Su, G. Tang and S. Fu, Numerical simulation of fluid Flow and Thermal Performance of a Dry-Cooling Tower Under Cross Wind Condition, Journal of Wind Engineering and Industrial Aerodynamics, Vol. 79, No. 3, pp. 289-306, (1999).

[16] R. Al-Waked and M. Behnia, The Performance of Natural Draft Dry Cooling Towers under Cross wind: CFD Study, International Journal of energy Research, Vol. 28, pp. 147-161, (2004).

[17] D.G. Kroger, Air-Cooled Heat Exchanger and Cooling Towers, Thermal Flow Performance Evaluation and Design, Begell House, Inc, New York, (1998).

[18] Q.Wei, B.Zhang, K.Liu, X.Du and X.Meng, A Study of the Unfavorable Effects of Wind on the Cooling Efficiency of dry cooling towers, Journal of Wind Engineering and Industrial aerodynamics, Vol.54/55, pp.633-643, (1995).

ضمائم

الف- گزارش حل FLUENT

ب- برنامه کامپیوتری

ج- مقالات ارائه شده در کنفرانسهای داخلی و خارجی

ضمیمه الف- گزارش حل نرم افزار FLUENT

```
FLUENT
Version: 3d, segregated, ske (3d, segregated, standard k-epsilon)
Release: 6.0.12
Title:

Models
------

   Model                         Settings
   ----------------------------------------------------------------
   Space                         3D
   Time                          Steady
   Viscous                       Standard k-epsilon turbulence model
   Wall Treatment                Standard Wall Functions
   Heat Transfer                 Enabled
   Solidification and Melting    Disabled
   Radiation                     None
   Species Transport             Disabled
   Coupled Dispersed Phase       Disabled
   Pollutants                    Disabled
   Soot                          Disabled

Boundary Conditions
-------------------

   Zones

      name                       id   type
      -------------------------------------------
      fluidt                     2    fluid
      fluidq                     3    fluid
      fluida                     4    fluid
      inlet1                     14   interior
      wall2-shadow               22   wall
      wsd-shadow                 21   wall
      inti                       5    interior
      wsd                        6    wall
      intu                       7    wall
      inlet2                     8    velocity-inlet
      out2                       9    pressure-outlet
      right                      10   wall
      bot                        11   wall
      left                       12   wall
      top                        13   wall
      wall2                      15   wall
      out1                       16   interior
      wall1                      17   wall
      default-interior           19   interior
      default-interior:001       1    interior
      default-interior:018       18   interior

   Boundary Conditions
```

fluidt

Condition	Value
Material Name	air
Specify source terms?	no
Source Terms	()
Specify fixed values?	no
Local Coordinate System for Fixed Velocities	no
Fixed Values	()
Motion Type	0
X-Velocity Of Zone	0
Y-Velocity Of Zone	0
Z-Velocity Of Zone	0
Rotation speed	0
X-Origin of Rotation-Axis	0
Y-Origin of Rotation-Axis	0
Z-Origin of Rotation-Axis	0
X-Component of Rotation-Axis	0
Y-Component of Rotation-Axis	0
Z-Component of Rotation-Axis	1
Laminar zone?	no
Porous zone?	no
Conical porous zone?	no
X-Component of Direction-1 Vector	1
Y-Component of Direction-1 Vector	1
Z-Component of Direction-1 Vector	1
X-Component of Direction-2 Vector	0
Y-Component of Direction-2 Vector	1
Z-Component of Direction-2 Vector	0
X-Coordinate of Point on Cone Axis	1
Y-Coordinate of Point on Cone Axis	0
Z-Coordinate of Point on Cone Axis	0
Half Angle of Cone Relative to its Axis	0
Direction-1 Viscous Resistance	0
Direction-2 Viscous Resistance	0
Direction-3 Viscous Resistance	0
Direction-1 Inertial Resistance	0
Direction-2 Inertial Resistance	0
Direction-3 Inertial Resistance	0
C0 Coefficient for Power-Law	0
C1 Coefficient for Power-Law	0
Porosity	1
Solid Material Name	aluminum

fluidq

Condition	Value


```
----------------------------------------------------------------------------
-----------------------------------------------------
         Material Name                                         air
         Specify source terms?                                 yes
         Source Terms                                          ((mass (inactive
. #f) (constant . 0) (profile  )) (x-momentum (inactive . #f) (constant .
0) (profile  )) (y-momentum (inactive . #f) (constant . 0) (profile  ))
(z-momentum (inactive . #f) (constant . 0) (profile  )) (k (inactive .
#f) (constant . 0) (profile  )) (epsilon (inactive . #f) (constant . 0)
(profile  )) (energy (constant . 153778) (profile  )))
         Specify fixed values?                                 no
         Local Coordinate System for Fixed Velocities          no
         Fixed Values                                          ((x-velocity
(inactive . #f) (constant . 0) (profile  )) (y-velocity (inactive . #f)
(constant . 0) (profile  )) (z-velocity (inactive . #f) (constant . 0)
(profile  )) (k (inactive . #f) (constant . 0) (profile  )) (epsilon
(inactive . #f) (constant . 0) (profile  )) (temperature (inactive . #f)
(constant . 0) (profile  )))
         Motion Type                                           0
         X-Velocity Of Zone                                    0
         Y-Velocity Of Zone                                    0
         Z-Velocity Of Zone                                    0
         Rotation speed                                        0
         X-Origin of Rotation-Axis                             0
         Y-Origin of Rotation-Axis                             0
         Z-Origin of Rotation-Axis                             0
         X-Component of Rotation-Axis                          0
         Y-Component of Rotation-Axis                          0
         Z-Component of Rotation-Axis                          1
         Laminar zone?                                         no
         Porous zone?                                          yes
         Conical porous zone?                                  yes
         X-Component of Direction-1 Vector                     1
         Y-Component of Direction-1 Vector                     0
         Z-Component of Direction-1 Vector                     0
         X-Component of Direction-2 Vector                     0
         Y-Component of Direction-2 Vector                     1
         Z-Component of Direction-2 Vector                     0
         X-Coordinate of Point on Cone Axis                    10
         Y-Coordinate of Point on Cone Axis                    0
         Z-Coordinate of Point on Cone Axis                    0
         Half Angle of Cone Relative to its Axis               0
         Direction-1 Viscous Resistance                        0
         Direction-2 Viscous Resistance                        0
         Direction-3 Viscous Resistance                        0
         Direction-1 Inertial Resistance                       0
         Direction-2 Inertial Resistance                       0
         Direction-3 Inertial Resistance                       0
         C0 Coefficient for Power-Law                          92.860001
         C1 Coefficient for Power-Law                          1.76
         Porosity                                              0.69999999
         Solid Material Name                                   aluminum

      fluida
```

Condition	Value
Material Name	air
Specify source terms?	no
Source Terms	()
Specify fixed values?	no
Local Coordinate System for Fixed Velocities	no
Fixed Values	()
Motion Type	0
X-Velocity Of Zone	0
Y-Velocity Of Zone	0
Z-Velocity Of Zone	0
Rotation speed	0
X-Origin of Rotation-Axis	0
Y-Origin of Rotation-Axis	0
Z-Origin of Rotation-Axis	0
X-Component of Rotation-Axis	0
Y-Component of Rotation-Axis	0
Z-Component of Rotation-Axis	1
Laminar zone?	no
Porous zone?	no
Conical porous zone?	no
X-Component of Direction-1 Vector	1
Y-Component of Direction-1 Vector	1
Z-Component of Direction-1 Vector	1
X-Component of Direction-2 Vector	0
Y-Component of Direction-2 Vector	1
Z-Component of Direction-2 Vector	0
X-Coordinate of Point on Cone Axis	1
Y-Coordinate of Point on Cone Axis	0
Z-Coordinate of Point on Cone Axis	0
Half Angle of Cone Relative to its Axis	0
Direction-1 Viscous Resistance	0
Direction-2 Viscous Resistance	0
Direction-3 Viscous Resistance	0
Direction-1 Inertial Resistance	0
Direction-2 Inertial Resistance	0
Direction-3 Inertial Resistance	0
C0 Coefficient for Power-Law	0
C1 Coefficient for Power-Law	0
Porosity	1
Solid Material Name	aluminum

inlet1

Condition	Value

wall2-shadow

Condition	Value
Wall Thickness	0

Heat Generation Rate	0
Material Name	aluminum
Thermal BC Type	3
Temperature	300
Heat Flux	0
Convective Heat Transfer Coefficient	0
Free Stream Temperature	300
Enable shell conduction?	no
Wall Motion	0
Shear Boundary Condition	0
Define wall motion relative to adjacent cell zone?	yes
Apply a rotational velocity to this wall?	no
Velocity Magnitude	0
X-Component of Wall Translation	1
Y-Component of Wall Translation	0
Z-Component of Wall Translation	0
Define wall velocity components?	no
X-Component of Wall Translation	0
Y-Component of Wall Translation	0
Z-Component of Wall Translation	0
External Emissivity	1
External Radiation Temperature	300
Wall Roughness Height	0
Wall Roughness Constant	0.5
Rotation Speed	0
X-Position of Rotation-Axis Origin	0
Y-Position of Rotation-Axis Origin	0
Z-Position of Rotation-Axis Origin	0
X-Component of Rotation-Axis Direction	0
Y-Component of Rotation-Axis Direction	0
Z-Component of Rotation-Axis Direction	1
X-component of shear stress	0
Y-component of shear stress	0
Z-component of shear stress	0
Surface tension gradient	0

wsd-shadow

Condition	Value
Wall Thickness	0
Heat Generation Rate	0
Material Name	aluminum
Thermal BC Type	3
Temperature	300
Heat Flux	0
Convective Heat Transfer Coefficient	0
Free Stream Temperature	300
Enable shell conduction?	no
Wall Motion	0
Shear Boundary Condition	0
Define wall motion relative to adjacent cell zone?	yes
Apply a rotational velocity to this wall?	no
Velocity Magnitude	0

X-Component of Wall Translation	1
Y-Component of Wall Translation	0
Z-Component of Wall Translation	0
Define wall velocity components?	no
X-Component of Wall Translation	0
Y-Component of Wall Translation	0
Z-Component of Wall Translation	0
External Emissivity	1
External Radiation Temperature	300
Wall Roughness Height	0
Wall Roughness Constant	0.5
Rotation Speed	0
X-Position of Rotation-Axis Origin	0
Y-Position of Rotation-Axis Origin	0
Z-Position of Rotation-Axis Origin	0
X-Component of Rotation-Axis Direction	0
Y-Component of Rotation-Axis Direction	0
Z-Component of Rotation-Axis Direction	1
X-component of shear stress	0
Y-component of shear stress	0
Z-component of shear stress	0
Surface tension gradient	0

inti

Condition	Value

wsd

Condition	Value
Wall Thickness	0
Heat Generation Rate	0
Material Name	aluminum
Thermal BC Type	3
Temperature	300
Heat Flux	0
Convective Heat Transfer Coefficient	0
Free Stream Temperature	300
Enable shell conduction?	no
Wall Motion	0
Shear Boundary Condition	0
Define wall motion relative to adjacent cell zone?	yes
Apply a rotational velocity to this wall?	no
Velocity Magnitude	0
X-Component of Wall Translation	1
Y-Component of Wall Translation	0
Z-Component of Wall Translation	0
Define wall velocity components?	no
X-Component of Wall Translation	0
Y-Component of Wall Translation	0
Z-Component of Wall Translation	0
External Emissivity	1

External Radiation Temperature	300
Wall Roughness Height	0
Wall Roughness Constant	0.5
Rotation Speed	0
X-Position of Rotation-Axis Origin	0
Y-Position of Rotation-Axis Origin	0
Z-Position of Rotation-Axis Origin	0
X-Component of Rotation-Axis Direction	0
Y-Component of Rotation-Axis Direction	0
Z-Component of Rotation-Axis Direction	1
X-component of shear stress	0
Y-component of shear stress	0
Z-component of shear stress	0
Surface tension gradient	0

intu

Condition	Value
Wall Thickness	0
Heat Generation Rate	0
Material Name	aluminum
Thermal BC Type	1
Temperature	300
Heat Flux	0
Convective Heat Transfer Coefficient	0
Free Stream Temperature	300
Enable shell conduction?	no
Wall Motion	0
Shear Boundary Condition	0
Define wall motion relative to adjacent cell zone?	yes
Apply a rotational velocity to this wall?	no
Velocity Magnitude	0
X-Component of Wall Translation	1
Y-Component of Wall Translation	0
Z-Component of Wall Translation	0
Define wall velocity components?	no
X-Component of Wall Translation	0
Y-Component of Wall Translation	0
Z-Component of Wall Translation	0
External Emissivity	1
External Radiation Temperature	300
Wall Roughness Height	0
Wall Roughness Constant	0.5
Rotation Speed	0
X-Position of Rotation-Axis Origin	0
Y-Position of Rotation-Axis Origin	0
Z-Position of Rotation-Axis Origin	0
X-Component of Rotation-Axis Direction	0
Y-Component of Rotation-Axis Direction	0
Z-Component of Rotation-Axis Direction	1
X-component of shear stress	0
Y-component of shear stress	0
Z-component of shear stress	0

Surface tension gradient	0

inlet2

Condition	Value
Velocity Specification Method	2
Reference Frame	0
Velocity Magnitude	10
Coordinate System	0
X-Velocity	0
Y-Velocity	0
Z-Velocity	0
X-Component of Flow Direction	1
Y-Component of Flow Direction	0
Z-Component of Flow Direction	0
X-Component of Axis Direction	1
Y-Component of Axis Direction	0
Z-Component of Axis Direction	0
X-Coordinate of Axis Origin	0
Y-Coordinate of Axis Origin	0
Z-Coordinate of Axis Origin	0
Angular velocity	0
Temperature	300
Turbulence Specification Method	2
Turb. Kinetic Energy	1
Turb. Dissipation Rate	1
Turbulence Intensity	0.25
Turbulence Length Scale	1
Hydraulic Diameter	1
Turbulent Viscosity Ratio	25

out2

Condition	Value
Gauge Pressure	0
Radial Equilibrium Pressure Distribution	no
Backflow Total Temperature	300
Turbulence Specification Method	2
Backflow Turb. Kinetic Energy	1
Backflow Turb. Dissipation Rate	1
Backflow Turbulence Intensity	0.25
Backflow Turbulence Length Scale	1
Backflow Hydraulic Diameter	1
Backflow Turbulent Viscosity Ratio	25

right

Condition	Value
Wall Thickness	0
Heat Generation Rate	0
Material Name	aluminum

Thermal BC Type	1
Temperature	300
Heat Flux	0
Convective Heat Transfer Coefficient	0
Free Stream Temperature	300
Enable shell conduction?	no
Wall Motion	0
Shear Boundary Condition	0
Define wall motion relative to adjacent cell zone?	yes
Apply a rotational velocity to this wall?	no
Velocity Magnitude	0
X-Component of Wall Translation	1
Y-Component of Wall Translation	0
Z-Component of Wall Translation	0
Define wall velocity components?	no
X-Component of Wall Translation	0
Y-Component of Wall Translation	0
Z-Component of Wall Translation	0
External Emissivity	1
External Radiation Temperature	300
Wall Roughness Height	0
Wall Roughness Constant	0.5
Rotation Speed	0
X-Position of Rotation-Axis Origin	0
Y-Position of Rotation-Axis Origin	0
Z-Position of Rotation-Axis Origin	0
X-Component of Rotation-Axis Direction	0
Y-Component of Rotation-Axis Direction	0
Z-Component of Rotation-Axis Direction	1
X-component of shear stress	0
Y-component of shear stress	0
Z-component of shear stress	0
Surface tension gradient	0

bot

Condition	Value
Wall Thickness	0
Heat Generation Rate	0
Material Name	aluminum
Thermal BC Type	1
Temperature	300
Heat Flux	0
Convective Heat Transfer Coefficient	0
Free Stream Temperature	300
Enable shell conduction?	no
Wall Motion	0
Shear Boundary Condition	0
Define wall motion relative to adjacent cell zone?	yes
Apply a rotational velocity to this wall?	no
Velocity Magnitude	0
X-Component of Wall Translation	1
Y-Component of Wall Translation	0

Z-Component of Wall Translation	0
Define wall velocity components?	no
X-Component of Wall Translation	0
Y-Component of Wall Translation	0
Z-Component of Wall Translation	0
External Emissivity	1
External Radiation Temperature	300
Wall Roughness Height	0
Wall Roughness Constant	0.5
Rotation Speed	0
X-Position of Rotation-Axis Origin	0
Y-Position of Rotation-Axis Origin	0
Z-Position of Rotation-Axis Origin	0
X-Component of Rotation-Axis Direction	0
Y-Component of Rotation-Axis Direction	0
Z-Component of Rotation-Axis Direction	1
X-component of shear stress	0
Y-component of shear stress	0
Z-component of shear stress	0
Surface tension gradient	0

left

Condition	Value
Wall Thickness	0
Heat Generation Rate	0
Material Name	aluminum
Thermal BC Type	1
Temperature	300
Heat Flux	0
Convective Heat Transfer Coefficient	0
Free Stream Temperature	300
Enable shell conduction?	no
Wall Motion	0
Shear Boundary Condition	0
Define wall motion relative to adjacent cell zone?	yes
Apply a rotational velocity to this wall?	no
Velocity Magnitude	0
X-Component of Wall Translation	1
Y-Component of Wall Translation	0
Z-Component of Wall Translation	0
Define wall velocity components?	no
X-Component of Wall Translation	0
Y-Component of Wall Translation	0
Z-Component of Wall Translation	0
External Emissivity	1
External Radiation Temperature	300
Wall Roughness Height	0
Wall Roughness Constant	0.5
Rotation Speed	0
X-Position of Rotation-Axis Origin	0
Y-Position of Rotation-Axis Origin	0
Z-Position of Rotation-Axis Origin	0

X-Component of Rotation-Axis Direction	0
Y-Component of Rotation-Axis Direction	0
Z-Component of Rotation-Axis Direction	1
X-component of shear stress	0
Y-component of shear stress	0
Z-component of shear stress	0
Surface tension gradient	0

top

Condition	Value
Wall Thickness	0
Heat Generation Rate	0
Material Name	aluminum
Thermal BC Type	1
Temperature	300
Heat Flux	0
Convective Heat Transfer Coefficient	0
Free Stream Temperature	300
Enable shell conduction?	no
Wall Motion	0
Shear Boundary Condition	0
Define wall motion relative to adjacent cell zone?	yes
Apply a rotational velocity to this wall?	no
Velocity Magnitude	0
X-Component of Wall Translation	1
Y-Component of Wall Translation	0
Z-Component of Wall Translation	0
Define wall velocity components?	no
X-Component of Wall Translation	0
Y-Component of Wall Translation	0
Z-Component of Wall Translation	0
External Emissivity	1
External Radiation Temperature	300
Wall Roughness Height	0
Wall Roughness Constant	0.5
Rotation Speed	0
X-Position of Rotation-Axis Origin	0
Y-Position of Rotation-Axis Origin	0
Z-Position of Rotation-Axis Origin	0
X-Component of Rotation-Axis Direction	0
Y-Component of Rotation-Axis Direction	0
Z-Component of Rotation-Axis Direction	1
X-component of shear stress	0
Y-component of shear stress	0
Z-component of shear stress	0
Surface tension gradient	0

wall2

Condition	Value
Wall Thickness	0

Heat Generation Rate	0
Material Name	aluminum
Thermal BC Type	3
Temperature	300
Heat Flux	0
Convective Heat Transfer Coefficient	0
Free Stream Temperature	300
Enable shell conduction?	no
Wall Motion	0
Shear Boundary Condition	0
Define wall motion relative to adjacent cell zone?	yes
Apply a rotational velocity to this wall?	no
Velocity Magnitude	0
X-Component of Wall Translation	1
Y-Component of Wall Translation	0
Z-Component of Wall Translation	0
Define wall velocity components?	no
X-Component of Wall Translation	0
Y-Component of Wall Translation	0
Z-Component of Wall Translation	0
External Emissivity	1
External Radiation Temperature	300
Wall Roughness Height	0
Wall Roughness Constant	0.5
Rotation Speed	0
X-Position of Rotation-Axis Origin	0
Y-Position of Rotation-Axis Origin	0
Z-Position of Rotation-Axis Origin	0
X-Component of Rotation-Axis Direction	0
Y-Component of Rotation-Axis Direction	0
Z-Component of Rotation-Axis Direction	1
X-component of shear stress	0
Y-component of shear stress	0
Z-component of shear stress	0
Surface tension gradient	0

out1

Condition	Value

wall1

Condition	Value
Wall Thickness	0
Heat Generation Rate	0
Material Name	aluminum
Thermal BC Type	1
Temperature	300
Heat Flux	0
Convective Heat Transfer Coefficient	0
Free Stream Temperature	300
Enable shell conduction?	no

```
      Wall Motion                                          0
      Shear Boundary Condition                             0
      Define wall motion relative to adjacent cell zone?   yes
      Apply a rotational velocity to this wall?            no
      Velocity Magnitude                                   0
      X-Component of Wall Translation                      1
      Y-Component of Wall Translation                      0
      Z-Component of Wall Translation                      0
      Define wall velocity components?                     no
      X-Component of Wall Translation                      0
      Y-Component of Wall Translation                      0
      Z-Component of Wall Translation                      0
      External Emissivity                                  1
      External Radiation Temperature                       300
      Wall Roughness Height                                0
      Wall Roughness Constant                              0.5
      Rotation Speed                                       0
      X-Position of Rotation-Axis Origin                   0
      Y-Position of Rotation-Axis Origin                   0
      Z-Position of Rotation-Axis Origin                   0
      X-Component of Rotation-Axis Direction               0
      Y-Component of Rotation-Axis Direction               0
      Z-Component of Rotation-Axis Direction               1
      X-component of shear stress                          0
      Y-component of shear stress                          0
      Z-component of shear stress                          0
      Surface tension gradient                             0

   default-interior

      Condition   Value
      -----------------

   default-interior:001

      Condition   Value
      -----------------

   default-interior:018

      Condition   Value
      -----------------

Solver Controls
---------------

   Equations

      Equation     Solved
      -------------------
      Flow         yes
      Turbulence   yes
      Energy       yes
```

Numerics

Numeric	Enabled
Absolute Velocity Formulation	yes

Relaxation

Variable	Relaxation Factor
Pressure	0.30000001
Density	0.80000001
Body Forces	0.69999999
Momentum	0.2
Turbulence Kinetic Energy	0.80000001
Turbulence Dissipation Rate	0.80000001
Turbulent Viscosity	0.80000001
Energy	0.80000001

Linear Solver

Variable	Solver Type	Termination Criterion	Residual Reduction Tolerance
Pressure	V-Cycle	0.1	
X-Momentum	Flexible	0.1	0.7
Y-Momentum	Flexible	0.1	0.7
Z-Momentum	Flexible	0.1	0.7
Turbulence Kinetic Energy	Flexible	0.1	0.7
Turbulence Dissipation Rate	Flexible	0.1	0.7
Energy	Flexible	0.1	0.7

Discretization Scheme

Variable	Scheme
Pressure	Standard
Pressure-Velocity Coupling	SIMPLE
Momentum	First Order Upwind
Turbulence Kinetic Energy	First Order Upwind
Turbulence Dissipation Rate	First Order Upwind
Energy	First Order Upwind

Solution Limits

Quantity	Limit
Minimum Absolute Pressure	1
Maximum Absolute Pressure	5000000
Minimum Temperature	1
Maximum Temperature	5000
Minimum Turb. Kinetic Energy	9.9999998e-15

```
      Minimum Turb. Dissipation Rate    9.9999997e-21
      Maximum Turb. Viscosity Ratio     1000000

Material Properties
-------------------

   Material: air (fluid)

      Property                          Units      Method
Value(s)
      ---------------------------------------------------------------------
----------------
      Density                           kg/m3      incompressible-ideal-gas
#f
      Cp (Specific Heat)                j/kg-k     constant
1006.43
      Thermal Conductivity              w/m-k      constant
0.0242
      Viscosity                         kg/m-s     constant
1.7894001e-05
      Molecular Weight                  kg/kgmol   constant
28.966
      L-J Characteristic Length         angstrom   constant
3.711
      L-J Energy Parameter              k          constant
78.6
      Thermal Expansion Coefficient     1/k        constant
0
      Degrees of Freedom                           constant
0

   Material: aluminum (solid)

      Property                Units     Method     Value(s)
      ---------------------------------------------------
      Density                 kg/m3     constant   2719
      Cp (Specific Heat)      j/kg-k    constant   871
      Thermal Conductivity    w/m-k     constant   202.4
```

ضمیمه ب – برنامه کامپیوتری

```
      program sharp

      use portlib

      include 'params.cmd'
      include 'precs.cmd'
      include 'arrays.cmd'
      include 'param.cmd'
      include 'bcon.cmd'

      real*8 cputim

      external blkdat

      namelist /fname/ name,fngrd,fninp,fnout,fnplt,fnrst,fnsav,fndlt,fnrs1, &
               fnrs2,fnrs3,fnrs4,fnrs5,dfgrd,dfinp,dfout,dfplt,dfrst, &
               dfsav,dfdlt,dfrs1,dfrs2,dfrs3,dfrs4,dfrs5

      namelist /logic/ restrt,savers,steady,source,wshear
      namelist /phys/ gamma,rgas,cyl,gx,gy,scaleg
      namelist /init/ pi,ui,vi,ti
      namelist /tim/ autot,cfl,delt,implct,dampi,rlxq,epsi,fcycle,twsta,twfin
      namelist /visc/ invisd,cmu,cku,ivisc,iturb,itcon,sc1,sc2,prt
      namelist /split/ isplit,iflux,rlx,lspeed,damp,jupdat
      namelist /minmod/ inorxi,inoret,limit,dlim
      namelist /other/ lgs,itord,rdxi,rdet,epsig,iflip

      namelist /post/ ifreq,ianim,iplot,pltdt,isympl,iprint,prtdt,isavfm,irstfm, &
              velmx,rstdt

      namelist /bcs/ wtlb,ptlb,ttlb,uxlb,uylb,pslb,tslb,ulb,vlb,rplb,brlb, &
              wtrb,ptrb,ttrb,uxrb,uyrb,psrb,tsrb,urb,vrb,rprb,brrb, &
              wtbb,ptbb,ttbb,uxbb,uybb,psbb,tsbb,ubb,vbb,rpbb,brbb, &
              wttb,pttb,tttb,uxtb,uytb,pstb,tstb,utb,vtb,rptb,brtb,impbc

      include 'aindex.cmd'
!
! initialize the cpu timer
!
      cputim=timef()
!
! open input unit file
!
      call ocfile (luinp,fninp,dfinp,'formatted  ',1)
!
! read problem title and file names
!
      read(luinp,nml=fname)
!
! read all the logical variables
!
      read(luinp,nml=logic)
!
! read physical properties
!
      read(luinp,nml=phys)
!
! read initial conditions
!
```

```
  read(luinp,nml=init)
!
! read time control parameters
!
  read(luinp,nml=tim)
!
! read viscous parameters
!
  read(luinp,nml=visc)
!
! read space and flux split parameters
!
  read(luinp,nml=split)
!
! read minmod interpolation parameters
!
  read(luinp,nml=minmod)
!
! read other parameters
!
  read(luinp,nml=other)
!
! read post processing parameters
!
  read(luinp,nml=post)
!
! read boundary condition data and close input unit file
!
  read(luinp,nml=bcs)
!
! close input unit file
!
  call ocfile (luinp,fninp,dfinp,'formatted  ',0)
!
! open output unit file
!
  call ocfile (luout,fnout,dfout,'formatted  ',1)
!
! print initial input data
!
   call print(1)
!
! read input parameters for mesh and find all geometrical parameters
!
  call metric
!
! set initial conditions
!
  call setup
!
! read the initial field from restart file
!
  if (cycle.le.fcycle .and. restrt) then
    call readfi

    do j=1,jmax
    do i=1,imax
      ij=ind(i,j)
      mulm(ij)=cmu
      mu(ij)=cmu
      kulm(ij)=cku
      ku(ij)=cku
```

```
    enddo
    enddo
  endif
!
! set initial boundary conditions and print
!
  call bcexp
  call print (3)
!
! open plot unit files
!
  call ocfile (luplt,fnplt,dfplt,'formatted  ',1)
  call ocfile (ludlt,fndlt,dfdlt,'formatted  ',1)
  call ocfile (lurs1,fnrs1,dfrs1,'formatted  ',1)
  call ocfile (lurs2,fnrs2,dfrs2,'formatted  ',1)
  call ocfile (lurs3,fnrs3,dfrs3,'formatted  ',1)
  call ocfile (lurs4,fnrs4,dfrs4,'formatted  ',1)
  call ocfile (lurs5,fnrs5,dfrs5,'formatted  ',1)
!
! prepare title for tecplot
!
  write(luplt,'(a,a80,a)') ' Title = " ',name,' " '
  write(luplt,'(a)') ' Variables = x,y,ro,u,v,p,t,m,to,i,j,ij'

  write(ludlt,'(a,a80,a)') ' Title = " ',name,' " '
  write(lurs1,'(a,a80,a)') ' Title = " ',name,' " '
  write(lurs2,'(a,a80,a)') ' Title = " ',name,' " '
  write(lurs3,'(a,a80,a)') ' Title = " ',name,' " '
  write(lurs4,'(a,a80,a)') ' Title = " ',name,' " '
  write(lurs5,'(a,a80,a)') ' Title = " ',name,' " '

  write(ludlt,'(a)') ' Variables = cycle, delt'
  write(lurs1,'(a)') ' Variables = cycle, rol2'
  write(lurs2,'(a)') ' Variables = cycle, rul2'
  write(lurs3,'(a)') ' Variables = cycle, rvl2'
  write(lurs4,'(a)') ' Variables = cycle, el2'
  write(lurs5,'(a)') ' Variables = cycle, tl2'

  write(ludlt,'(a)') ' Zone'
  write(lurs1,'(a)') ' Zone'
  write(lurs2,'(a)') ' Zone'
  write(lurs3,'(a)') ' Zone'
  write(lurs4,'(a)') ' Zone'
  write(lurs5,'(a)') ' Zone'

  go to 2
!
! start time cycle
!
1 ttime=ttime+delt
  cycle=cycle+1
!
! find local minimum delta and print it on paper
!
  call delta

  if (delt.le.em10) then
    write(luscn,'(a)') ' emergency stop, time step out of control'
    write(luout,'(a)') ' emergency stop, time step out of control'

    go to 3
  endif
```

```
 call print (2)
!
! evaluate the jacobian matrices for implicit operator
!
 if (implct.gt.0) call caljac
!
! explicit evaluation of the steady state term
!
 call calstd
!
! solve the governing equations
!
 call solve
!
! calculate the primitive variables
!
 call calprm
!
! set boundary conditions
!
 call bcexp
!
! update the viscosity and thermal conductivity fields
!
2 if (invisd.lt.1) then
    if (ivisc.gt.0) call viscl
    if (iturb.gt.0) call visct
    if (itcon.gt.0) call tdifl
    if (iturb.gt.0) call tdift
 endif
!
! check for steady state convergence
!
 call convrg (er1,er2,er3,er4,sl2)
!
! set the advance time arrays into the time-n arrays
!
 do j=1,jmax
 do i=1,imax
   ij=ind(i,j)
   dqn1(ij)=dq1(ij)
   dqn2(ij)=dq2(ij)
   dqn3(ij)=dq3(ij)
   dqn4(ij)=dq4(ij)
   dq1(ij)=0.0
   dq2(ij)=0.0
   dq3(ij)=0.0
   dq4(ij)=0.0
 enddo
 enddo
!
! check to see if final calculational time twfin is reached
!
 if (ttime.ge.twfin .or. iflag.ge.1) then
   if ((iplot.eq.2 .or. iplot.eq.3) .and. ianim.eq.1) then
     write(luplt,'(2a,i3,a,i3)') ' Zone',' i= ',im1,' j= ',jm1

     call wrtplt
   endif

   call print (3)
```

```
    if (iflag.gt.0) then
      if (iprint.eq.0) write(luscn,'(a,1pe12.5)') &
        ' steady state solution reached at time = ',ttime
      if (iprint.eq.1 .or. iprint.eq.3) write(luscn,'(a,1pe12.5)') &
        ' steady state solution reached at time = ',ttime
      if (iprint.eq.2 .or. iprint.eq.3) write(luout,'(a,1pe12.5)') &
        ' steady state solution reached at time = ',ttime
    endif

    go to 3
  endif
!
! plot velocity vector and mesh
!
  if (cycle.le.fcycle) then
    if ((iplot.eq.2 .or. iplot.eq.3) .and. ianim.eq.1) then
      write(luplt,'(2a,i3,a,i3)') 'Zone',' i= ',im1,' j= ',jm1

      call wrtplt
    endif
  else
    if (ttime+em10.ge.twplt) then
      twplt=twplt+pltdt

      if ((iplot.eq.2 .or. iplot.eq.3) .and. ianim.eq.1) then
        write(luplt,'(2a,i3,a,i3)') ' Zone',' i= ',im1,' j= ',jm1

        call wrtplt
      endif
    endif
  endif
!
! print field variable data on paper and/or screen
!
  if (cycle.le.fcycle) then
    call print (3)
  else
    if (ttime+em6.ge.twprt) then
      twprt=twprt+prtdt

      call print (3)
    endif
  endif
!
! write residual errors and deltat
!
  if (cycle.gt.0) then
    erlog1=slog10(er1)
    erlog2=slog10(er2)
    erlog3=slog10(er3)
    erlog4=slog10(er4)
    erlog5=slog10(sl2)

    write(ludlt,'(1x,i6,1pe12.3)') cycle,delt
    write(lurs1,'(1x,i6,1pe12.3)') cycle,erlog1
    write(lurs2,'(1x,i6,1pe12.3)') cycle,erlog2
    write(lurs3,'(1x,i6,1pe12.3)') cycle,erlog3
    write(lurs4,'(1x,i6,1pe12.3)') cycle,erlog4
    write(lurs5,'(1x,i6,1pe12.3)') cycle,erlog5
  endif
!
```

```
! advance cycle
!
  if (mod(cycle,ifreq).eq.0) then
    write(luscn,'(/,6x,a,1pe12.5,5x,a,1pe12.5,5x,a,i5)') &
        ' time= ',ttime,' delt= ',delt,' cycle= ',cycle

    write(luscn,'(a)') ' L2-norm of residuals :'
    write(luscn,'(a,1pe11.3)') ' mass       =',er1
    write(luscn,'(a,1pe11.3)') ' x-momentum =',er2
    write(luscn,'(a,1pe11.3)') ' y-momentum =',er3
    write(luscn,'(a,1pe11.3)') ' energy     =',er4
    write(luscn,'(a,1pe11.3)') ' L2-norm    =',sl2

    write(luscn,*) 'do you want to stop now ?'

    read(5,*) istop

    if (istop.eq.0) then
      go to 1
    else
      if (iplot.eq.2 .or. iplot.eq.3) then
        write(luplt,'(2a,i3,a,i3)') ' Zone',' i= ',im1,' j= ',jm1

        call wrtplt
      endif

      call print (3)

      go to 3
    endif
  endif

  if (cycle.eq.fcycle) go to 1
  if (iflag.lt.1) go to 1
!
! write the final field into save file
!
3 if (savers) call writfi
!
! calculate specific impulse Isp
!
  call calisp
!
! find the grind time
!
  cputim=timef()

  idcycl=cycle-fcycle
  if (idcycl.le.0) idcycl=1
  ijmaxc=idcycl*imax*jmax
  timeef=cputim/sfloat(ijmaxc)

  write(luscn,'(a,1pe12.5,a)') ' total cpu time = ',cputim,' seconds'
  write(luscn,'(a,1pe12.5,a)') &
     ' computational effort for this run was ',timeef,' second'
  write(luout,'(a,1pe12.5,a)') ' total cpu time = ',cputim,' seconds'
  write(luout,'(a,1pe12.5,a)') &
     ' computational effort for this run was ',timeef,' second'

  write(luscn,'(a)') ' L2-norm of residuals :'
  write(luscn,'(a,1pe11.3)') ' mass       =',er1
  write(luscn,'(a,1pe11.3)') ' x-momentum =',er2
```

```
 write(luscn,'(a,1pe11.3)') ' y-momentum =',er3
 write(luscn,'(a,1pe11.3)') ' energy     =',er4
 write(luscn,'(a,1pe11.3)') ' L2-norm    =',sl2

 write(luout,'(a)') ' L2-norm of residuals :'
 write(luout,'(a,1pe11.3)') ' mass       =',er1
 write(luout,'(a,1pe11.3)') ' x-momentum =',er2
 write(luout,'(a,1pe11.3)') ' y-momentum =',er3
 write(luout,'(a,1pe11.3)') ' energy     =',er4
 write(luout,'(a,1pe11.3)') ' L2-norm    =',sl2
!
! close plot unit files
!
 call ocfile (luplt,fnplt,dfplt,'formatted  ',0)
 call ocfile (ludlt,fndlt,dfdlt,'formatted  ',0)
 call ocfile (lurs1,fnrs1,dfrs1,'formatted  ',0)
 call ocfile (lurs2,fnrs2,dfrs2,'formatted  ',0)
 call ocfile (lurs3,fnrs3,dfrs3,'formatted  ',0)
 call ocfile (lurs4,fnrs4,dfrs4,'formatted  ',0)
 call ocfile (lurs5,fnrs5,dfrs5,'formatted  ',0)

 stop
 end


 subroutine ab1t11
!
!*****************************************************************************
! at the right face of each cell this subroutine calculates constants a1
! through a11 which are needed in eveluation of ev vector and dev matrix. also
! at the top face of each cell it calculates constants b1 through b11 which are
! needed in eveluation of fv vector and dfv matrix.
!*****************************************************************************
!
 include 'params.cmd'
 include 'precs.cmd'
 include 'arrays.cmd'
 include 'param.cmd'
 include 'bcon.cmd'

 include 'aindex.cmd'

 real*8 jacw

 yrht=1.0
 ytop=1.0

 do j=1,jm1
 do i=1,im1
   ij=ind(i,j)
   imj=ij-1
   ipj=ij+1
   ijm=ij-imax
   ijp=ij+imax
   imjm=ij-1-imax
   ipjm=ij+1-imax
   imjp=ij-1+imax
!
! compute a1 throught a11
!
   if (j.ne.1) then
     if (cyl.gt.0.0) yrht=0.5*(y(ij)+y(ijm))
```

```
if (yrht.lt.em10) yrht=ep10*ep10

if (i.eq.1) then
  arew=arer(ij)/jac(ipj)
  ij=ind(i+1,j)

  call geom (2,0,unx,uny,area,volt1,volt2)

  jacw=volt1+volt2
  ij=ind(i,j)
elseif (i.eq.im1) then
  arew=arer(ij)/jac(ij)

  call geom (2,0,unx,uny,area,volt1,volt2)

  jacw=volt1+volt2
else
  arew=2.0*arer(ij)/(jac(ij)+jac(ipj))

  call geom (2,0,unx,uny,area,volt1,volt2)

  jacw=volt1+volt2
endif

if (i.eq.1) then
  yw=(y(ij)+y(ijm))/2.0
  yr=(y(ij)+y(ijm)+y(ipj)+y(ipjm))/4.0
  yrr=(y(ipj)+y(ipjm))/2.0
  yxir=(-yrr+4.0*yr-3.0*yw)
  xw=(x(ij)+x(ijm))/2.0
  xr=(x(ij)+x(ijm)+x(ipj)+x(ipjm))/4.0
  xrr=(x(ipj)+x(ipjm))/2.0
  xxir=(-xrr+4.0*xr-3.0*xw)
elseif (i.eq.im1) then
  yw=(y(ij)+y(ijm))/2.0
  yl=(y(ij)+y(ijm)+y(imj)+y(imjm))/4.0
  yll=(y(imj)+y(imjm))/2.0
  yxir=(yll-4.0*yl+3.0*yw)
  xw=(x(ij)+x(ijm))/2.0
  xl=(x(ij)+x(ijm)+x(imj)+x(imjm))/4.0
  xll=(x(imj)+x(imjm))/2.0
  xxir=(xll-4.0*xl+3.0*xw)
elseif (i.ne.1 .and. i.ne.im1) then
  yl=(y(ij)+y(ijm)+y(imj)+y(imjm))/4.0
  yr=(y(ij)+y(ijm)+y(ipj)+y(ipjm))/4.0
  yxir=(yr-yl)
  xl=(x(ij)+x(ijm)+x(imj)+x(imjm))/4.0
  xr=(x(ij)+x(ijm)+x(ipj)+x(ipjm))/4.0
  xxir=(xr-xl)
endif

etxr=-yxir/jacw
etyr=xxir/jacw

a1(ij)=(1.0+c1*unxr(ij)*unxr(ij))*arew*arer(ij)
a2(ij)=c1*unxr(ij)*unyr(ij)*arew*arer(ij)
a3(ij)=(1.0+c1*unyr(ij)*unyr(ij))*arew*arer(ij)
a4(ij)=arew*arer(ij)
a5(ij)=(2.0*c2*unxr(ij)*etxr+unyr(ij)*etyr)*arer(ij)
a6(ij)=(-c2*unxr(ij)*etyr+unyr(ij)*etxr)*arer(ij)
a7(ij)=-c2*cyl*unxr(ij)/yrht*arer(ij)
a8(ij)=(unxr(ij)*etyr-c2*unyr(ij)*etxr)*arer(ij)
```

```
      a9(ij)=(unxr(ij)*etxr+2.0*c2*unyr(ij)*etyr)*arer(ij)
      a10(ij)=-c2*cyl*unyr(ij)/yrht*arer(ij)
      a11(ij)=(unxr(ij)*etxr+unyr(ij)*etyr)*arer(ij)
    endif
!
! compute b1 throught b11
!
    if (i.ne.1) then
      if (cyl.gt.0.0) ytop=0.5*(y(ij)+y(imj))
      if (ytop.lt.em10) ytop=ep10*ep10

      if (j.eq.1) then
        arew=aret(ij)/jac(ijp)
        ij=ind(i,j+1)

        call geom (2,0,unx,uny,area,volt1,volt2)

        jacw=volt1+volt2
        ij=ind(i,j)
      elseif (j.eq.jm1) then
        arew=aret(ij)/jac(ij)

        call geom (2,0,unx,uny,area,volt1,volt2)

        jacw=volt1+volt2
      else
        arew=2.0*aret(ij)/(jac(ij)+jac(ijp))

        call geom (2,0,unx,uny,area,volt1,volt2)

        jacw=volt1+volt2
      endif

      if (j.eq.1) then
        yw=(y(ij)+y(imj))/2.0
        yt=(y(ij)+y(imj)+y(ijp)+y(imjp))/4.0
        ytt=(y(ijp)+y(imjp))/2.0
        yett=(-ytt+4.0*yt-3.0*yw)
        xw=(x(ij)+x(imj))/2.0
        xt=(x(ij)+x(imj)+x(ijp)+x(imjp))/4.0
        xtt=(x(ijp)+x(imjp))/2.0
        xett=(-xtt+4.0*xt-3.0*xw)
      elseif (j.eq.jm1) then
        yw=(y(ij)+y(imj))/2.0
        yb=(y(ij)+y(imj)+y(ijm)+y(imjm))/4.0
        ybb=(y(ijm)+y(imjm))/2.0
        yett=(ybb-4.0*yb+3.0*yw)
        xw=(x(ij)+x(imj))/2.0
        xb=(x(ij)+x(imj)+x(ijm)+x(imjm))/4.0
        xbb=(x(ijm)+x(imjm))/2.0
        xett=(xbb-4.0*xb+3.0*xw)
      elseif (j.ne.1 .and. j.ne.jm1) then
        yb=(y(ij)+y(imj)+y(ijm)+y(imjm))/4.0
        yt=(y(ij)+y(imj)+y(ijp)+y(imjp))/4.0
        yett=(yt-yb)
        xb=(x(ij)+x(imj)+x(ijm)+x(imjm))/4.0
        xt=(x(ij)+x(imj)+x(ijp)+x(imjp))/4.0
        xett=(xt-xb)
      endif

      xixt=yett/jacw
      xiyt=-xett/jacw
```

```
     b1(ij)=(1.0+c1*unxt(ij)*unxt(ij))*arew*aret(ij)
     b2(ij)=c1*unxt(ij)*unyt(ij)*arew*aret(ij)
     b3(ij)=(1.0+c1*unyt(ij)*unyt(ij))*arew*aret(ij)
     b4(ij)=arew*aret(ij)
     b5(ij)=(2.*c2*xixt*unxt(ij)+xiyt*unyt(ij))*aret(ij)
     b6(ij)=(-c2*xiyt*unxt(ij)+xixt*unyt(ij))*aret(ij)
     b7(ij)=-c2*cyl*unxt(ij)/ytop*aret(ij)
     b8(ij)=(-c2*xixt*unyt(ij)+xiyt*unxt(ij))*aret(ij)
     b9(ij)=(xixt*unxt(ij)+2.0*c2*xiyt*unyt(ij))*aret(ij)
     b10(ij)=-c2*cyl*unyt(ij)/ytop*aret(ij)
     b11(ij)=(xixt*unxt(ij)+xiyt*unyt(ij))*aret(ij)
    endif
  enddo
  enddo

  return
  end
  subroutine abcdet (jl,jh)
!
!*******************************************************************************
! this subroutine forms the matrices aa, bb, cc, d for each constant xi  -line
! over the range of j = jl to jh.
!*******************************************************************************
!
  include 'params.cmd'
  include 'precs.cmd'
  include 'arrays.cmd'
  include 'param.cmd'
  include 'bcon.cmd'

  dimension aal(16), ccl(16), cij(16), cijm(16), cijp(16), dij(4), delq(4)

  include 'aindex.cmd'

  do j=jl,jh
    ij=ind(i,j)
    indd=(j-1)*4
    rdtja=jac(ij)/dt(ij)

    dd(indd+1)=dq1(ij)*rdtja
    dd(indd+2)=dq2(ij)*rdtja
    dd(indd+3)=dq3(ij)*rdtja
    dd(indd+4)=dq4(ij)*rdtja

    if (lgs.gt.0) then
      dd(indd+1)=omg1(ij)
      dd(indd+2)=omg2(ij)
      dd(indd+3)=omg3(ij)
      dd(indd+4)=omg4(ij)
    endif

    do kj=1,4
    do ki=1,4
      kikj=ki+(kj-1)*4
      kikjj=kikj+(j-1)*16
      indx=kikj+(ij-1)*16
      indym=indx-imax*16
      unity=0.0
      if (ki.eq.kj) unity=rdtja
      aa(kikjj)=-bp(indym)
      bb(kikjj)=unity+bp(indx)-bm(indym)-rlxm*dhdq(indx)
```

```
      cc(kikjj)=bm(indx)

      if (lgs.gt.0) then
        indxm=indx-16
        aal(kikj)=-ap(indxm)
        bb(kikjj)=bb(kikjj)+ap(indx)-am(indxm)-rlx*dhdq(indx)
        ccl(kikj)=am(indx)
        cij(kikj)=bb(kikjj)
        cijm(kikj)=aa(kikjj)
        cijp(kikj)=cc(kikjj)
      endif
    enddo
    enddo

    if (lgs.gt.0) then
      imj=ij-1
      delq(1)=dq1(imj)
      delq(2)=dq2(imj)
      delq(3)=dq3(imj)
      delq(4)=dq4(imj)

      call multmv (aal,delq,delq,4,4)

      dd(indd+1)=dd(indd+1)-delq(1)
      dd(indd+2)=dd(indd+2)-delq(2)
      dd(indd+3)=dd(indd+3)-delq(3)
      dd(indd+4)=dd(indd+4)-delq(4)

      ipj=ij+1
      delq(1)=dq1(ipj)
      delq(2)=dq2(ipj)
      delq(3)=dq3(ipj)
      delq(4)=dq4(ipj)

      call multmv (ccl,delq,delq,4,4)

      dd(indd+1)=dd(indd+1)-delq(1)
      dd(indd+2)=dd(indd+2)-delq(2)
      dd(indd+3)=dd(indd+3)-delq(3)
      dd(indd+4)=dd(indd+4)-delq(4)

      dij(1)=dd(indd+1)
      dij(2)=dd(indd+2)
      dij(3)=dd(indd+3)
      dij(4)=dd(indd+4)

      call errors (cij,cijm,cijp,dij)
    endif
  enddo

  return
  end

  subroutine abcdxi (il,ih)
!
!*****************************************************************************
! this subroutine forms the matrices aa, bb, cc,dd for each constant et  -line
! over the range of i = il to ih.
!*****************************************************************************
!
  include 'params.cmd'
  include 'precs.cmd'
```

```
include 'arrays.cmd'
include 'param.cmd'
include 'bcon.cmd'

dimension aal(16), ccl(16), delq(4)

include 'aindex.cmd'

do i=il,ih
  ij=ind(i,j)
  indd=(i-1)*4
  rdtja=jac(ij)/dt(ij)

  dd(indd+1)=omg1(ij)
  dd(indd+2)=omg2(ij)
  dd(indd+3)=omg3(ij)
  dd(indd+4)=omg4(ij)

  do kj=1,4
  do ki=1,4
    kikj=ki+(kj-1)*4
    kikji=kikj+(i-1)*16
    indx=kikj+(ij-1)*16
    indxm=indx-16
    unity=0.0
    if (ki.eq.kj) unity=rdtja
    aa(kikji)=-ap(indxm)
    bb(kikji)=unity+ap(indx)-am(indxm)-rlx*dhdq(indx)
    cc(kikji)=am(indx)

    if (lgs.gt.0) then
      indym=indx-imax*16
      aal(kikj)=-bp(indym)
      bb(kikji)=bb(kikji)+bp(indx)-bm(indym)-rlxm*dhdq(indx)
      ccl(kikj)=bm(indx)
    endif
 enddo
 enddo

 if (lgs.gt.0) then
   ijm=ij-imax
   delq(1)=dq1(ijm)
   delq(2)=dq2(ijm)
   delq(3)=dq3(ijm)
   delq(4)=dq4(ijm)

   call multmv (aal,delq,delq,4,4)

   dd(indd+1)=dd(indd+1)-delq(1)
   dd(indd+2)=dd(indd+2)-delq(2)
   dd(indd+3)=dd(indd+3)-delq(3)
   dd(indd+4)=dd(indd+4)-delq(4)

   ijp=ij+imax
   delq(1)=dq1(ijp)
   delq(2)=dq2(ijp)
   delq(3)=dq3(ijp)
   delq(4)=dq4(ijp)

   call multmv (ccl,delq,delq,4,4)

   dd(indd+1)=dd(indd+1)-delq(1)
```

```
      dd(indd+2)=dd(indd+2)-delq(2)
      dd(indd+3)=dd(indd+3)-delq(3)
      dd(indd+4)=dd(indd+4)-delq(4)
    endif
  enddo

  return
  end

  subroutine adcell
!
!*****************************************************************************
! this subroutine forms the matrices a and d for each cell (i,j) which is used
! in the evaluation of source term correction.
!*****************************************************************************
!
  include 'params.cmd'
  include 'precs.cmd'
  include 'arrays.cmd'
  include 'param.cmd'
  include 'bcon.cmd'

  rdtja=jac(ij)/dt(ij)

  d(1)=omg1(ij)
  d(2)=omg2(ij)
  d(3)=omg3(ij)
  d(4)=omg4(ij)
!
! get the jacobian dh/dq
!
  call caldhq

  do kj=1,4
  do ki=1,4
    kikj=ki+(kj-1)*4
    ind=kikj+(ij-1)*16
    unity=0.0
    if (ki.eq.kj) unity=rdtja
    a(kikj)=unity-dhdq(ind)
  enddo
  enddo

  return
  end

  subroutine addsrc
!
!*****************************************************************************
! this subroutine calculates the extra source terms.
!*****************************************************************************
!
  include 'params.cmd'
  include 'precs.cmd'
  include 'arrays.cmd'
  include 'param.cmd'
  include 'bcon.cmd'

  include 'aindex.cmd'
!
! top wall sources
!
```

```
 j=jm1

 do i=2,im1
   ij=ind(i,j)
!
! internal source boundary condition
!
   if (wttb(i).le.-1) then
     unxw=unxt(ij)
     unyw=unyt(ij)
     arew=aret(ij)
     pcell=p(ij)
     tcell=tstb(i)
     dcell=pcell/rgas/tcell
     rvw=rptb(i)*brtb(i)
     vw=rvw/dcell
     ucell=-unxw*vw
     vcell=-unyw*vw
     mdot=arew*rvw
     momu=arew*(rvw*ucell)
     momv=arew*(rvw*vcell)
     ener=mdot*(cpg*tcell+0.5*vw*vw)
     omg1(ij)=omg1(ij)+mdot
     omg2(ij)=omg2(ij)+momu
     omg3(ij)=omg3(ij)+momv
     omg4(ij)=omg4(ij)+ener
   endif
 enddo

 return
 end

 subroutine bcexp
!
!*******************************************************************************
! set boundary conditions
!
! option  0: symmetry plane
! option  1: rigid free-slip boundary condition with shear.
! option  2: rigid no-slip boundary condition with no shear.
! option  3: driven constant velocity lead boundary (cavity).
!
! option  4: subsonic inflow boundary condition, pt, tt, theta.
! option  5: subsonic inflow boundary condition, ts, u, v.
! option  6: subsonic inflow boundary condition, ps, u, v.
!
! option 10: supersonic inflow boundary condition.
!
! option 11: outflow boundary condition.
!
! option 15: mass injection bondary condition, isentropic
! option 16: mass injection bondary condition, non-isentropic
!*******************************************************************************
!
 include 'params.cmd'
 include 'precs.cmd'
 include 'arrays.cmd'
 include 'param.cmd'
 include 'bcon.cmd'

 include 'aindex.cmd'
```

```
!
! left wall
!
 i=1

 do j=2,jm1
   ij=ind(i,j)

   if (wtlb(j).ne.10) call calqpm ('r',inorxi)
!
! free-slip boundary condition
!
   if (wtlb(j).le.1) then
     p(ij)=pp
     ubar=up*unxr(ij)+vp*unyr(ij)
     u(ij)=up-2.0*ubar*unxr(ij)
     v(ij)=vp-2.0*ubar*unyr(ij)
     t(ij)=tp
   endif
!
! no-slip boundary condition
!
   if (wtlb(j).eq.2) then
     p(ij)=pp
     u(ij)=-up
     v(ij)=-vp
     t(ij)=tp
   endif
!
! driven constant velocity lead boundary (cavity).
!
   if (wtlb(j).eq.3) then
     p(ij)=pp
     u(ij)=2.0*ulb(j)-up
     v(ij)=2.0*vlb(j)-vp
     t(ij)=tp
   endif
!
! subsonic in-flow boundary condition, pt, tt, theta.
!
   if (wtlb(j).eq.4) then
     cwsq=gamma*rgas*tp
     cw=ssqrt(cwsq)
     mwsq=(up*up+vp*vp)/cwsq
     mw=ssqrt(mwsq+em20)
     vel=cw*mw
     factor=1.0+gm1/2.0*mwsq
     p(ij)=ptlb(j)/factor**ggm1
     u(ij)=vel*uxlb(j)
     v(ij)=vel*uylb(j)
     t(ij)=ttlb(j)/factor
   endif
!
! subsonic inflow boundary condition, ts, u, v.
!
   if (wtlb(j).eq.5) then
     p(ij)=pp
     u(ij)=ulb(j)
     v(ij)=vlb(j)
     t(ij)=tslb(j)
   endif
!
```

```
! subsonic inflow boundary condition, ps, u, v.
!
    if (wtlb(j).eq.6) then
      p(ij)=pslb(j)
      u(ij)=ulb(j)
      v(ij)=vlb(j)
      t(ij)=tp
    endif
!
! supersonic in-flow boundary condition
!
    if (wtlb(j).eq.10) then
      p(ij)=pslb(j)
      u(ij)=ulb(j)
      v(ij)=vlb(j)
      t(ij)=tslb(j)
    endif
!
! out-flow boundary condition
!
    if (wtlb(j).eq.11) then
      ipj=ij+1
      p(ij)=pp
      if (mach(ipj).lt.1.0) p(ij)=pslb(j)
      u(ij)=up
      v(ij)=vp
      t(ij)=tp
    endif
!
! mass injection boundary condition, isentropic
!
    if (wtlb(j).eq.15) then
      const=pp/rp**gamma
      t(ij)=2.0*tslb(j)-tp
      dens=(rgas*t(ij)/const)**(1.0/gm1)
      p(ij)=rgas*dens*t(ij)
      vw=rplb(j)*brlb(j)/dens
      u(ij)=unxr(ij)*vw
      v(ij)=unyr(ij)*vw
      u(ij)=2.0*u(ij)-up
      v(ij)=2.0*v(ij)-vp
    endif
!
! mass injection boundary condition, non-isentropic
!
    if (wtlb(j).eq.16) then
      t(ij)=2.0*tslb(j)-tp
      p(ij)=pp
      dens=p(ij)/rgas/t(ij)
      vw=rplb(j)*brlb(j)/dens
      u(ij)=unxr(ij)*vw
      v(ij)=unyr(ij)*vw
      u(ij)=2.0*u(ij)-up
      v(ij)=2.0*v(ij)-vp
    endif

    q1(ij)=p(ij)/rgas/t(ij)
    q2(ij)=q1(ij)*u(ij)
    q3(ij)=q1(ij)*v(ij)
    velsq=u(ij)*u(ij)+v(ij)*v(ij)
    q4(ij)=p(ij)/gm1+0.5*q1(ij)*velsq
  enddo
```

```
!
! right wall
!
 i=im1

 do j=2,jm1
   ij=ind(i,j)

   if (wtrb(j).ne.10) call calqpm ('r',inorxi)

   imj=ij
   ij=imj+1
!
! free-slip boundary condition
!
   if (wtrb(j).le.1) then
     p(ij)=pm
     ubar=um*unxr(imj)+vm*unyr(imj)
     u(ij)=um-2.0*ubar*unxr(imj)
     v(ij)=vm-2.0*ubar*unyr(imj)
     t(ij)=tm
   endif
!
! no-slip boundary condition
!
   if (wtrb(j).eq.2) then
     p(ij)=pm
     u(ij)=-um
     v(ij)=-vm
     t(ij)=tm
   endif
!
! driven constant velocity lead boundary (cavity).
!
   if (wtrb(j).eq.3) then
     p(ij)=pm
     u(ij)=2.0*urb(j)-um
     v(ij)=2.0*vrb(j)-vm
     t(ij)=tm
   endif
!
! subsonic in-flow boundary condition, pt, tt, theta.
!
   if (wtrb(j).eq.4) then
     cwsq=gamma*rgas*tm
     cw=ssqrt(cwsq)
     mwsq=(um*um+vm*vm)/cwsq
     mw=ssqrt(mwsq+em20)
     vel=cw*mw
     factor=1.0+gm1/2.0*mwsq
     p(ij)=ptrb(j)/factor**ggm1
     u(ij)=vel*uxrb(j)
     v(ij)=vel*uyrb(j)
     t(ij)=ttrb(j)/factor
   endif
!
! subsonic inflow boundary condition, ts, u, v.
!
   if (wtrb(j).eq.5) then
     p(ij)=pm
     u(ij)=urb(j)
     v(ij)=vrb(j)
```

```
      t(ij)=tsrb(j)
    endif
!
! subsonic inflow boundary condition, ps, u, v.
!
    if (wtrb(j).eq.6) then
      p(ij)=psrb(j)
      u(ij)=urb(j)
      v(ij)=vrb(j)
      t(ij)=tm
    endif
!
! supersonic in-flow boundary condition
!
    if (wtrb(j).eq.10) then
      p(ij)=psrb(j)
      u(ij)=urb(j)
      v(ij)=vrb(j)
      t(ij)=tsrb(j)
    endif
!
! out-flow boundary condition
!
    if (wtrb(j).eq.11) then
      p(ij)=pm
      if (mach(imj).lt.1.0) p(ij)=psrb(j)
      u(ij)=um
      v(ij)=vm
      t(ij)=tm
    endif
!
! mass injection boundary condition, isentropic
!
    if (wtrb(j).eq.15) then
      const=pm/rm**gamma
      t(ij)=2.0*tsrb(j)-tm
      dens=(rgas*t(ij)/const)**(1.0/gm1)
      p(ij)=rgas*dens*t(ij)
      vw=rprb(j)*brrb(j)/dens
      u(ij)=-unxr(imj)*vw
      v(ij)=-unyr(imj)*vw
      u(ij)=2.0*u(ij)-um
      v(ij)=2.0*v(ij)-vm
    endif
!
! mass injection boundary condition, non-isentropic
!
    if (wtrb(j).eq.16) then
      t(ij)=2.0*tsrb(j)-tm
      p(ij)=pm
      dens=p(ij)/rgas/t(ij)
      vw=rprb(j)*brrb(j)/dens
      u(ij)=-unxr(imj)*vw
      v(ij)=-unyr(imj)*vw
      u(ij)=2.0*u(ij)-um
      v(ij)=2.0*v(ij)-vm
    endif

    q1(ij)=p(ij)/rgas/t(ij)
    q2(ij)=q1(ij)*u(ij)
    q3(ij)=q1(ij)*v(ij)
    velsq=u(ij)*u(ij)+v(ij)*v(ij)
```

```
    q4(ij)=p(ij)/gm1+0.5*q1(ij)*velsq
  enddo
!
! bottom wall
!
  j=1

  do i=2,im1
    ij=ind(i,j)

    if (wtbb(i).ne.10) call calqpm ('t',inoret)
!
! free-slip boundary condition
!
    if (wtbb(i).le.1) then
      p(ij)=pp
      vbar=up*unxt(ij)+vp*unyt(ij)
      u(ij)=up-2.0*vbar*unxt(ij)
      v(ij)=vp-2.0*vbar*unyt(ij)
      t(ij)=tp
    endif
!
! no-slip boundary condition
!
    if (wtbb(i).eq.2) then
      p(ij)=pp
      u(ij)=-up
      v(ij)=-vp
      t(ij)=tp
    endif
!
! driven constant velocity lead boundary (cavity).
!
    if (wtbb(i).eq.3) then
      p(ij)=pp
      u(ij)=2.0*ubb(i)-up
      v(ij)=2.0*vbb(i)-vp
      t(ij)=tp
    endif
!
! subsonic in-flow boundary condition, pt, tt, theta.
!
    if (wtbb(i).eq.4) then
      cwsq=gamma*rgas*tp
      cw=ssqrt(cwsq)
      mwsq=(up*up+vp*vp)/cwsq
      mw=ssqrt(mwsq+em20)
      vel=cw*mw
      factor=1.0+gm1/2.0*mwsq
      p(ij)=ptbb(i)/factor**ggm1
      u(ij)=vel*uxbb(i)
      v(ij)=vel*uybb(i)
      t(ij)=ttbb(i)/factor
    endif
!
! subsonic inflow boundary condition, ts, u, v.
!
    if (wtbb(i).eq.5) then
      p(ij)=pp
      u(ij)=ubb(i)
      v(ij)=vbb(i)
      t(ij)=tbb(i)
```

```
      endif
!
! subsonic inflow boundary condition, ps, u, v.
!
      if (wtbb(i).eq.6) then
        p(ij)=psbb(i)
        u(ij)=ubb(i)
        v(ij)=vbb(i)
        t(ij)=tp
      endif
!
! supersonic in-flow boundary condition
!
      if (wtbb(i).eq.10) then
        u(ij)=ubb(i)
        v(ij)=vbb(i)
        p(ij)=psbb(i)
        t(ij)=tsbb(i)
      endif
!
! out-flow boundary condition
!
      if (wtbb(i).eq.11) then
        ijp=i+imax
        p(ij)=pp
        if (mach(ijp).lt.1.0) p(ij)=psbb(i)
        u(ij)=up
        v(ij)=vp
        t(ij)=tp
      endif
!
! mass injection boundary condition, isentropic
!
      if (wtbb(i).eq.15) then
        const=pp/rp**gamma
        t(ij)=2.0*tsbb(i)-tp
        dens=(rgas*t(ij)/const)**(1.0/gm1)
        p(ij)=rgas*dens*t(ij)
        vw=rpbb(i)*brbb(i)/dens
        u(ij)=unxt(ij)*vw
        v(ij)=unyt(ij)*vw
        u(ij)=2.0*u(ij)-up
        v(ij)=2.0*v(ij)-vp
      endif
!
! mass injection boundary condition, non-isentropic
!
      if (wtbb(i).eq.16) then
        t(ij)=2.0*tsbb(i)-tp
        p(ij)=pp
        dens=p(ij)/rgas/t(ij)
        vw=rpbb(i)*brbb(i)/dens
        u(ij)=unxt(ij)*vw
        v(ij)=unyt(ij)*vw
        u(ij)=2.0*u(ij)-up
        v(ij)=2.0*v(ij)-vp
      endif

      q1(ij)=p(ij)/rgas/t(ij)
      q2(ij)=q1(ij)*u(ij)
      q3(ij)=q1(ij)*v(ij)
      velsq=u(ij)*u(ij)+v(ij)*v(ij)
```

```
    q4(ij)=p(ij)/gm1+0.5*q1(ij)*velsq
  enddo
!
! top wall
!
  j=jm1

  do i=2,im1
    ij=ind(i,j)

    if (wttb(i).ne.10) call calqpm ('t',inoret)

    ijm=ij
    ij=ijm+imax
!
! free-slip boundary condition
!
    if (wttb(i).le.1) then
      p(ij)=pm
      vbar=um*unxt(ijm)+vm*unyt(ijm)
      u(ij)=um-2.0*vbar*unxt(ijm)
      v(ij)=vm-2.0*vbar*unyt(ijm)
      t(ij)=tm
    endif
!
! no-slip boundary condition
!
    if (wttb(i).eq.2) then
      p(ij)=pm
      u(ij)=-um
      v(ij)=-vm
      t(ij)=tm
    endif
!
! driven constant velocity lead boundary (cavity).
!
    if (wttb(i).eq.3) then
      p(ij)=pm
      u(ij)=2.0*utb(i)-um
      v(ij)=2.0*vtb(i)-vm
      t(ij)=tm
    endif
!
! subsonic in-flow boundary condition, pt, tt, theta.
!
    if (wttb(i).eq.4) then
      cwsq=gamma*rgas*tm
      cw=ssqrt(cwsq)
      mwsq=(um*um+vm*vm)/cwsq
      mw=ssqrt(mwsq+em20)
      vel=cw*mw
      factor=1.0+gm1/2.0*mwsq
      p(ij)=pttb(i)/factor**ggm1
      u(ij)=vel*uxtb(i)
      v(ij)=vel*uytb(i)
      t(ij)=tttb(i)/factor
    endif
!
! subsonic inflow boundary condition, ts, u, v.
!
    if (wttb(i).eq.5) then
      p(ij)=pm
```

```
      u(ij)=utb(i)
      v(ij)=vtb(i)
      t(ij)=tstb(i)
    endif
!
! subsonic inflow boundary condition, ps, u, v.
!
    if (wttb(i).eq.6) then
      p(ij)=pstb(i)
      u(ij)=utb(i)
      v(ij)=vtb(i)
      t(ij)=tm
    endif
!
! supersonic in-flow boundary condition
!
    if (wttb(i).eq.10) then
      p(ij)=pstb(i)
      u(ij)=utb(i)
      v(ij)=vtb(i)
      t(ij)=tstb(i)
    endif
!
! out-flow boundary condition
!
    if (wttb(i).eq.11) then
      p(ij)=pm
      if (mach(ijm).lt.1.0) p(ij)=pstb(i)
      u(ij)=um
      v(ij)=vm
      t(ij)=tm
    endif
!
! mass injection boundary condition, isentropic
!
    if (wttb(i).eq.15) then
      const=pm/rm**gamma
      t(ij)=2.0*tstb(i)-tm
      dens=(rgas*t(ij)/const)**(1.0/gm1)
      p(ij)=rgas*dens*t(ij)
      vw=rptb(i)*brtb(i)/dens
      u(ij)=-unxt(ijm)*vw
      v(ij)=-unyt(ijm)*vw
      u(ij)=2.0*u(ij)-um
      v(ij)=2.0*v(ij)-vm
    endif
!
! mass injection boundary condition, non-isentropic
!
    if (wttb(i).eq.16) then
      t(ij)=2.0*tstb(i)-tm
      p(ij)=pm
      dens=p(ij)/rgas/t(ij)
      vw=rptb(i)*brtb(i)/dens
      u(ij)=-unxt(ijm)*vw
      v(ij)=-unyt(ijm)*vw
      u(ij)=2.0*u(ij)-um
      v(ij)=2.0*v(ij)-vm
    endif

    q1(ij)=p(ij)/rgas/t(ij)
    q2(ij)=q1(ij)*u(ij)
```

```
    q3(ij)=q1(ij)*v(ij)
    velsq=u(ij)*u(ij)+v(ij)*v(ij)
    q4(ij)=p(ij)/gm1+0.5*q1(ij)*velsq
  enddo
!
! user special boundary condition
!
  return
  end

  subroutine wrtplt
!
!******************************************************************************
! write a special plot file to allow the use of post processing.
!******************************************************************************
!
  include 'params.cmd'
  include 'precs.cmd'
  include 'arrays.cmd'
  include 'param.cmd'
  include 'bcon.cmd'

  real*8 den(ijdim), ptmp(ijdim), ttmp(ijdim), utmp(ijdim), vtmp(ijdim), &
        mtmp(ijdim), sx(ijdim), sy(ijdim), v1(ijdim), v2(ijdim), v3(ijdim), &
        v4(ijdim)

  include 'aindex.cmd'

  if (cycle.le.fcycle) then
    do j=1,jm1
    do i=1,im1
      ij=ind(i,j)
      sx(ij)=x(ij)
      sy(ij)=y(ij)
    enddo
    enddo
  endif

  do j=2,jmax
  do i=2,imax
    ij=ind(i,j)
    imj=ij-1
    ijm=ij-imax
    imjm=ij-1-imax
    rvij=1.0/jac(ij)
    rvimj=1.0/jac(imj)
    rvijm=1.0/jac(ijm)
    rvimjm=1.0/jac(imjm)
    rvsum=rvij+rvimj+rvijm+rvimjm
    q1node=(rvij*q1(ij)+rvimj*q1(imj)+rvijm*q1(ijm)+rvimjm*q1(imjm))/rvsum
    q2node=(rvij*q2(ij)+rvimj*q2(imj)+rvijm*q2(ijm)+rvimjm*q2(imjm))/rvsum
    q3node=(rvij*q3(ij)+rvimj*q3(imj)+rvijm*q3(ijm)+rvimjm*q3(imjm))/rvsum
    q4node=(rvij*q4(ij)+rvimj*q4(imj)+rvijm*q4(ijm)+rvimjm*q4(imjm))/rvsum

    v1(imjm)=q1node
    v2(imjm)=q2node
    v3(imjm)=q3node
    v4(imjm)=q4node
  enddo
  enddo

  do j=1,jm1
```

```
      else
        write(lusav) q1(ij),q2(ij),q3(ij),q4(ij)
      endif
    enddo
    enddo

    call ocfile (lusav,fnsav,dfsav,flfm,0)

    if (iprint.eq.0) write(luscn,'(a,1pe12.5)') ' restart data at time= ',ttime
    if (iprint.eq.1 .or. iprint.eq.3) &
      write(luscn,'(a,1pe12.5)') ' restart data at time= ',ttime
    if (iprint.eq.2 .or. iprint.eq.3) &
      write(luout,'(a,1pe12.5)') ' restart data at time= ',ttime

    return
    end

    subroutine wallsh
!
!******************************************************************************
! this subroutine calculates the extra source terms due to presence of shear
! force at boundaries.
!******************************************************************************
!
    include 'params.cmd'
    include 'precs.cmd'
    include 'arrays.cmd'
    include 'param.cmd'
    include 'bcon.cmd'

    character*30 fnwal, dfwal
    integer luwal

    data fnwal, dfwal, luwal /'wallsh.out',' ',99/

    include 'aindex.cmd'

    indx(ki,kj)=ki+(kj-1)*4+(ij-1)*16

    call ocfile (luwal,fnwal,dfwal,'formatted  ',1)
!
! left wall
!
   id=0
   i=2

   do j=2,jm1
     if (wtlb(j).eq.1) then
       ij=ind(i,j)
       imj=ij-1

       if (arer(imj).gt.em20) then
         imjm=imj-imax
         ijm=ij-imax
         delx=x(imj)-x(imjm)
         dely=y(imj)-y(imjm)
         cons=-x(imj)*y(imjm)+x(imjm)*y(imj)
         xcen=(x(ij)+x(imj)+x(imjm)+x(ijm))/4.0
         ycen=(y(ij)+y(imj)+y(imjm)+y(ijm))/4.0
         dist=(ycen*delx-xcen*dely+cons)/ssqrt(delx*delx+dely*dely)
         dist=sabs(dist)
         uw=(u(ij)+u(imj))/2.0
```

```
        vw=(v(ij)+v(imj))/2.0
        vwall=ssqrt(uw*uw+vw*vw)
        creno=dist*vwall*q1(ij)/mulm(ij)
        factor=1.0
        if (creno.gt.130.3) factor=creno/(0.75+2.19*slog(creno))**2
        const=factor*mulm(ij)/dist
        shearx=-uw*const
        sheary=-vw*const
        omg2(ij)=omg2(ij)+arer(imj)*shearx
        omg3(ij)=omg3(ij)+arer(imj)*sheary

        if (iprint.eq.10 .and. id.eq.0) write(luwal,'(/a,//,3x,a,7x,a, &
          10x,a,10x,a,10x,a,/,1x)') ' variables along the left-wall are', &
          'node','h/cp','p','t','mach'

        if (iprint.eq.10) write(luwal,'(4x,i2,4x,1p4e12.4)') &
          j,1.125*const,p(ij),t(ij),mach(ij)

        id=1

        if (implct.ge.1) then
          const=-const*arer(imj)/q1(ij)/2.0
          duwdq1=-u(ij)
          duwdq2=1.0
          dvwdq1=-v(ij)
          dvwdq3=1.0
          dhdq(indx(2,1))=dhdq(indx(2,1))+const*duwdq1
          dhdq(indx(2,2))=dhdq(indx(2,2))+const*duwdq2
          dhdq(indx(3,1))=dhdq(indx(3,1))+const*dvwdq1
          dhdq(indx(3,3))=dhdq(indx(3,3))+const*dvwdq3
        endif
      endif
    endif
  enddo
!
! right wall
!
  id=0
  i=im1

  do j=2,jm1
    if (wtrb(j).eq.1) then
      ij=ind(i,j)

      if (arer(ij).ge.em20) then
        imj=ij-1
        imjm=imj-imax
        ijm=ij-imax
        ipj=ij+1
        delx=x(ij)-x(ijm)
        dely=y(ij)-y(ijm)
        cons=-x(ij)*y(ijm)+x(ijm)*y(ij)
        xcen=(x(ij)+x(imj)+x(imjm)+x(ijm))/4.0
        ycen=(y(ij)+y(imj)+y(imjm)+y(ijm))/4.0
        dist=(ycen*delx-xcen*dely+cons)/ssqrt(delx*delx+dely*dely)
        dist=sabs(dist)
        uw=(u(ij)+u(ipj))/2.0
        vw=(v(ij)+v(ipj))/2.0
        vwall=ssqrt(uw*uw+vw*vw)
        creno=dist*vwall*q1(ij)/mulm(ij)
        factor=1.0
        if (creno.gt.130.3) factor=creno/(0.75+2.19*slog(creno))**2
```

```
      10x,a,10x,a,10x,a,/)') ' variables along the bottom-wall are', &
      'node','h/cp','p','t','mach'

    if (iprint.eq.10) write(luwal,'(4x,i2,4x,1p4e12.4)') &
      j,1.125*const,p(ij),t(ij),mach(ij)

    id=1

    if (implct.ge.1) then
      const=-const*aret(ijm)/q1(ij)/2.0
      duwdq1=-u(ij)
      duwdq2=1.0
      dvwdq1=-v(ij)
      dvwdq3=1.0
      dhdq(indx(2,1))=dhdq(indx(2,1))+const*duwdq1
      dhdq(indx(2,2))=dhdq(indx(2,2))+const*duwdq2
      dhdq(indx(3,1))=dhdq(indx(3,1))+const*dvwdq1
      dhdq(indx(3,3))=dhdq(indx(3,3))+const*dvwdq3
    endif
   endif
  endif
 enddo
!
! top wall
!
 id=0
 j=jm1

 do i=2,im1
   if (wttb(i).eq.1) then
     ij=ind(i,j)

     if (aret(ij).ge.em20) then
       imj=ij-1
       imjm=imj-imax
       ijm=ij-imax
       ijp=ij+imax
       delx=x(ij)-x(imj)
       dely=y(ij)-y(imj)
       cons=-x(ij)*y(imj)+x(imj)*y(ij)
       xcen=(x(ij)+x(imj)+x(imjm)+x(ijm))/4.0
       ycen=(y(ij)+y(imj)+y(imjm)+y(ijm))/4.0
       dist=(ycen*delx-xcen*dely+cons)/ssqrt(delx*delx+dely*dely)
       dist=sabs(dist)
       uw=(u(ij)+u(ijp))/2.0
       vw=(v(ij)+v(ijp))/2.0
       vwall=ssqrt(uw*uw+vw*vw)
       creno=dist*vwall*q1(ij)/mulm(ij)
       factor=1.0
       if (creno.gt.130.3) factor=creno/(0.75+2.19*slog(creno))**2
       const=factor*mulm(ij)/dist
       shearx=-uw*const
       sheary=-vw*const
       omg2(ij)=omg2(ij)+aret(ij)*shearx
       omg3(ij)=omg3(ij)+aret(ij)*sheary

       if (iprint.eq.10 .and. id.eq.0) write(luwal,'(/,a,//,3x,a,7x,a, &
         10x,a,10x,a,10x,a,/)') ' variables along the top-wall are', &
         'node','h/cp','p','t','mach'

       if (iprint.eq.10) write(luwal,'(4x,i2,4x,1p4e12.4)') &
         j,1.125*const,p(ij),t(ij),mach(ij)
```

```
        id=1

        if (implct.ge.1) then
          const=-const*aret(ij)/q1(ij)/2.0
          duwdq1=-u(ij)
          duwdq2=1.0
          dvwdq1=-v(ij)
          dvwdq3=1.0
          dhdq(indx(2,1))=dhdq(indx(2,1))+const*duwdq1
          dhdq(indx(2,2))=dhdq(indx(2,2))+const*duwdq2
          dhdq(indx(3,1))=dhdq(indx(3,1))+const*dvwdq1
          dhdq(indx(3,3))=dhdq(indx(3,3))+const*dvwdq3
        endif
      endif
    endif
  enddo

  call ocfile (luwal,fnwal,dfwal,'formatted  ',0)
!
! user special wall shear
!
  return
  end

!
!*****************************************************************************
! viscon calculates laminar and turbulent viscosity and thermal diffusivity.
!*****************************************************************************
!
  subroutine viscl
!
!*****************************************************************************
! calculate the laminar viscosity based on sutherland's law of viscosity. this
! formula is
!
! mu = sc1*(temp**1.5)/(temp+sc2)
!*****************************************************************************
!
  include 'params.cmd'
  include 'precs.cmd'
  include 'arrays.cmd'
  include 'param.cmd'
  include 'bcon.cmd'

  inclde 'aindex.cmd'

  do j=1,jmax
  do i=1,imax
    ij=ind(i,j)
    mulm(ij)=sc1*(t(ij)**1.5)/(t(ij)+sc2)
    if (iturb.eq.0) mu(ij)=mulm(ij)
  enddo
  enddo

  return
  end
!-----------------------------------------------------------------------------
  subroutine visct
!
!*****************************************************************************
! calculate the turbulent viscosity based on simple algebraic subgrid scale
```

```
! tubulent viscosity model of deardorff.
!*******************************************************************************
!
 include 'params.cmd'
 include 'precs.cmd'
 include 'arrays.cmd'
 include 'param.cmd'
 include 'bcon.cmd'

 real*8 jacc

 data ckd/0.02044/

 include 'aindex.cmd'

 do j=2,jm1
 do i=2,im1
   ij=ind(i,j)

   if (cyl.gt.0.0) then
     call geom (2,0,unx,uny,area,volt1,volt2)

     jacc=volt1+volt2
   else
     jacc=jac(ij)
   endif

   vc=v(ij)
   imj=ij-1
   ijm=ij-imax
   imjm=ijm-1
   ipj=ij+1
   ijp=ij+imax
   xr=(x(ij)+x(ijm))/2.0
   xl=(x(imj)+x(imjm))/2.0
   xt=(x(ij)+x(imj))/2.0
   xb=(x(ijm)+x(imjm))/2.0
   yr=(y(ij)+y(ijm))/2.0
   yl=(y(imj)+y(imjm))/2.0
   yt=(y(ij)+y(imj))/2.0
   yb=(y(ijm)+y(imjm))/2.0
   xixc=(yt-yb)/jacc
   xiyc=-(xt-xb)/jacc
   etxc=-(yr-yl)/jacc
   etyc=(xr-xl)/jacc
   delx1=xr-xl
   delx2=xt-xb
   dely1=yr-yl
   dely2=yt-yb
   side1=ssqrt(delx1*delx1+dely1*dely1)
   side2=ssqrt(delx2*delx2+dely2*dely2)
   sidem=smax1(side1,side2)
   ur=u(ipj)
   vr=v(ipj)
   ul=u(imj)
   vl=v(imj)
   ut=u(ijp)
   vt=v(ijp)
   ub=u(ijm)
   vb=v(ijm)
   dudx=(ur-ul)/2.0
   dude=(ut-ub)/2.0
```

```
    dvdx=(vr-vl)/2.0
    dvde=(vt-vb)/2.0
    d11=2.0*(xixc*dudx+etxc*dude)
    d22=2.0*(xiyc*dvdx+etyc*dvde)
    d33=2.0*cyl*vc/yc(ij)
    d12=xiyc*dudx+etyc*dude+xixc*dvdx+etxc*dvde
    mutb=ckd*q1(ij)*sidem*sidem*ssqrt(d11*d11+d22*d22+d33*d33+2.0*d12*d12)
    mu(ij)=mulm(ij)+mutb
  enddo
  enddo

  i1=1
  im=imax

  do j=2,jm1
    i1j=ind(i1,j)
    imj=ind(im,j)
    mu(i1j)=mu(i1j+1)
    mu(imj)=mu(imj-1)
  enddo

  j1=1
  jm=jmax

  do i=2,im1
    ij1=ind(i,j1)
    ijm=ind(i,jm)
    mu(ij1)=mu(ij1+imax)
    mu(ijm)=mu(ijm-imax)
  enddo

  mu(ind(1,1))=mu(ind(2,2))
  mu(ind(1,jmax))=mu(ind(2,jm1))
  mu(ind(imax,1))=mu(ind(im1,2))
  mu(ind(imax,jmax))=mu(ind(im1,jm1))

  return
  end
!------------------------------------------------------------------------------
  subroutine tdifl
!*****************************************************************************
! calculate the laminar thermal conductivity.
!*****************************************************************************
!
  include 'params.cmd'
  include 'precs.cmd'
  include 'arrays.cmd'
  include 'param.cmd'
  include 'bcon.cmd'

  include 'aindex.cmd'

  return
  end
!------------------------------------------------------------------------------
  subroutine tdift
!
!*****************************************************************************
! this subroutine calculates the turbulent thermal conductivity.
!*****************************************************************************
!
  include 'params.cmd'
```

```
 include 'precs.cmd'
 include 'arrays.cmd'
 include 'param.cmd'
 include 'bcon.cmd'

 include 'aindex.cmd'

 do j=1,jmax
 do i=1,imax
   ij=ind(i,j)
   mutb=mu(ij)-mulm(ij)
   ku(ij)=kulm(ij)+mutb*cpg/prt
 enddo
 enddo

 return
 end

 subroutine solve
!
!*****************************************************************************
! solve the governing equations using an approximate factorization technique if
! implct=1. this splitting technique reduces the two-dimensional solution to
! the solution of two one-dimensional problems. two sweeps are carried out. one
! in xi-direction and one in eta-direction.
!*****************************************************************************
!
 include 'params.cmd'
 include 'precs.cmd'
 include 'arrays.cmd'
 include 'param.cmd'
 include 'bcon.cmd'

 dimension cij(16), dij(4), ipvt(4)

 include 'aindex.cmd'

 isweep=0
 terr(1)=0.0
 terr(2)=0.0
 terr(3)=0.0
 terr(4)=0.0

 do j=2,jm1
 do i=2,im1
   ij=ind(i,j)
   terr(1)=terr(1)+sabs(omg1(ij))
   terr(2)=terr(2)+sabs(omg2(ij))
   terr(3)=terr(3)+sabs(omg3(ij))
   terr(4)=terr(4)+sabs(omg4(ij))
 enddo
 enddo

 if (implct.gt.0) go to 1
!
! explicit solution
!
 do j=2,jm1
 do i=2,im1
   ij=ind(i,j)
   rdtja=jac(ij)/dt(ij)
   dq1(ij)=omg1(ij)/rdtja
```

```
    dq2(ij)=omg2(ij)/rdtja
    dq3(ij)=omg3(ij)/rdtja
    dq4(ij)=omg4(ij)/rdtja
  enddo
  enddo

  go to 3

1 continue
!
! implicit solution:
!    isplit=1 is the adi two-factor split or line gauss-seidel
!    isplit=2 is the two-factor eigenvalue split
!    isplit>2 is the point iterative sor scheme
!
  if (isplit.gt.1) go to 2
!
! implicit adi two-factor split scheme or line gauss-seidel
!
  isweep=isweep+1
  errl(1)=0.0
  errl(2)=0.0
  errl(3)=0.0
  errl(4)=0.0

  do j=2,jm1
!
! form the coefficient matrices and solve the block-tridiaginal in xi-direction
!
    call abcdxi (2,im1)

    if (impbc.gt.0) call bcimp

    call btrid (aa,bb,cc,dd,ilow,ihgh,4)

    do i=ilow,ihgh
      ij=ind(i,j)
      indx=(i-1)*4
      dq1(ij)=dd(indx+1)
      dq2(ij)=dd(indx+2)
      dq3(ij)=dd(indx+3)
      dq4(ij)=dd(indx+4)
    enddo
  enddo

  do i=2,im1
!
! form the coefficient matrices and solve the block-tridiaginal in et-direction
!
    call abcdet (2,jm1)

    if (impbc.gt.0) call bcimp

    call btrid (aa,bb,cc,dd,jlow,jhgh,4)

    do j=jlow,jhgh
      ij=ind(i,j)
      indx=(j-1)*4
      dq1(ij)=dd(indx+1)
      dq2(ij)=dd(indx+2)
      dq3(ij)=dd(indx+3)
      dq4(ij)=dd(indx+4)
```

```
   enddo
  enddo

  if (lgs.gt.0) write(luscn,'(a,i2,1p4e12.3)') &
    ' LGS errors at sweep ',isweep,errl(1),errl(2),errl(3),errl(4)

  if (errl(1) .gt. epsig*terr(1) .and. errl(2) .gt. epsig*terr(2) .and. &
     errl(3) .gt. epsig*terr(3) .and. errl(4) .gt. epsig*terr(4) .and. &
     isweep.lt.lgs) go to 1

  go to 3

2 if (isplit.le.2) then
!
! implicit solution using the two-factor eigenvalue split
!
    icycle=0
    if (iflip.eq.1) icycle=1
    if (iflip.gt.1) icycle=mod(cycle,2)

    if (icycle.eq.0) then
      incr=1
      ilow=2
      ihgh=im1
      jlow=2
      jhgh=jm1
    else
      incr=-1
      ilow=im1
      ihgh=2
      jlow=jm1
      jhgh=2
    endif

    do j=jlow,jhgh,incr
    do i=ilow,ihgh,incr
!
! forward sweep on even cycle and backward sweep on odd cycle
!
      ij=ind(i,j)

      if (icycle.eq.0) call lomat (cij,dij,icycle)
      if (icycle.ne.0) call upmat (cij,dij,icycle)
      call ludcmp (cij,4,4,ipvt,flag)

      if (flag.eq.0) then
        write(luscn,'(a,2i4)') ' near singular matrix at point ij = ',i,j

        stop
      endif

      call lubksb (cij,4,4,ipvt,dij)

      dq1(ij)=dij(1)
      dq2(ij)=dij(2)
      dq3(ij)=dij(3)
      dq4(ij)=dij(4)
    enddo
    enddo

    do j=jlow,jhgh,incr
    do i=ilow,ihgh,incr
```

```
!
! diagonal inversion
!
      ij=ind(i,j)

      call dmat
    enddo
    enddo

    do j=jhgh,jlow,-incr
    do i=ihgh,ilow,-incr
!
! backward sweep on even cycle and forward sweep on odd cycle
!
      ij=ind(i,j)

      if (icycle.eq.0) call upmat (cij,dij,icycle)
      if (icycle.ne.0) call lomat (cij,dij,icycle)
      call ludcmp (cij,4,4,ipvt,flag)

      if (flag.eq.0) then
        write(luscn,'(a,2i4)') ' near singular matrix at point ij = ',i,j

        stop
      endif

      call lubksb (cij,4,4,ipvt,dij)

      dq1(ij)=dij(1)
      dq2(ij)=dij(2)
      dq3(ij)=dij(3)
      dq4(ij)=dij(4)
    enddo
    enddo
  endif
!
! point iterative scheme
!
 do iter=1,isplit-2
 do j=2,jm1
 do i=2,im1
   ij=ind(i,j)

   call pmat (cij,dij)
   call ludcmp (cij,4,4,ipvt,flag)

   if (flag.eq.0) then
     write(luscn,'(a,2i4)') ' near singular matrix at point ij = ',i,j

     stop
   endif

   call lubksb (cij,4,4,ipvt,dij)

   dq1(ij)=dij(1)
   dq2(ij)=dij(2)
   dq3(ij)=dij(3)
   dq4(ij)=dij(4)
 enddo
 enddo
 enddo
```

```
 do ki=1,4
   kikj=ki+(kj-1)*4
   ind=kikj+(ij-1)*16
   indxm=ind-16
   indym=ind-imax*16
   unity=0.0
   if (ki.eq.kj) unity=rdtja
   cij(kikj)=unity+ap(ind)+bp(ind)-am(indxm)-bm(indym)-dhdq(ind)
 enddo
 enddo

 call multmv (cij,dij,dij,4,4)

 dq1(ij)=dij(1)/rdtja
 dq2(ij)=dij(2)/rdtja
 dq3(ij)=dij(3)/rdtja
 dq4(ij)=dij(4)/rdtja

 return
 end
!-----------------------------------------------------------------------------
 subroutine lomat (cij,dij,icycle)
!
!******************************************************************************
! form the matrices cij, dij for each cell (i,j) for lower triangular forward
! sweep.
!******************************************************************************
!
 include 'params.cmd'
 include 'precs.cmd'
 include 'arrays.cmd'
 include 'param.cmd'
 include 'bcon.cmd'

 real*8 cij, dij

 dimension cij(16), cimj(16), cijm(16), dij(4), dq(4)

 imj=ij-1
 ijm=ij-imax
 rdtja=jac(ij)/dt(ij)

 if (icycle.eq.0) then
   dij(1)=omg1(ij)
   dij(2)=omg2(ij)
   dij(3)=omg3(ij)
   dij(4)=omg4(ij)
 else
   dij(1)=dq1(ij)*rdtja
   dij(2)=dq2(ij)*rdtja
   dij(3)=dq3(ij)*rdtja
   dij(4)=dq4(ij)*rdtja
 endif

 do kj=1,4
 do ki=1,4
   kikj=ki+(kj-1)*4
   ind=kikj+(ij-1)*16
   indxm=ind-16
   indym=ind-imax*16
   unity=0.0
   if (ki.eq.kj) unity=rdtja
```

```
    cij(kikj)=unity+ap(ind)+bp(ind)-am(indxm)-bm(indym)-dhdq(ind)
    cimj(kikj)=-ap(indxm)
    cijm(kikj)=-bp(indym)
  enddo
  enddo

  dq(1)=dq1(imj)
  dq(2)=dq2(imj)
  dq(3)=dq3(imj)
  dq(4)=dq4(imj)

  call multmv (cimj,dq,dq,4,4)

  do ki=1,4
    dij(ki)=dij(ki)-dq(ki)
  enddo

  dq(1)=dq1(ijm)
  dq(2)=dq2(ijm)
  dq(3)=dq3(ijm)
  dq(4)=dq4(ijm)

  call multmv (cijm,dq,dq,4,4)

  do ki=1,4
    dij(ki)=dij(ki)-dq(ki)
  enddo

  return
  end
!-----------------------------------------------------------------------------
  subroutine upmat (cij,dij,icycle)
!
!*******************************************************************************
! form the matrices cij, dij for each cell (i,j) for uper triangular backward
! sweep.
!*******************************************************************************
!
  include 'params.cmd'
  include 'precs.cmd'
  include 'arrays.cmd'
  include 'param.cmd'
  include 'bcon.cmd'

  real*8 cij, dij

  dimension cij(16), cipj(16), cijp(16), dij(4), dq(4)

  ipj=ij+1
  ijp=ij+imax
  rdtja=jac(ij)/dt(ij)

  if (icycle.ne.0) then
    dij(1)=omg1(ij)
    dij(2)=omg2(ij)
    dij(3)=omg3(ij)
    dij(4)=omg4(ij)
  else
    dij(1)=dq1(ij)*rdtja
    dij(2)=dq2(ij)*rdtja
    dij(3)=dq3(ij)*rdtja
    dij(4)=dq4(ij)*rdtja
```

```
 endif

 do kj=1,4
 do ki=1,4
   kikj=ki+(kj-1)*4
   ind=kikj+(ij-1)*16
   indxm=ind-16
   indym=ind-imax*16
   unity=0.0
   if (ki.eq.kj) unity=rdtja
   cij(kikj)=unity+ap(ind)+bp(ind)-am(indxm)-bm(indym)-dhdq(ind)
   cipj(kikj)=am(ind)
   cijp(kikj)=bm(ind)
 enddo
 enddo

 dq(1)=dq1(ipj)
 dq(2)=dq2(ipj)
 dq(3)=dq3(ipj)
 dq(4)=dq4(ipj)

 call multmv (cipj,dq,dq,4,4)

 do ki=1,4
   dij(ki)=dij(ki)-dq(ki)
 enddo

 dq(1)=dq1(ijp)
 dq(2)=dq2(ijp)
 dq(3)=dq3(ijp)
 dq(4)=dq4(ijp)

 call multmv (cijp,dq,dq,4,4)

 do ki=1,4
   dij(ki)=dij(ki)-dq(ki)
 enddo

 return
 end
!-----------------------------------------------------------------------------
 subroutine pmat (cij,dij)
!
!*******************************************************************************
! form the matrices cij and dij for each cell (i,j) for point iterative sor.
!*******************************************************************************
!
 include 'params.cmd'
 include 'precs.cmd'
 include 'arrays.cmd'
 include 'param.cmd'
 include 'bcon.cmd'

 real*8 cij,dij

 dimension cij(16), cimj(16), cijm(16), dij(4), cipj(16), cijp(16), dq(4)

 imj=ij-1
 ipj=ij+1
 ijm=ij-imax
 ijp=ij+imax
 rdtja=jac(ij)/dt(ij)
```

```
dij(1)=omg1(ij)
dij(2)=omg2(ij)
dij(3)=omg3(ij)
dij(4)=omg4(ij)

do kj=1,4
do ki=1,4
  kikj=ki+(kj-1)*4
  ind=kikj+(ij-1)*16
  indxm=ind-16
  indym=ind-imax*16
  unity=0.0
  if (ki.eq.kj) unity=rdtja
  cij(kikj)=unity+ap(ind)+bp(ind)-am(indxm)-bm(indym)-dhdq(ind)
  cimj(kikj)=-ap(indxm)
  cipj(kikj)=am(ind)
  cijm(kikj)=-bp(indym)
  cijp(kikj)=bm(ind)
enddo
enddo

dq(1)=dq1(imj)
dq(2)=dq2(imj)
dq(3)=dq3(imj)
dq(4)=dq4(imj)

call multmv (cimj,dq,dq,4,4)

do ki=1,4
  dij(ki)=dij(ki)-dq(ki)
enddo

dq(1)=dq1(ijm)
dq(2)=dq2(ijm)
dq(3)=dq3(ijm)
dq(4)=dq4(ijm)

call multmv (cijm,dq,dq,4,4)

do ki=1,4
  dij(ki)=dij(ki)-dq(ki)
enddo

dq(1)=dq1(ipj)
dq(2)=dq2(ipj)
dq(3)=dq3(ipj)
dq(4)=dq4(ipj)

call multmv (cipj,dq,dq,4,4)

do ki=1,4
  dij(ki)=dij(ki)-dq(ki)
enddo

dq(1)=dq1(ijp)
dq(2)=dq2(ijp)
dq(3)=dq3(ijp)
dq(4)=dq4(ijp)

call multmv (cijp,dq,dq,4,4)
```

```
! set constant terms for plotting
!
  xmin=0.0
  xmax=0.0
  ymin=0.0
  ymax=0.0

  do j=1,jmax
  do i=1,imax
    ij=ind(i,j)
    xmin=smin1(xmin,x(ij))
    xmax=smax1(xmax,x(ij))
    ymin=smin1(ymin,y(ij))
    ymax=smax1(ymax,y(ij))
  enddo
  enddo

  if (isympl.gt.0) ymin=-ymax
  d1=xmax-xmin
  d2=ymax-ymin
  d3=smax1(d1,d2)
  sf=1.0/d3
  xshft=0.5*(1.0-d1*sf)
  yshft=0.5*(1.0-d2*sf)
  dxmin=ep10

  do j=2,jm1
  do i=2,im1
    ij=ind(i,j)
    imj=ij-1
    ijm=ij-imax
    imjm=ij-1-imax
    dx=sabs(x(ij)-x(imj))
    dx=smax1(dx,sabs(x(ij)-x(imjm)))
    dx=smax1(dx,sabs(x(ij)-x(ijm)))
    dxmin=smin1(dx,dxmin)
  enddo
  enddo

  dymin=ep10

  do j=2,jm1
  do i=2,im1
    ij=ind(i,j)
    imj=ij-1
    ijm=ij-imax
    imjm=ij-1-imax
    dy=sabs(y(ij)-y(imj))
    dy=smax1(dy,sabs(y(ij)-y(imjm)))
    dy=smax1(dy,sabs(y(ij)-y(ijm)))
    dymin=smin1(dy,dymin)
  enddo
  enddo

  velmx1=velmx*smin1(dxmin,dymin)
!
! calculate laminar and total viscosity for mesh
!
  if (.not.restrt) then
    if (invisd.eq.0) then
      do j=1,jmax
      do i=1,imax
```

```
        ij=ind(i,j)
        mulm(ij)=cmu
        mu(ij)=cmu
      enddo
      enddo
    endif
!
! calculate laminar and turbulent thermal conductivity for mesh
!
    if (invisd.eq.0) then
      do j=1,jmax
      do i=1,imax
        ij=ind(i,j)
        kulm(ij)=cku
        ku(ij)=cku
      enddo
      enddo
    endif
!
! set initial p, u, v, t
!
    do j=1,jmax
    do i=1,imax
      ij=ind(i,j)
      p(ij)=pi
      u(ij)=ui
      v(ij)=vi
      t(ij)=ti
    enddo
    enddo
!
! set initial r, ru, rv, e
!
    do j=1,jmax
    do i=1,imax
      ij=ind(i,j)
      q1(ij)=p(ij)/t(ij)/rgas
      q2(ij)=q1(ij)*u(ij)
      q3(ij)=q1(ij)*v(ij)
      velsq=u(ij)*u(ij)+v(ij)*v(ij)
      q4(ij)=p(ij)/gm1+0.5*q1(ij)*velsq
    enddo
    enddo
 endif
!
! evaluate metric coefficients a1...a11 and b1...b11 for viscous terms
! evaluation
!
 if (invisd.lt.1) call ab1t11

 return
 end

 subroutine readfi
!
!*****************************************************************************
! open the restart input file and read the initial field.
!*****************************************************************************
!
 include 'params.cmd'
 include 'precs.cmd'
 include 'arrays.cmd'
```

```
 include 'param.cmd'
 include 'bcon.cmd'

 character*11 flfm

 include 'aindex.cmd'

 flfm='unformatted'
 if (irstfm.gt.0) flfm='formatted  '

 call ocfile (lurst,fnrst,dfrst,flfm,1)

 do j=1,jmax
 do i=1,imax
   ij=ind(i,j)

   if (irstfm.gt.0) then
     read(lurst,'(2x,1p4e15.8)') q1(ij),q2(ij),q3(ij),q4(ij)
   else
     read(lurst) q1(ij),q2(ij),q3(ij),q4(ij)
   endif
 enddo
 enddo

 call ocfile (lurst,fnrst,dfrst,flfm,0)

 do j=1,jmax
 do i=1,imax
   ij=ind(i,j)
   u(ij)=q2(ij)/q1(ij)
   v(ij)=q3(ij)/q1(ij)
   velsq=u(ij)*u(ij)+v(ij)*v(ij)
   p(ij)=gm1*(q4(ij)-0.5*q1(ij)*velsq)
   t(ij)=p(ij)/rgas/q1(ij)
   to(ij)=t(ij)
   c=ssqrt(gamma*rgas*t(ij))
   vel=ssqrt(velsq+em20)
   mach(ij)=vel/c
 enddo
 enddo

 return
 end

 subroutine print (n)
!
!*****************************************************************************
! this subroutine provides formatted writes to paper.
!*****************************************************************************
!
 include 'params.cmd'
 include 'precs.cmd'
 include 'arrays.cmd'
 include 'param.cmd'
 include 'bcon.cmd'

 include 'aindex.cmd'
!
! print (2) write time step, cycle information
!
 if (n.eq.1 .or.n.eq.2) then
   if (iprint.eq.2 .or. iprint.eq.3) &
```

```
      write(luout,'(/,6x,a,1pe12.5,5x,a,1pe12.5,5x,a,i5)') &
        ' time= ',ttime,' delt= ',delt,' cycle= ',cycle

    return
  endif

  if (n.eq.3) then
    if (iprint.eq.0) return
    if (iprint.eq.1 .or. iprint.eq.3) write(luscn,'(a80)') name
    if (iprint.eq.2 .or. iprint.eq.3) write(luout,'(a80)') name

    if (iprint.eq.1 .or. iprint.eq.3) &
      write(luscn,'(a,1pe10.3)') ' ttime =',ttime
    if (iprint.eq.2 .or. iprint.eq.3) &
      write(luout,'(a,1pe10.3)') ' ttime =',ttime

    if (iprint.eq.1 .or. iprint.eq.3) write(luscn,'(a,2x,a,4x,a,6x,a,10x, &
      a,10x,a,10x,a,8x,a,6x,a,10x,a)') ' i ',' j ','density','u','v','p', &
      't','mach','mu','tcon'
    if (iprint.eq.2 .or. iprint.eq.3) write(luout,'(a,2x,a,4x,a,6x,a,10x, &
      a,10x,a,10x,a,8x,a,6x,a,10x,a)') ' i ',' j ','density','u','v','p', &
      't','mach','mu','tcon'

    do j=1,jmax
    do i=1,imax
      ij=ind(i,j)

      if (iprint.eq.1 .or. iprint.eq.3) &
        write(luscn,'(i3,2x,i3,2x,8(1pe10.3,1x))') i,j,q1(ij),u(ij),v(ij), &
           p(ij),t(ij),mach(ij),mu(ij),ku(ij)

      if (iprint.eq.2 .or. iprint.eq.3) &
        write(luout,'(i3,2x,i3,2x,8(1pe10.3,1x))') i,j,q1(ij),u(ij),v(ij), &
           p(ij),t(ij),mach(ij),mu(ij),ku(ij)
    enddo
    enddo

    return
  endif

  end

  subroutine ocfile (unfl,nmfl,dfnfl,fmfl,open)

  integer unfl, open
  character*11 fmfl
  character*30 nmfl, dfnfl

  if (open.gt.0) then
    open (unit=unfl,file=nmfl,access='sequential',form=fmfl,status='unknown')
    rewind (unit=unfl)
  else
    close (unit=unfl)
  endif

  return
  end

  subroutine metric
!
!****************************************************************************
! read the grid information from an unformatted output file created by tomcat
```

```
!
! find the jacobian for the corner cells
!
 ij=ind(1,1)
 ic=ind(2,2)
 jac(ij)=jac(ic)

 ij=ind(1,jmax)
 ic=ind(2,jm1)
 jac(ij)=jac(ic)

 ij=ind(imax,1)
 ic=ind(im1,2)
 jac(ij)=jac(ic)

 ij=ind(imax,jmax)
 ic=ind(im1,jm1)
 jac(ij)=jac(ic)
!
! find the jacobian for the boundary cells
!
 i=1
 do j=2,jm1
   ij=ind(i,j)
   ipj=ij+1
   jac(ij)=jac(ipj)
 enddo

 i=im1

 do j=2,jm1
   ij=ind(i,j)
   ipj=ij+1
   jac(ipj)=jac(ij)
 enddo

 j=1

 do i=2,im1
   ij=ind(i,j)
   ijp=ij+imax
   jac(ij)=jac(ijp)
 enddo

 j=jm1

 do i=2,im1
   ij=ind(i,j)
   ijp=ij+imax
   jac(ijp)=jac(ij)
 enddo
!
! evaluate the area and components of unit normal at right face of each cell, also evaluate
! height of face center for axisymmetric runs
!
 do j=2,jm1
 do i=1,im1
   ij=ind(i,j)
   ijm=ij-imax

   call geom (1,1,unxr(ij),unyr(ij),arer(ij),volt1,volt2)
```

```
   if (cyl.gt.0.0) arer(ij)=arer(ij)*0.5*(y(ij)+y(ijm))
  enddo
  enddo
!
! evaluate the area and components of unit normal at top face of each cell, also evaluate
! hight of face center for axisymmetric runs
!
  do j=1,jm1
  do i=2,im1
   ij=ind(i,j)
   imj=ij-1

   call geom (1,2,unxt(ij),unyt(ij),aret(ij),volt1,volt2)

   if (cyl.gt.0.0) aret(ij)=aret(ij)*0.5*(y(ij)+y(imj))
  enddo
  enddo

  if (iprint.ne.4) return

  write(luscn,'(1x,a80)') name
  write(luscn,'(a,2x,a,8x,a,12x,a,10x,a,9x,a,9x,a,9x,a,9x,a,9x,a,9x,a,/)') &
      ' i ',' j ','x','y','jac','arer','unxr','unyr','aret','unxt','unyt'

  do j=1,jm1
  do i=1,im1
   ij=ind(i,j)

   write(luscn,'(i3,2x,i3,2x,9(1pe12.5,1x))') i,j,x(ij),y(ij),jac(ij), &
       arer(ij),unxr(ij),unyr(ij),aret(ij),unxt(ij),unyt(ij)
  enddo
  enddo

  return
  end

!
!*****************************************************************************
! matrix.for contains matrix operation files.
!*****************************************************************************
!
  subroutine trid (aa,bb,cc,dd,il,iu)
!
!*****************************************************************************
! subroutine to solve non-periodic tridiagonal system
!
! a    = sub diagonal matrix
! b    = diagonal matrix
! c    = up diagonal matrix
! d    = right hand side vector
! il   = lower value of index for which matrices are defined
! iu   = upper value of index for which matrices are defined
!        (solution is sought for trid (a,b,c) * x = d
!        for indices of x between il and iu (inclusive)
!        solution written in d vector (original content lost)
!*****************************************************************************
!
  implicit real*8 (a-h,o-z)
  real*8 aa, bb, cc, dd

  dimension aa(1), bb(1), cc(1), dd(1)
!
```

```
! establish upper triangular matrix
!
  lp=il+1

  do i=lp,iu
    r=aa(i)/bb(i-1)
    bb(i)=bb(i)-r*cc(i-1)
    dd(i)=dd(i)-r*dd(i-1)
  enddo
!
! back substitution
!
  dd(iu)=dd(iu)/bb(iu)

  do i=lp,iu
    j=iu-i+il
    dd(j)=(dd(j)-cc(j)*dd(j+1))/bb(j)
  enddo

  return
  end
!-----------------------------------------------------------------------------
  subroutine btrid (a,b,c,d,il,iu,order)
!
!**********************************************************************************
! subroutine to solve non-periodic block tridiagonal system of equations with
! pivoting strategy with the dimensions of the block matrices being n x n
! (n is any number greater than 1)
!
! a     = sub diagonal matrix
! b     = diagonal matrix
! c     = sup diagonal matrix
! d     = right hand side vector
! il    = lower value of index for which matrices are defined
! iu    = upper value of index for which matrices are defined
!         (solution is sought for btrid (a,b,c) * x = d
!         for indices of x between il and iu (inclusive)
!         solution written in d vector (original content lost)
! order = order of a, b, c matrices and length of d
!         at each point denoted by i
!         (order can be greater than 1)
!**********************************************************************************
!
  implicit real*8 (a-h,o-z)
  real*8 a,b,c,d
  integer order,ordsq

  dimension a(1), b(1), c(1), d(1), indx(8)

  ordsq=order*order
!
! forward elimination
!
  i=il
  iomat=1+(i-1)*ordsq
  iovec=1+(i-1)*order

  call ludcmp (b(iomat),order,order,indx,flag)

  if (flag.eq.0) then
    write(6,'(a,i4)') ' the matrix is near singular at point i = ',i
```

```
    stop
   endif

   call lubksb (b(iomat),order,order,indx,d(iovec))

   do j=1,order
     iomatj=iomat+(j-1)*order

     call lubksb (b(iomat),order,order,indx,c(iomatj))
   enddo

 1 continue

   i=i+1
   iomat=1+(i-1)*ordsq
   iovec=1+(i-1)*order
   i1mat=iomat-ordsq
   i1vec=iovec-order

   call mulput (a(iomat),d(i1vec),d(iovec),order)

   do j=1,order
     iomatj=iomat+(j-1)*order
     i1matj=i1mat+(j-1)*order

     call mulput (a(iomat),c(i1matj),b(iomatj),order)
   enddo

   call ludcmp (b(iomat),order,order,indx,flag)

   if (flag.eq.0) then
     write(6,'(a,i4)') ' the matrix is near singular at point i = ',i

     stop
   endif

   call lubksb (b(iomat),order,order,indx,d(iovec))

   if (i.eq.iu) go to 2

   do j=1,order
     iomatj=iomat+(j-1)*order

     call lubksb (b(iomat),order,order,indx,c(iomatj))
   enddo

   go to 1

2 continue
!
! back substitution
!
  i=iu

3 continue
    i=i-1
    iomat=1+(i-1)*ordsq
    iovec=1+(i-1)*order
    i1vec=iovec+order

    call mulput (c(iomat),d(i1vec),d(iovec),order)
  if (i.gt.il) go to 3
```

```
 return
 end
!-----------------------------------------------------------------------------
 subroutine ludcmp (a,n,np,indx,d)
!
!******************************************************************************
! compute l-u decomposition of a given matrix n x n with physical dimension np
! by crout's method with partial pivoting. indx is an output vector which
! records the row permutation affected by partial pivoting. d is +1 or -1
! depending on whether the number of row interchanges was even or odd. d equal
! to zero indicates that the matrix is singular. this routine is used in
! combination with lubksb to solve linear equations or invert a matrix.
!******************************************************************************
!
 implicit real*8 (a-h,o-z)
 real*8 a,d

 dimension a(1), indx(n), vv(10)

 data tiny/1.0d 20/

 d=1.0

 do i=1,n
   aamax=0.0

   do j=1,n
     ij=i+(j-1)*n
     if (sabs(a(ij)).gt.aamax) aamax=sabs(a(ij))
   enddo

   if (aamax.eq.0.0) then
     d=0.0
     return
   endif

   vv(i)=1.0/aamax
 enddo

 do j=1,n
   if (j.gt.1) then
     do i=1,j-1
       ij=i+(j-1)*n
       sum=a(ij)

       if (i.gt.1) then
         do k=1,i-1
           ik=i+(k-1)*n
           kj=k+(j-1)*n
           sum=sum-a(ik)*a(kj)
         enddo

         a(ij)=sum
       endif
     enddo
   endif

   aamax=0.0

   do i=j,n
     ij=i+(j-1)*n
```

```
      sum=a(ij)

      if (j.gt.1) then
        do k=1,j-1
          ik=i+(k-1)*n
          kj=k+(j-1)*n
          sum=sum-a(ik)*a(kj)
        enddo

        a(ij)=sum
      endif

      dum=vv(i)*sabs(sum)

      if (dum.ge.aamax) then
        imax=i
        aamax=dum
      endif
    enddo

    if (j.ne.imax) then
      do k=1,n
        imxk=imax+(k 1)*n
        jk=j+(k 1)*n
        dum=a(imxk)
        a(imxk)=a(jk)
        a(jk)=dum
      enddo

      d=-d
      vv(imax)=vv(j)
    endif

    indx(j)=imax
    jj=j+(j-1)*n
    if (sabs(a(jj)).le.tiny) a(jj)=tiny

    if (j.ne.n) then
      dum=1.0/a(jj)

      do i=j+1,n
        ij=i+(j-1)*n
        a(ij)=a(ij)*dum
      enddo
    endif
  enddo

  return
  end
!-----------------------------------------------------------------------------
  subroutine lubksb (a,n,np,indx,b)
!
!***********************************************************************************
! solve linear algebraic system of equations a*x = b and store results in
! vector b. matrix a is input in l-u decomposition form. b is input as right
! hand side vector. a, n, np and indx are not modified by this routine. this
! routine takes into account the possibility that b will begin with many zero
! elements, so it is efficient in matrix inversion.
!***********************************************************************************
!
  implicit real*8 (a-h,o-z)
  real*8 a,b
```

```
 dimension a(1), indx(n), b(n)

 ii=0

 do i=1,n
   ll=indx(i)
   sum=b(ll)
   b(ll)=b(i)

   if (ii.ne.0) then
     do j=ii,i-1
       ij=i+(j-1)*n
       sum=sum-a(ij)*b(j)
     enddo
   else if (sum.ne.0.0) then
     ii=i
   endif

   b(i)=sum
 enddo

 do i=n,1,-1
   sum=b(i)

   if (i.lt.n) then
     do j=i+1,n
       ij=i+(j -1)*n
       sum=sum-a(ij)*b(j)
     enddo
   endif

   ii=i+(i-1)*n
   b(i)=sum/a(ii)
 enddo

 return
 end
!-----------------------------------------------------------------------------
 subroutine mulput (a,b,c,order)
!
!*******************************************************************************
! multiply a vector b by a matrix a, subtract result from another vector c and
! store result in c. c is overwritten.
!*******************************************************************************
!
 implicit real*8 (a-h,o-z)
 real*8 a,b,c
 integer order

 dimension a(1), b(1), c(1)

 do jr=1,order
   sum=0.0

   do jc=1,order
     ia=jr+(jc-1)*order
     sum=sum+a(ia)*b(jc)
   enddo

   c(jr)=c(jr)-sum
 enddo
```

```
 return
 end
!-----------------------------------------------------------------------------
 subroutine multmv (a, b, c, p, q)
!
!******************************************************************************
! multiply matrix a by vector b, that is c = a b
! a is a (p,q) matrix, b is a (q) vector, c is a (p) vector
!******************************************************************************
!
 implicit real*8 (a-h,o-z)
 real*8 a,b,c
 integer p, q

 dimension a(1), b(1), c(1), temp(10)

 do i=1,p
   sum=0.0

   do j=1,q
     ij=i+(j-1)*p
     sum=sum+a(ij)*b(j)
   enddo

   temp(i)=sum
 enddo

 do i=1,p
   c(i)=temp(i)
 enddo

 return
 end
!-----------------------------------------------------------------------------
 subroutine mult2m (a, b, c, p, q, r)
!
!******************************************************************************
! multiply two matrices a, b. the product matrix is c = a b
! a is a (p,q) matrix, b is a (q,r) matrix, c is a (p,r) matrix
!******************************************************************************
!
 implicit real*8 (a-h,o-z)
 real*8 a,b,c
 integer p,q,r

 dimension a(1), b(1), c(1), temp(100)

 do j=1,r
   do i=1,p
     ij=i+(j-1)*p
     sum=0.0

     do k=1,q
       ik=i+(k 1)*p
       kj=k+(j 1)*q
       sum=sum+a(ik)*b(kj)
     enddo

     temp(ij)=sum
   enddo
 enddo
```

```
 do j=1,r
 do i=1,p
   ij=i+(j-1)*p
   c(ij)=temp(ij)
 enddo
 enddo

 return
 end

 subroutine geom (iopt,ifac,unx,uny,area,volt1,volt2)
!
!*****************************************************************
! find geometrical information
!
! ifac=0 : no face
! ifac=1 : right face
! ifac=2 : top face
! ifac=3 : left face
! ifac=4 : bottom face
!
! iopt=1 : find components of unit normal and area of face ifac
! iopt=2 : find volumes of two triangles making the cell
! iopt=3 : find all the above
!*****************************************************************
!
 include 'params.cmd'
 include 'precs.cmd'
 include 'arrays.cmd'
 include 'param.cmd'
 include 'bcon.cmd'

 real*8 unx, uny, area, volt1, volt2

 ijm=ij-imax
 imj=ij-1
 imjm=ij-1-imax

 if (ifac.eq.1) then
   x1=x(ijm)
   y1=y(ijm)
   x2=x(ij)
   y2=y(ij)
   sgn=1.0
 elseif (ifac.eq.2) then
   x1=x(ij)
   y1=y(ij)
   x2=x(imj)
   y2=y(imj)
   sgn=1.0
 elseif (ifac.eq.3) then
   x1=x(imj)
   y1=y(imj)
   x2=x(imjm)
   y2=y(imjm)
   sgn=-1.0
 elseif (ifac.eq.4) then
   x1=x(imjm)
   y1=y(imjm)
   x2=x(ijm)
   y2=y(ijm)
```

```
    sgn=-1.0
  endif

  if (iopt.ne.2) then
    dely=y2-y1
    delx=x2-x1
    area=ssqrt(dely*dely+delx*delx)
    unx=sgn*dely/area
    uny=-sgn*delx/area
  elseif (iopt.eq.2 .or. iopt.eq.3) then
    x1=x(imjm)
    y1=y(imjm)
    x2=x(ijm)
    y2=y(ijm)
    x3=x(ij)
    y3=y(ij)
    x4=x(imj)
    y4=y(imj)
    delx12=x2-x1
    dely12=y2-y1
    delx13=x3-x1
    dely13=y3-y1
    delx14=x4-x1
    dely14=y4-y1
    volt1=0.5*(delx12*dely13-delx13*dely12)
    volt2=0.5*(delx13*dely14-delx14*dely13)
  endif

  return
  end

!
!*******************************************************************************
! functn contains system library functions used in code. user could use single
! or double precision. real*4 for single precision, real*8 for double precision
!*******************************************************************************
!
!-----------------------------------------------------------------------------
  function sabs (arg1)

  real*8 arg1, sabs

  sabs=dabs(arg1)

  return
  end
!-----------------------------------------------------------------------------
  function sfloat (arg1)

  real*8 sfloat
  integer arg1

  sfloat=float(arg1)

  return
  end
!-----------------------------------------------------------------------------
  function slog (arg1)

  real*8 arg1, slog

  slog=dlog(arg1)
```

```
 return
 end
!------------------------------------------------------------------------------
 function slog10 (arg1)

 real*8 arg1, slog10

 slog10=dlog10(arg1)

 return
 end
!------------------------------------------------------------------------------
 function smax1 (arg1,arg2)

 real*8 arg1, arg2, smax1

 smax1=dmax1(arg1,arg2)

 return
 end
!------------------------------------------------------------------------------
 function smin1 (arg1,arg2)

 real*8 arg1, arg2, smin1

 smin1=dmin1(arg1,arg2)

 return
 end
!------------------------------------------------------------------------------
 function ssign (arg1,arg2)

 real*8 arg1, arg2, ssign

 ssign=dsign(arg1,arg2)

 return
 end
!------------------------------------------------------------------------------
 function ssqrt (arg1)

 real*8 arg1, ssqrt

 ssqrt=dsqrt(arg1)

 return
 end

 subroutine errors (cij,cijm,cijp,dij)
!
!*******************************************************************************
! calculate the total errors in the computational domain for line gauss-seidel.
!*******************************************************************************
!
 include 'params.cmd'
 include 'precs.cmd'
 include 'arrays.cmd'
 include 'param.cmd'
 include 'bcon.cmd'

 real*8 cij, cijm, cijp, dij
```

```
dimension cij(16), cijm(16), cijp(16), dij(4), delq(4)

include 'aindex.cmd'

delq(1)=dq1(ij)
delq(2)=dq2(ij)
delq(3)=dq3(ij)
delq(4)=dq4(ij)

call multmv (cij,delq,delq,4,4)

dij(1)=dij(1)-delq(1)
dij(2)=dij(2)-delq(2)
dij(3)=dij(3)-delq(3)
dij(4)=dij(4)-delq(4)

ijm=ij-imax
delq(1)=dq1(ijm)
delq(2)=dq2(ijm)
delq(3)=dq3(ijm)
delq(4)=dq4(ijm)

call multmv (cijm,delq,delq,4,4)

dij(1)=dij(1)-delq(1)
dij(2)=dij(2)-delq(2)
dij(3)=dij(3)-delq(3)
dij(4)=dij(4)-delq(4)

ijp=ij+imax
delq(1)=dq1(ijp)
delq(2)=dq2(ijp)
delq(3)=dq3(ijp)
delq(4)=dq4(ijp)

call multmv (cijp,delq,delq,4,4)

dij(1)=dij(1)-delq(1)
dij(2)=dij(2)-delq(2)
dij(3)=dij(3)-delq(3)
dij(4)=dij(4)-delq(4)

errl(1)=errl(1)+sabs(dij(1))
errl(2)=errl(2)+sabs(dij(2))
errl(3)=errl(3)+sabs(dij(3))
errl(4)=errl(4)+sabs(dij(4))

return
end

subroutine dflux (unxw,unyw,hare,df1,df2,df3,df4)
!
!*****************************************************************************
! find the values of artificial viscosity component of e and f fluxes at right
! or top wall of a cell.
!*****************************************************************************
!
include 'params.cmd'
include 'precs.cmd'
include 'arrays.cmd'
include 'param.cmd'
```

```
include 'bcon.cmd'

real*8 unxw, unyw, hare, df1, df2, df3, df4

dimension evr(4,4), evl(4,4), delq(4)

do kj=1,4
do ki=1,4
  kikj=ki+(kj-1)*4
  avea(kikj)=0.0
enddo
enddo

ubm=unxw*um+unyw*vm
cmsq=gamma*rgas*tm
cm=ssqrt(cmsq)

ubp=unxw*up+unyw*vp
cpsq=gamma*rgas*tp
cp=ssqrt(cpsq)

hp=(ep+pp)/rp
hm=(em+pm)/rm

rpm=ssqrt(rp/rm)
rpmp1=1.0+rpm
uh=(um+up*rpm)/rpmp1
vh=(vm+vp*rpm)/rpmp1
ubh=unxw*uh+unyw*vh
hh=(hm+hp*rpm)/rpmp1
qsq=0.5*(uh*uh+vh*vh)
chsq=gm1*(hh-qsq)
chsq=smax1(chsq,smin1(cpsq,cmsq))
ch=ssqrt(chsq)
gqsq=gm1*qsq/ch

delq(1)=rp-rm
delq(2)=rup-rum
delq(3)=rvp-rvm
delq(4)=ep-em

ev1=sabs(ubh-ch)
ev2=sabs(ubh)
ev3=sabs(ubh+ch)

ev1p=(ubp-cp)
ev3p=(ubp+cp)

ev1m=(ubm-cm)
ev3m=(ubm+cm)

dev1=ev1p-ev1m
dev3=ev3p-ev3m
dev1s=0.5*smax1(dev1,zero)
dev3s=0.5*smax1(dev3,zero)

if (ev1.le.dev1s) ev1=ev1*ev1/dev1+0.25*dev1
if (ev3.le.dev3s) ev3=ev3*ev3/dev3+0.25*dev3

evr(1,1)=0.5*ev1/ch
evr(2,1)=0.5*ev1*(uh/ch-unxw)
evr(3,1)=0.5*ev1*(vh/ch-unyw)
```

```
 evr(4,1)=0.5*ev1*(hh/ch-ubh)

 evr(1,2)=ev2/ch
 evr(2,2)=ev2*uh/ch
 evr(3,2)=ev2*vh/ch
 evr(4,2)=ev2*qsq/ch

 evr(1,3)=0.0
 evr(2,3)=-ev2*unyw
 evr(3,3)=ev2*unxw
 evr(4,3)=ev2*(vh*unxw-uh*unyw)

 evr(1,4)=0.5*ev3/ch
 evr(2,4)=0.5*ev3*(uh/ch+unxw)
 evr(3,4)=0.5*ev3*(vh/ch+unyw)
 evr(4,4)=0.5*ev3*(hh/ch+ubh)

 evl(1,1)=gqsq+ubh
 evl(1,2)=-gm1*uh/ch-unxw
 evl(1,3)=-gm1*vh/ch-unyw
 evl(1,4)=gm1/ch

 evl(2,1)=-gqsq+ch
 evl(2,2)=gm1*uh/ch
 evl(2,3)=gm1*vh/ch
 evl(2,4)=-gm1/ch

 evl(3,1)=uh*unyw-vh*unxw
 evl(3,2)=-unyw
 evl(3,3)=unxw
 evl(3,4)=0.0

 evl(4,1)=gqsq-ubh
 evl(4,2)=-gm1*uh/ch+unxw
 evl(4,3)=-gm1*vh/ch+unyw
 evl(4,4)=gm1/ch

 call mult2m (evr,evl,evl,4,4,4)

 evmax=smax1(ev1,ev2)
 evmax=smax1(evmax,ev3)

 do kj=1,4
 do ki=1,4
   kikj=ki+(kj-1)*4
   if (lspeed.gt.0) avea(kikj)=hare*evl(ki,kj)
   if (lspeed.lt.1 .and. ki.eq.kj .and. steady) avea(kikj)=hare*evmax
 enddo
 enddo

 call multmv (evl,delq,delq,4,4)

 df1=hare*delq(1)
 df2=hare*delq(2)
 df3=hare*delq(3)
 df4=hare*delq(4)

 return
 end

 subroutine delta
!
```

```
!*****************************************************************************
! compute the maximum value of delt needed for stability of explicit scheme as
! well as implicit scheme with explicit boundary condition.
!*****************************************************************************
!
  include 'params.cmd'
  include 'precs.cmd'
  include 'arrays.cmd'
  include 'param.cmd'
  include 'bcon.cmd'

  include 'aindex.cmd'

  if (cycle.le.fcycle+1) return

  deltn=ep10

  do j=2,jm1
  do i=2,im1
    ij=ind(i,j)
    volume=jac(ij)
    cons=volume*cfl
    c=ssqrt(gamma*rgas*t(ij))
    ubar=unxr(ij)*u(ij)+unyr(ij)*v(ij)
    vbar=unxt(ij)*u(ij)+unyt(ij)*v(ij)
    eigxim=sabs(ubar)+c
    eigetm=sabs(vbar)+c
    dtcxi=cons/eigxim/(arer(ij)+em10)
    dtcet=cons/eigetm/(aret(ij)+em10)
    dt(ij)=dtcxi*dtcet/(dtcxi+dtcet)
    deltn=smin1(deltn,dt(ij))

    if (invisd.lt.1 .and. implct.lt.1) then
      diff=smax1(mu(ij),ku(ij)/cvg)
      aresq=arer(ij)*arer(ij)+aret(ij)*aret(ij)
      dtvis=0.2*q1(ij)*cons*volume/diff/aresq
      dt(ij)=smin1(dt(ij),dtvis)
      deltn=smin1(deltn,dtvis)
    endif
  enddo
  enddo

  deltn=smin1(deltn,pltdt)
  if (deltn.lt.delt) delt=deltn
  if (autot.gt.0.0) delt=deltn

  if (.not.steady) then
    do j=2,jm1
    do i=2,im1
      ij=ind(i,j)
      dt(ij)=delt
    enddo
    enddo
  endif

  return
  end

  subroutine convrg (dq1dq1,dq2dq2,dq3dq3,dq4dq4,dqtdqt)
!
!*****************************************************************************
! check for convergence to steady state soluiton.
```

```
!**************************************************************************
!
 include 'params.cmd'
 include 'precs.cmd'
 include 'arrays.cmd'
 include 'param.cmd'
 include 'bcon.cmd'

 include 'aindex.cmd'

 real*8 dq1dq1, dq2dq2, dq3dq3, dq4dq4, dqtdqt

 iflag=0

 delq1=em20
 delq2=em20
 delq3=em20
 delq4=em20

 dq1dq1=em20
 dq2dq2=em20
 dq3dq3=em20
 dq4dq4=em20
 dqtdqt=em20

 q1q1=em20
 q2q2=em20
 q3q3=em20
 q4q4=em20
 qtqt=em20

 if (cycle.le.fcycle) return

do j=2,jm1
do i=2,im1
   ij=ind(i,j)
   d1=dq1(ij)
   d2=dq2(ij)
   d3=dq3(ij)
   d4=dq4(ij)
   qi=q1(ij)
   qii=q2(ij)
   qiii=q3(ij)
   qiv=q4(ij)

   dq1dq1=dq1dq1+d1*d1*jac(ij)*jac(ij)
   dq2dq2=dq2dq2+d2*d2*jac(ij)*jac(ij)
   dq3dq3=dq3dq3+d3*d3*jac(ij)*jac(ij)
   dq4dq4=dq4dq4+d4*d4*jac(ij)*jac(ij)

   q1q1=q1q1+qi*qi*jac(ij)*jac(ij)
   q2q2=q2q2+qii*qii*jac(ij)*jac(ij)
   q3q3=q3q3+qiii*qiii*jac(ij)*jac(ij)
   q4q4=q4q4+qiv*qiv*jac(ij)*jac(ij)

   delq1=delq1+sabs(d1*jac(ij))
   delq2=delq2+sabs(d2*jac(ij))
   delq3=delq3+sabs(d3*jac(ij))
   delq4=delq4+sabs(d4*jac(ij))
enddo
enddo
```

```
dqtdqt=dq1dq1+dq2dq2+dq3dq3+dq4dq4
qtqt=q1q1+q2q2+q3q3+q4q4

if (mod(cycle,ifreq).eq.0) then
  dq1max=em20
  dq2max=em20
  dq3max=em20
  dq4max=em20
  dqtmax=em20

  i1max=1000
  i2max=1000
  i3max=1000
  i4max=1000
  itmax=1000

  j1max=1000
  j2max=1000
  j3max=1000
  j4max=1000
  jtmax=1000

  do j=2,jm1
  do i=2,im1
    ij=ind(i,j)
    d1=dq1(ij)
    d2=dq2(ij)
    d3=dq3(ij)
    d4=dq4(ij)
    dq1abs=dabs(d1)*jac(ij)
    dq2abs=dabs(d2)*jac(ij)
    dq3abs=dabs(d3)*jac(ij)
    dq4abs=dabs(d4)*jac(ij)
    dqtabs=dq1abs+dq2abs+dq3abs+dq4abs

    if (dq1abs.gt.dq1max) then
      dq1max=dq1abs
      i1max=i
      j1max=j
    endif

    if (dq2abs.gt.dq2max) then
      dq2max=dq2abs
      i2max=i
      j2max=j
    endif

    if (dq3abs.gt.dq3max) then
      dq3max=dq3abs
      i3max=i
      j3max=j
    endif

    if (dq4abs.gt.dq4max) then
      dq4max=dq4abs
      i4max=i
      j4max=j
    endif

    if (dqtabs.gt.dqtmax) then
      dqtmax=dqtabs
      itmax=i
```

```
        jtmax=j
      endif
    enddo
    enddo

    dq1max=ssqrt(sfloat(im2jm2))*dq1max/ssqrt(q1q1)
    dq2max=ssqrt(sfloat(im2jm2))*dq2max/ssqrt(q2q2)
    dq3max=ssqrt(sfloat(im2jm2))*dq3max/ssqrt(q3q3)
    dq4max=ssqrt(sfloat(im2jm2))*dq4max/ssqrt(q4q4)
    dqtmax=ssqrt(sfloat(im2jm2))*dqtmax/ssqrt(qtqt)

    write(luscn,'(a,1pe11.3,a,2i4)') &
        ' dq1max = ',dq1max,' at i ,j =',i1max,j1max
    write(luscn,'(a,1pe11.3,a,2i4)') &
        ' dq2max = ',dq2max,' at i ,j =',i2max,j2max
    write(luscn,'(a,1pe11.3,a,2i4)') &
        ' dq3max = ',dq3max,' at i ,j =',i3max,j3max
    write(luscn,'(a,1pe11.3,a,2i4)') &
        ' dq4max = ',dq4max,' at i ,j =',i4max,j4max
    write(luscn,'(a,1pe11.3,a,2i4)') &
        ' dqtmax = ',dqtmax,' at i ,j =',itmax,jtmax
  endif

  dq1dq1=ssqrt(dq1dq1/q1q1)
  dq2dq2=ssqrt(dq2dq2/q2q2)
  dq3dq3=ssqrt(dq3dq3/q3q3)
  dq4dq4=ssqrt(dq4dq4/q4q4)
  dqtdqt=ssqrt(dqtdqt/qtqt)

  delq1=terr(1)/delq1
  delq2=terr(2)/delq2
  delq3=terr(3)/delq3
  delq4=terr(4)/delq4

  if (delq1.le.epsi .and. delq2.le.epsi .and. delq3.le.epsi .and. &
    delq4.le.epsi) iflag=1

  return
  end

  subroutine calvis
!
!*******************************************************************************
! calculate the viscous flux vector ev at the right face and vector fv at the
! top face of cell (i,j) using central differencing.
!*******************************************************************************
!
  include 'params.cmd'
  include 'precs.cmd'
  include 'arrays.cmd'
  include 'param.cmd'
  include 'bcon.cmd'

  real*8 kur, kut

  ev1=0.0
  ev2=0.0
  ev3=0.0
  ev4=0.0

  fv1=0.0
  fv2=0.0
```

```fortran
 fv3=0.0
 fv4=0.0

 ipj=ij+1
 ijp=ij+imax
 ipjp=ijp+1
 ijm=ij-imax
 ipjm=ijm+1
 imj=ij-1
 imjp=imj+imax
!
! calculate ev vector
!
 if (j.eq.1 .or. j.eq.jmax) go to 1
 if (i.eq.1 .and. wtlb(j).le.1) go to 1
 if (i.eq.im1 .and. wtrb(j).le.1) go to 1

 mur=2.0*mu(ij)*mu(ipj)/(mu(ij)+mu(ipj))
 kur=2.0*ku(ij)*ku(ipj)/(ku(ij)+ku(ipj))
 uc=u(ij)
 vc=v(ij)
 tc=t(ij)
 ur=(u(ij)+u(ipj))/2.0
 vr=(v(ij)+v(ipj))/2.0
 urr=u(ipj)
 vrr=v(ipj)
 trr=t(ipj)
 urt=(u(ijp)+u(ipjp))/2.0
 vrt=(v(ijp)+v(ipjp))/2.0
 trt=(t(ijp)+t(ipjp))/2.0
 urbs=(u(ijm)+u(ipjm))/2.0
 vrbs=(v(ijm)+v(ipjm))/2.0
 trb=(t(ijm)+t(ipjm))/2.0
 dudx=(urr-uc)
 dude=(urt-urbs)/2.0
 dvdx=(vrr-vc)
 dvde=(vrt-vrbs)/2.0
 dtdx=(trr-tc)
 dtde=(trt-trb)/2.0

 ev1=0.0
 ev2=mur*(a1(ij)*dudx+a5(ij)*dude+a2(ij)*dvdx+a6(ij)*dvde+a7(ij)*vr)
 ev3=mur*(a2(ij)*dudx+a8(ij)*dude+a3(ij)*dvdx+a9(ij)*dvde+a10(ij)*vr)
 ev4=kur*(a4(ij)*dtdx+a11(ij)*dtde)+ur*ev2+vr*ev3

1 continue
!
! calculate fv vector
!
 if (i.eq.1 .or. i.eq.imax) return
 if (j.eq.1 .and. wtbb(i).le.1) return
 if (j.eq.jm1 .and. wttb(i).le.1) return

 mut=2.0*mu(ij)*mu(ijp)/(mu(ij)+mu(ijp))
 kut=2.0*ku(ij)*ku(ijp)/(ku(ij)+ku(ijp))
 uc=u(ij)
 vc=v(ij)
 tc=t(ij)
 ut=(u(ij)+u(ijp))/2.0
 vt=(v(ij)+v(ijp))/2.0
 utt=u(ijp)
 vtt=v(ijp)
```

```
ttt=t(ijp)
utr=(u(ipj)+u(ipjp))/2.0
vtr=(v(ipj)+v(ipjp))/2.0
ttr=(t(ipj)+t(ipjp))/2.0
utl=(u(imj)+u(imjp))/2.0
vtl=(v(imj)+v(imjp))/2.0
ttl=(t(imj)+t(imjp))/2.0
dudx=(utr-utl)/2.0
dude=(utt-uc)
dvdx=(vtr-vtl)/2.0
dvde=(vtt-vc)
dtdx=(ttr-ttl)/2.0
dtde=(ttt-tc)

fv1=0.0
fv2=mut*(b5(ij)*dudx+b1(ij)*dude+b6(ij)*dvdx+b2(ij)*dvde+b7(ij)*vt)
fv3=mut*(b8(ij)*dudx+b2(ij)*dude+b9(ij)*dvdx+b3(ij)*dvde+b10(ij)*vt)
fv4=kut*(b11(ij)*dtdx+b4(ij)*dtde)+ut*fv2+vt*fv3

return
end

subroutine calstd
!
!*******************************************************************************
! call on other subroutines to get the necessary arrays for calculation of
! steady state part of the solution which is put into array omg.
!*******************************************************************************
!
include 'params.cmd'
include 'precs.cmd'
include 'arrays.cmd'
include 'param.cmd'
include 'bcon.cmd'

include 'aindex.cmd'

do j=2,jm1
do i=2,im1
  ij=ind(i,j)
  omg1(ij)=0.0
  omg2(ij)=0.0
  omg3(ij)=0.0
  omg4(ij)=0.0
enddo
enddo

do j=1,jm1
do i=1,im1
  ij=ind(i,j)
  ipj=ij+1
  ijp=ij+imax
!
! get the flux at the right face
!
  call calepm

  omg1(ij)=omg1(ij)-ei1
  omg2(ij)=omg2(ij)-ei2
  omg3(ij)=omg3(ij)-ei3
  omg4(ij)=omg4(ij)-ei4
```

```
      omg1(ipj)=omg1(ipj)+ei1
      omg2(ipj)=omg2(ipj)+ei2
      omg3(ipj)=omg3(ipj)+ei3
      omg4(ipj)=omg4(ipj)+ei4

      if (icheck .and. iflux.eq.3) then
        do kj=1,4
        do ki=1,4
          kikj=ki+(kj-1)*4
          indx=kikj+(ij-1)*16
          ap(indx)=ap(indx)+avea(kikj)
          am(indx)=am(indx)-avea(kikj)
        enddo
        enddo
      endif
!
! get the flux at the top face
!
      call calfpm

      omg1(ij)=omg1(ij)-fi1
      omg2(ij)=omg2(ij)-fi2
      omg3(ij)=omg3(ij)-fi3
      omg4(ij)=omg4(ij)-fi4

      omg1(ijp)=omg1(ijp)+fi1
      omg2(ijp)=omg2(ijp)+fi2
      omg3(ijp)=omg3(ijp)+fi3
      omg4(ijp)=omg4(ijp)+fi4

      if (icheck .and. iflux.eq.3) then
        do kj=1,4
        do ki=1,4
          kikj=ki+(kj-1)*4
          indx=kikj+(ij-1)*16
          bp(indx)=bp(indx)+avea(kikj)
          bm(indx)=bm(indx)-avea(kikj)
        enddo
        enddo
      endif
!
! get the source term h due to gravity and axisymmetry
!
      call calsrc

      omg1(ij)=omg1(ij)+h1
      omg2(ij)=omg2(ij)+h2
      omg3(ij)=omg3(ij)+h3
      omg4(ij)=omg4(ij)+h4
!
! get the viscous fluxes ev and fv
!
      if (invisd.eq.0) then
        call calvis

        omg1(ij)=omg1(ij)+ev1+fv1
        omg2(ij)=omg2(ij)+ev2+fv2
        omg3(ij)=omg3(ij)+ev3+fv3
        omg4(ij)=omg4(ij)+ev4+fv4

        omg1(ipj)=omg1(ipj)-ev1
        omg2(ipj)=omg2(ipj)-ev2
```

```
      omg3(ipj)=omg3(ipj)-ev3
      omg4(ipj)=omg4(ipj)-ev4

      omg1(ijp)=omg1(ijp)-fv1
      omg2(ijp)=omg2(ijp)-fv2
      omg3(ijp)=omg3(ijp)-fv3
      omg4(ijp)=omg4(ijp)-fv4
    endif
  enddo
  enddo
!
! call on additional sources
!
  if (source) call addsrc
  if (wshear .and. invisd.lt.1) call wallsh
!
! print omg for each cell
!
  if (iprint.eq.5) then
    do j=2,jm1
    do i=2,im1
      ij=ind(i,j)

      write(luscn,'(3i5,1p4e15.4)') i,j,ij,omg1(ij),omg2(ij),omg3(ij),omg4(ij)
    enddo
    enddo
  endif

  return
  end

  subroutine calsrc
!
!*******************************************************************************
! compute the source term associated with gravity and axisymmetry.
!*******************************************************************************
!
  include 'params.cmd'
  include 'precs.cmd'
  include 'arrays.cmd'
  include 'param.cmd'
  include 'bcon.cmd'

  real*8 jacc

  h1=0.0
  h2=0.0
  h3=0.0
  h4=0.0

  if (i.eq.1 .or. i.eq.imax .or. j.eq.1 .or. j.eq.jmax) return

  jacc=jac(ij)

  if (cyl.gt.0.0) then
    call geom (2,0,unx,uny,area,volt1,volt2)

    jacc=volt1+volt2
  endif

  pc=p(ij)
  uc=u(ij)
```

```
  vc=v(ij)
  tathth=0.0

  if (invisd.eq.0 .and. cyl.eq.1.0) then
    imj=ij-1
    ijm=ij-imax
    imjm=ijm-1
    ipj=ij+1
    ijp=ij+imax
    xr=(x(ij)+x(ijm))/2.0
    xl=(x(imj)+x(imjm))/2.0
    xt=(x(ij)+x(imj))/2.0
    xb=(x(ijm)+x(imjm))/2.0
    yr=(y(ij)+y(ijm))/2.0
    yl=(y(imj)+y(imjm))/2.0
    yt=(y(ij)+y(imj))/2.0
    yb=(y(ijm)+y(imjm))/2.0
    xixc=(yt-yb)/jacc
    xiyc=-(xt-xb)/jacc
    etxc=-(yr-yl)/jacc
    etyc=(xr-xl)/jacc
    muc=mu(ij)
    urr=u(ipj)
    vrr=v(ipj)
    ull=u(imj)
    vll=v(imj)
    utt=u(ijp)
    vtt=v(ijp)
    ubbs=u(ijm)
    vbbs=v(ijm)
    dudx=(urr-ull)/2.0
    dude=(utt-ubbs)/2.0
    dvdx=(vrr-vll)/2.0
    dvde=(vtt-vbbs)/2.0
    tathth=muc*c2*(2.0*vc/yc(ij)-xixc*dudx-etxc*dude-xiyc*dvdx-etyc*dvde)
  endif

  h1=0.0
  h2=jac(ij)*q1(ij)*gx
  h3=cyl*jacc*(pc-tathth)+jac(ij)*q1(ij)*gy
  h4=jac(ij)*q1(ij)*(uc*gx+vc*gy)

  return
  end

  subroutine calqpm (wall,iord)
!
!******************************************************************************
! find the values of t, u, v, and p or q1, q2, q3, and q4 at a specified cell
! wall based on the minmod property.
!******************************************************************************
!
  include 'params.cmd'
  include 'precs.cmd'
  include 'arrays.cmd'
  include 'param.cmd'
  include 'bcon.cmd'

  character*1 wall
!
! primitive variable interpolation
!
```

```
! right wall
!
  if (wall.eq.'r') then
     imj=ij-1
     ipj=ij+1
     ip2j=ij+2

     if (i.eq.1 .or. i.eq.im1 .or. iord.eq.1) then
       rm=q1(ij)
       um=u(ij)
       vm=v(ij)
       pm=p(ij)
     else
       dm=q1(ij)-q1(imj)
       dp=q1(ipj)-q1(ij)
       rm=q1(ij)+grad(dm,dp,iord,limit,dlim)

       dm=u(ij)-u(imj)
       dp=u(ipj)-u(ij)
       um=u(ij)+grad(dm,dp,iord,limit,dlim)

       dm=v(ij)-v(imj)
       dp=v(ipj)-v(ij)
       vm=v(ij)+grad(dm,dp,iord,limit,dlim)

       dm=p(ij)-p(imj)
       dp=p(ipj)-p(ij)
       pm=p(ij)+grad(dm,dp,iord,limit,dlim)
     endif

     if (i.eq.1 .or. i.eq.im1 .or. iord.eq.1) then
       rp=q1(ipj)
       up=u(ipj)
       vp=v(ipj)
       pp=p(ipj)
     else
       dm=q1(ipj)-q1(ij)
       dp=q1(ip2j)-q1(ipj)
       rp=q1(ipj)-grad(dp,dm,iord,limit,dlim)

       dm=u(ipj)-u(ij)
       dp=u(ip2j)-u(ipj)
       up=u(ipj)-grad(dp,dm,iord,limit,dlim)

       dm=v(ipj)-v(ij)
       dp=v(ip2j)-v(ipj)
       vp=v(ipj)-grad(dp,dm,iord,limit,dlim)

       dm=p(ipj)-p(ij)
       dp=p(ip2j)-p(ipj)
       pp=p(ipj)-grad(dp,dm,iord,limit,dlim)
     endif
  endif
!
! top wall
!
  if (wall.eq.'t') then
     ijm=ij-imax
     ijp=ij+imax
     ijp2=ijp+imax

     if (j.eq.1 .or. j.eq.jm1 .or. iord.eq.1) then
```

```
     rm=q1(ij)
     um=u(ij)
     vm=v(ij)
     pm=p(ij)
   else
     dm=q1(ij)-q1(ijm)
     dp=q1(ijp)-q1(ij)
     rm=q1(ij)+grad(dm,dp,iord,limit,dlim)

     dm=u(ij)-u(ijm)
     dp=u(ijp)-u(ij)
     um=u(ij)+grad(dm,dp,iord,limit,dlim)

     dm=v(ij)-v(ijm)
     dp=v(ijp)-v(ij)
     vm=v(ij)+grad(dm,dp,iord,limit,dlim)

     dm=p(ij)-p(ijm)
     dp=p(ijp)-p(ij)
     pm=p(ij)+grad(dm,dp,iord,limit,dlim)
   endif

   if (j.eq.1 .or. j.eq.jm1 .or. iord.eq.1) then
     rp=q1(ijp)
     up=u(ijp)
     vp=v(ijp)
     pp=p(ijp)
   else
     dm=q1(ijp)-q1(ij)
     dp=q1(ijp2)-q1(ijp)
     rp=q1(ijp)-grad(dp,dm,iord,limit,dlim)

     dm=u(ijp)-u(ij)
     dp=u(ijp2)-u(ijp)
     up=u(ijp)-grad(dp,dm,iord,limit,dlim)

     dm=v(ijp)-v(ij)
     dp=v(ijp2)-v(ijp)
     vp=v(ijp)-grad(dp,dm,iord,limit,dlim)

     dm=p(ijp)-p(ij)
     dp=p(ijp2)-p(ijp)
     pp=p(ijp)-grad(dp,dm,iord,limit,dlim)
   endif
  endif

  tm=pm/rgas/rm
  rum=rm*um
  rvm=rm*vm
  em=pm/gm1+0.5*rm*(um*um+vm*vm)
  hm=(em+pm)/rm

  tp=pp/rgas/rp
  rup=rp*up
  rvp=rp*vp
  ep=pp/gm1+0.5*rp*(up*up+vp*vp)
  hp=(ep+pp)/rp

  return
  end
!-----------------------------------------------------------------------------
  function grad (dm,dp,iord,minmod,cons)
```

```
!
!*******************************************************************************
! this function controls the amount and type of upwinding that is
! used in the interpolation routine.
!
! minmod=0 is the straight extrapolation without limiter.
! minmod=1 is the famous minmod-type limiter.
! minmod=2 is the van-leer's universal limiter.
! minmod=3 is the van-leer's second order upwind scheme that uses
!        harmonic mean of two gradients.
! minmod=4 is the van-albada's second order limiter.
! minmod=5 is the Koren third order limiter.
! minmod=6 is the second order upwind scheme that uses minimum of
!        two gradients.
!*******************************************************************************
!
  implicit real*8 (a-h,o-z)
  external smin1, smax1, ssign, sabs

  real*8 grad
  integer iord, minmod

  dimension rk(4), dw(4)

  data rk /-1.0d0, 1.0d0, 0.33333333d0, 0.0d0/
  data dw /2.0d0, 3.0d0, 4.0d0, 3.0d0/
  data eps /1.0d 10/

  if (minmod.eq.0) then
    if (cons.lt.0.0d0) grad=0.25d0*((1.0d0-rk(iord))*dm+(1.0d0+rk(iord))*dp)
    if (cons.le.1.0d0 .and. cons.ge.0.0d0) grad=0.5d0*dm
    if (cons.gt.1.0d0) grad=(dm+3.0d0*dp)/8.0d0

    return
  endif

5 if (minmod.eq.1) then
    dws=1.0d0+(dw(iord)-1.0d0)*cons
    sgnx=ssign(1.0d0,dm)
    absx=sabs(dm)
    dwy=dp*dws
    dmq=smin1(absx,dwy*sgnx)
    dmq=sgnx*smax1(0.0d0,dmq)
    sgnx=ssign(1.0d0,dp)
    absx=sabs(dp)
    dwy=dm*dws
    dpq=smin1(absx,dwy*sgnx)
    dpq=sgnx*smax1(0.0d0,dpq)
    grad=0.25d0*((1.0d0-rk(iord))*dmq+(1.0d0+rk(iord))*dpq)
  elseif (minmod.eq.2) then
    sl=2.0d0*dm*dp/(dm*dm+dp*dp+eps)
    ft=(1.0d0-rk(iord)*sl)*dm+(1.0d0+rk(iord)*sl)*dp
    grad=0.25d0*sl*ft
  elseif (minmod.eq.3) then
    grad=smax1(dm*dp,0.0d0)/(dm+dp+eps)
  elseif (minmod.eq.4) then
    dmq=dm*dm+1.0d-2
    dpq=dp*dp+1.0d-2
    grad=(dp*dmq+dm*dpq)/(dmq+dpq)
  elseif (minmod.eq.5) then
    sl=dp/(dm+eps)
    ft=2.0d0*sl*sl
```

```
    sl=(sl+ft)/(2.0d0-sl+ft)
    grad=0.5d0*sl*dm
  elseif (minmod.eq.6) then
    grad=0.0d0
    if (dm*dp.le.eps) return
    absdm=sabs(dm)
    absdp=sabs(dp)
    grad=0.5d0*smin1(absdm,absdp)*dm/absdm
  endif

  return
  end

  subroutine calprm
!
!*******************************************************************************
! calculate the primitive variables.
!*******************************************************************************
!
  include 'params.cmd'
  include 'precs.cmd'
  include 'arrays.cmd'
  include 'param.cmd'
  include 'bcon.cmd'

  include 'aindex.cmd'
!
! find the primitive values in the interior of mesh
!
  do j=2,jm1
  do i=2,im1
    ij=ind(i,j)
    u(ij)=q2(ij)/q1(ij)
    v(ij)=q3(ij)/q1(ij)
    velsq=u(ij)*u(ij)+v(ij)*v(ij)
    p(ij)=gm1*(q4(ij)-0.5*q1(ij)*velsq)
    to(ij)=t(ij)
    t(ij)=p(ij)/rgas/q1(ij)

    if (p(ij).le.0.0) write(luscn,'(a,i3,a,i3,a,1pe14.4)') &
      ' i = ',i,' j = ',j,' p = ',p(ij)

    if (t(ij).le.0.0) write(luscn,'(a,i3,a,i3,a,1pe14.4)') &
      ' i = ',i,' j = ',j,' p = ',t(ij)

    if (q1(ij).le.0.0) write(luscn,'(a,i3,a,i3,a,1pe14.4)') &
      ' i = ',i,' j = ',j,' p = ',q1(ij)

    if (q4(ij).le.0.0) write(luscn,'(a,i3,a,i3,a,1pe14.4)') &
      ' i = ',i,' j = ',j,' p = ',q4(ij)

    c=ssqrt(gamma*rgas*t(ij))
    vel=ssqrt(velsq+em20)
    mach(ij)=vel/c
  enddo
  enddo
!
! find the primitive values at the corners of mesh
!
  ij=ind(1,1)
  ic=ind(2,2)
  u(ij)=u(ic)
```

```
 v(ij)=v(ic)
 p(ij)=p(ic)
 to(ij)=t(ij)
 t(ij)=t(ic)
 mach(ij)=mach(ic)

 ij=ind(1,jmax)
 ic=ind(2,jm1)
 u(ij)=u(ic)
 v(ij)=v(ic)
 p(ij)=p(ic)
 to(ij)=t(ij)
 t(ij)=t(ic)
 mach(ij)=mach(ic)

 ij=ind(imax,1)
 ic=ind(im1,2)
 u(ij)=u(ic)
 v(ij)=v(ic)
 p(ij)=p(ic)
 to(ij)=t(ij)
 t(ij)=t(ic)
 mach(ij)=mach(ic)

 ij=ind(imax,jmax)
 ic=ind(im1,jm1)
 u(ij)=u(ic)
 v(ij)=v(ic)
 p(ij)=p(ic)
 to(ij)=t(ij)
 t(ij)=t(ic)
 mach(ij)=mach(ic)

 return
 end

 subroutine caljac
!
!*****************************************************************************
! calculate the jacobian of fluxes.
!*****************************************************************************
!
 include 'params.cmd'
 include 'precs.cmd'
 include 'arrays.cmd'
 include 'param.cmd'
 include 'bcon.cmd'

 include 'aindex.cmd'

 icheck=.false.

 if ((cycle-1.le.fcycle .or. mod(cycle 1,jupdat).eq.0) .and. implct.eq.1) &
   icheck=.true.

 if (.not.icheck) return

 do j=1,jm1
 do i=1,im1
   ij=ind(i,j)
!
! find dedq and devdq
```

```
!
    call caldei

    if (invisd.lt.1) call caldev
!
! find dfdq and dfvdq
!
    call caldfi
    if (invisd.lt.1) call caldfv
!
! find dhdq
!
    call caldhq

  enddo
  enddo

  return
  end

  subroutine calisp
!
!*****************************************************************************
! this subroutine find the specific impulse.
!*****************************************************************************
!
  include 'params.cmd'
  include 'precs.cmd'
  include 'arrays.cmd'
  include 'param.cmd'
  include 'bcon.cmd'

  character*10 wtype

  include 'aindex.cmd'
!
! right wall
!
  i=im1
  smass=em20
  simpx=em20
  simpy=em20
  wtype='right wall'

  do j=2,jm1
    ij=ind(i,j)
    vnorm=u(ij)*unxr(ij)+v(ij)*unyr(ij)
    smass=smass+arer(ij)*q1(ij)*vnorm
    simpx=simpx+arer(ij)*(q1(ij)*vnorm*u(ij)+unxr(ij)*p(ij))
    simpy=simpy+arer(ij)*(q1(ij)*vnorm*v(ij)+unyr(ij)*p(ij))
  enddo

  tmass=sabs(smass*(1.0+cyl*5.283185308))
  thrst=sabs(simpx*(1.0+cyl*5.283185308))
  simpx=simpx/smass/9.81

  write(luscn,'(/,2a,/,a,1pe12.5,/,a,1pe12.5,/,a,1pe12.5)') ' at the ',wtype, &
     ' mass flow rate     = ',tmass,' x-direction thrust = ',thrst, &
     ' x-direction Isp    = ',simpx

  write(luout,'(/,2a,/,a,1pe12.5,/,a,1pe12.5,/,a,1pe12.5)') ' at the ',wtype, &
     ' mass flow rate     = ',tmass,' x-direction thrust = ',thrst, &
```

```
     ' x-direction Isp    = ',simpx

  return
  end

  subroutine calfpm
!
!******************************************************************************
! calculate the positive and negative portion of the split convective flux in
! eta-direction based on a variety of schemes.
!
! iflux=1 does the van leer scheme
! iflux=2 does the steger-warming scheme
! iflux=3 does the roe scheme
!******************************************************************************
!
  include 'params.cmd'
  include 'precs.cmd'
  include 'arrays.cmd'
  include 'param.cmd'
  include 'bcon.cmd'

  data del1 /1.0d-10/

  fi1=0.0
  fi2=0.0
  fi3=0.0
  fi4=0.0

  if (i.eq.1 .or. i.eq.imax) return

  unxw=unxt(ij)
  unyw=unyt(ij)
  arew=aret(ij)
!
! find cell wall interpolated values
!
  call calqpm ('t',inoret)

  vbp=unxw*up+unyw*vp
  vbm=unxw*um+unyw*vm

  if (iflux.eq.1) then
!
! evaluate the flux based on van leer's flux-splitting
! calculate fp
!
    cm=ssqrt(gamma*rgas*tm)
    met=vbm/cm

    if (met.le.-clone) then
      f1p=0.0
      f2p=0.0
      f3p=0.0
      f4p=0.0
    elseif (met.gt.-clone .and. met.lt.clone) then
      metp=met+1.0
      f1p=0.25*rm*cm*metp*metp
      f2p=f1p*(unxw/gamma*(-vbm+2.0*cm)+um)
      f3p=f1p*(unyw/gamma*(-vbm+2.0*cm)+vm)
      f4p=f1p*(em+pm)/rm
    else
```

```
        f1p=rm*vbm
        f2p=rum*vbm+unxw*pm
        f3p=rvm*vbm+unyw*pm
        f4p=vbm*(em+pm)
      endif
!
! calculate fm
!
      cp=ssqrt(gamma*rgas*tp)
      met=vbp/cp

      if (met.le.-clone) then
        f1m=rp*vbp
        f2m=rup*vbp+unxw*pp
        f3m=rvp*vbp+unyw*pp
        f4m=vbp*(ep+pp)
      elseif (met.gt.-clone .and. met.lt.clone) then
        metm=met-1.0
        f1m=-0.25*rp*cp*metm*metm
        f2m=f1m*(unxw/gamma*(-vbp-2.0*cp)+up)
        f3m=f1m*(unyw/gamma*(-vbp-2.0*cp)+vp)
        f4m=f1m*(ep+pp)/rp
      else
        f1m=0.0
        f2m=0.0
        f3m=0.0
        f4m=0.0
      endif
!
! compute total flux
!
      fi1=arew*(f1p+f1m)
      fi2=arew*(f2p+f2m)
      fi3=arew*(f3p+f3m)
      fi4=arew*(f4p+f4m)
    elseif (iflux.eq.2) then
!
! evaluate the flux based on eigen value splitting (steger-warming
! flux-splitting)
! calculate fp
!
      cm=ssqrt(gamma*rgas*tm)
      met=vbm/cm

      if (met.le.-clone) then
        f1p=0.0
        f2p=0.0
        f3p=0.0
        f4p=0.0
      elseif (met.gt.-clone .and. met.le.zero) then
        ev3=vbm+cm
        ev3=(ev3+ssqrt(ev3*ev3+del1))/2.0
        cons=ev3*rm/gamma/2.0
        f1p=cons
        f2p=cons*(um+cm*unxw)
        f3p=cons*(vm+cm*unyw)
        f4p=cons*(cm*vbm+(um*um+vm*vm)/2.0+cm*cm/gm1)
      elseif (met.gt.zero .and. met.lt.clone) then
        ev1=vbm-cm
        ev1=(ev1-ssqrt(ev1*ev1+del1))/2.0
        cons=ev1*rm/gamma/2.0
        f1p=rm*vbm-cons
```

```
      f2p=rum*vbm+unxw*pm-cons*(um-cm*unxw)
      f3p=rvm*vbm+unyw*pm-cons*(vm-cm*unyw)
      f4p=vbm*(em+pm)-cons*(-cm*vbm+(um*um+vm*vm)/2.0+cm*cm/gm1)
    else
      f1p=rm*vbm
      f2p=rum*vbm+unxw*pm
      f3p=rvm*vbm+unyw*pm
      f4p=vbm*(em+pm)
    endif
!
! calculate fm
!
    cp=ssqrt(gamma*rgas*tp)
    met=vbp/cp

    if (met.le.-clone) then
      f1m=rp*vbp
      f2m=rup*vbp+unxw*pp
      f3m=rvp*vbp+unyw*pp
      f4m=vbp*(ep+pp)
    elseif (met.gt.-clone .and. met.le.zero) then
      ev3=vbp+cp
      ev3=(ev3+ssqrt(ev3*ev3+del1))/2.0
      cons=ev3*rp/gamma/2.0
      f1m=rp*vbp-cons
      f2m=rup*vbp+unxw*pp-cons*(up+cp*unxw)
      f3m=rvp*vbp+unyw*pp-cons*(vp+cp*unyw)
      f4m=vbp*(ep+pp)-cons*(cp*vbp+(up*up+vp*vp)/2.0+cp*cp/gm1)
    elseif (met.gt.zero .and. met.lt.clone) then
      ev1=vbp-cp
      ev1=(ev1-ssqrt(ev1*ev1+del1))/2.0
      cons=ev1*rp/gamma/2.0
      f1m=cons
      f2m=cons*(up-cp*unxw)
      f3m=cons*(vp-cp*unyw)
      f4m=cons*(-cp*vbp+(up*up+vp*vp)/2.0+cp*cp/gm1)
    else
      f1m=0.0
      f2m=0.0
      f3m=0.0
      f4m=0.0
    endif
!
! compute total flux
!
    fi1=arew*(f1p+f1m)
    fi2=arew*(f2p+f2m)
    fi3=arew*(f3p+f3m)
    fi4=arew*(f4p+f4m)
  elseif (iflux.eq.3) then
!
! evaluate the flux based on roe's flux-difference splitting
!
    hare=0.5*arew

    call dflux (unxw,unyw,hare,f1a,f2a,f3a,f4a)

    if ((j.eq.1 .and. wtbb(i).le.2) .or. (j.eq.jm1 .and. wttb(i).le.2)) then
      ph=pp+pm
      fi1=-damp*f1a
      fi2=hare*unxw*ph-damp*f2a
      fi3=hare*unyw*ph-damp*f3a
```

```
        fi4=-damp*f4a

        do kj=1,4
        do ki=1,4
          kikj=ki+(kj-1)*4
          avea(kikj)=dampi*avea(kikj)
        enddo
        enddo

        return
      endif
!
! calculate fp
!
      f1p=rm*vbm
      f2p=rum*vbm+unxw*pm
      f3p=rvm*vbm+unyw*pm
      f4p=vbm*(em+pm)
!
! calculate fm
!
      f1m=rp*vbp
      f2m=rup*vbp+unxw*pp
      f3m=rvp*vbp+unyw*pp
      f4m=vbp*(ep+pp)
!
! compute total flux
!
      fi1=hare*(f1p+f1m)-f1a
      fi2=hare*(f2p+f2m)-f2a
      fi3=hare*(f3p+f3m)-f3a
      fi4=hare*(f4p+f4m)-f4a
   endif

   return
   end

   subroutine calepm
!
!*****************************************************************************
! calculate the positive and negative portion of the split convective flux in
! xi-direction based on a variety of schemes.
!
! iflux=1 does the van leer scheme
! iflux=2 does the steger-warming scheme
! iflux=3 does the roe scheme
!*****************************************************************************
!
   include 'params.cmd'
   include 'precs.cmd'
   include 'arrays.cmd'
   include 'param.cmd'
   include 'bcon.cmd'

   data del1 /1.0d-10/

   ei1=0.0
   ei2=0.0
   ei3=0.0
   ei4=0.0

   if (j.eq.1 .or. j.eq.jmax) return
```

```
  unxw=unxr(ij)
  nyw=unyr(ij)
  arew=arer(ij)
!
! find cell wall interpolated values
!
  call calqpm ('r',inorxi)

  ubp=unxw*up+unyw*vp
  ubm=unxw*um+unyw*vm

  if (iflux.eq.1) then
!
! evaluate the flux based on van leer's flux-splitting
! calculate ep
!
    cm=ssqrt(gamma*rgas*tm)
    mxi=ubm/cm

    if (mxi.le.-clone) then
      e1p=0.0
      e2p=0.0
      e3p=0.0
      e4p=0.0
    elseif (mxi.gt.-clone .and. mxi.lt.clone) then
      mxip=mxi+1.0
      e1p=0.25*rm*cm*mxip*mxip
      e2p=e1p*(unxw/gamma*(-ubm+2.0*cm)+um)
      e3p=e1p*(unyw/gamma*(-ubm+2.0*cm)+vm)
      e4p=e1p*(em+pm)/rm
    else
      e1p=rm*ubm
      e2p=rum*ubm+unxw*pm
      e3p=rvm*ubm+unyw*pm
      e4p=ubm*(em+pm)
    endif
!
! calculate em
!
    cp=ssqrt(gamma*rgas*tp)
    mxi=ubp/cp

    if (mxi.le.-clone) then
      e1m=rp*ubp
      e2m=rup*ubp+unxw*pp
      e3m=rvp*ubp+unyw*pp
      e4m=ubp*(ep+pp)
    elseif (mxi.gt.-clone .and. mxi.lt.clone) then
      mxim=mxi-1.0
      e1m=-0.25*rp*cp*mxim*mxim
      e2m=e1m*(unxw/gamma*(-ubp-2.0*cp)+up)
      e3m=e1m*(unyw/gamma*(-ubp-2.0*cp)+vp)
      e4m=e1m*(ep+pp)/rp
    else
      e1m=0.0
      e2m=0.0
      e3m=0.0
      e4m=0.0
    endif
!
! compute total flux
```

```
!
    ei1=arew*(e1p+e1m)
    ei2=arew*(e2p+e2m)
    ei3=arew*(e3p+e3m)
    ei4=arew*(e4p+e4m)
  elseif (iflux.eq.2) then
!
! evaluate the flux based on eigen value splitting (steger-warming
! flux-splitting)
! calculate ep
!
    cm=ssqrt(gamma*rgas*tm)
    mxi=ubm/cm

    if (mxi.le.-clone) then
      e1p=0.0
      e2p=0.0
      e3p=0.0
      e4p=0.0
    elseif (mxi.gt.-clone .and. mxi.le.zero) then
      ev3=ubm+cm
      ev3=(ev3+ssqrt(ev3*ev3+del1))/2.0
      cons=ev3*rm/gamma/2.0
      e1p=cons
      e2p=cons*(um+cm*unxw)
      e3p=cons*(vm+cm*unyw)
      e4p=cons*(cm*ubm+(um*um+vm*vm)/2.0+cm*cm/gm1)
    elseif (mxi.gt.zero .and. mxi.lt.clone) then
      ev1=ubm-cm
      ev1=(ev1-ssqrt(ev1*ev1+del1))/2.0
      cons=ev1*rm/gamma/2.0
      e1p=rm*ubm-cons
      e2p=rum*ubm+unxw*pm-cons*(um-cm*unxw)
      e3p=rvm*ubm+unyw*pm-cons*(vm-cm*unyw)
      e4p=ubm*(em+pm)-cons*(-cm*ubm+(um*um+vm*vm)/2.0+cm*cm/gm1)
    else
      e1p=rm*ubm
      e2p=rum*ubm+unxw*pm
      e3p=rvm*ubm+unyw*pm
      e4p=ubm*(em+pm)
    endif
!
! calculate em
!
    cp=ssqrt(gamma*rgas*tp)
    mxi=ubp/cp

    if (mxi.le.-clone) then
      e1m=rp*ubp
      e2m=rup*ubp+unxw*pp
      e3m=rvp*ubp+unyw*pp
      e4m=ubp*(ep+pp)
    elseif (mxi.gt.-clone .and. mxi.le.zero) then
      ev3=ubp+cp
      ev3=(ev3+ssqrt(ev3*ev3+del1))/2.0
      cons=ev3*rp/gamma/2.0
      e1m=rp*ubp-cons
      e2m=rup*ubp+unxw*pp-cons*(up+cp*unxw)
      e3m=rvp*ubp+unyw*pp-cons*(vp+cp*unyw)
      e4m=ubp*(ep+pp)-cons*(cp*ubp+(up*up+vp*vp)/2.0+cp*cp/gm1)
    elseif (mxi.gt.zero .and. mxi.lt.clone) then
      ev1=ubp-cp
```

```
      ev1=(ev1-ssqrt(ev1*ev1+del1))/2.0
      cons=ev1*rp/gamma/2.0
      e1m=cons
      e2m=cons*(up-cp*unxw)
      e3m=cons*(vp-cp*unyw)
      e4m=cons*(-cp*ubp+(up*up+vp*vp)/2.0+cp*cp/gm1)
    else
      e1m=0.0
      e2m=0.0
      e3m=0.0
      e4m=0.0
    endif
!
! compute total flux
!
    ei1=arew*(e1p+e1m)
    ei2=arew*(e2p+e2m)
    ei3=arew*(e3p+e3m)
    ei4=arew*(e4p+e4m)
  elseif (iflux.eq.3) then
!
! evaluate the flux based on roe's flux-difference splitting
!
    hare=0.5*arew

    call dflux (unxw,unyw,hare,e1a,e2a,e3a,e4a)

    if ((i.eq.1 .and. wtlb(j).le.2) .or. (i.eq.im1 .and. wtrb(j).le.2)) then
      ph=pp+pm
      ei1=-damp*e1a
      ei2=hare*unxw*ph-damp*e2a
      ei3=hare*unyw*ph-damp*e3a
      ei4=-damp*e4a

      do kj=1,4
      do ki=1,4
        kikj=ki+(kj-1)*4
        avea(kikj)=dampi*avea(kikj)
      enddo
      enddo

      return
    endif
!
! calculate ep
!
    e1p=rm*ubm
    e2p=rum*ubm+unxw*pm
    e3p=rvm*ubm+unyw*pm
    e4p=ubm*(em+pm)
!
! calculate em
!
    e1m=rp*ubp
    e2m=rup*ubp+unxw*pp
    e3m=rvp*ubp+unyw*pp
    e4m=ubp*(ep+pp)
!
! compute total flux
!
    ei1=hare*(e1p+e1m)-e1a
    ei2=hare*(e2p+e2m)-e2a
```

```
   ei3=hare*(e3p+e3m)-e3a
   ei4=hare*(e4p+e4m)-e4a
  endif

  return
  end

  subroutine caldhq
!
!*****************************************************************************
! form jacobian matrix dh/dq of the source term due to gravity and axisymmetry
! for cell (i,j).
!*****************************************************************************
!
  include 'params.cmd'
  include 'precs.cmd'
  include 'arrays.cmd'
  include 'param.cmd'
  include 'bcon.cmd'

  real*8 jacc

  ind(ki,kj)=ki+(kj-1)*4+(ij-1)*16

  if (i.eq.1 .or. i.eq.imax .or. j.eq.1 .or. j.eq.jmax) return

  jacc=jac(ij)

  if (cyl.gt.0.0) then

    call geom (2,0,unx,uny,area,volt1,volt2)

    jacc=volt1+volt2
  endif

  uc=u(ij)
  vc=v(ij)
  dpdq1=0.5*gm1*(uc*uc+vc*vc)
  dpdq2=-gm1*uc
  dpdq3=-gm1*vc
  dpdq4=gm1

  dhdq(ind(2,1))=jac(ij)*gx

  dhdq(ind(3,1))=cyl*jacc*dpdq1+jac(ij)*gy
  dhdq(ind(3,2))=cyl*jacc*dpdq2
  dhdq(ind(3,3))=cyl*jacc*dpdq3
  dhdq(ind(3,4))=cyl*jacc*dpdq4

  dhdq(ind(4,2))=jac(ij)*gx
  dhdq(ind(4,3))=jac(ij)*gy

  return
  end

  subroutine caldfv
!
!*****************************************************************************
! form the jacobian matrix dfv/dq of the viscous flux in eta-direction at top
! wall of cell (i,j).
!*****************************************************************************
!
```

```
include 'params.cmd'
include 'precs.cmd'
include 'arrays.cmd'
include 'param.cmd'
include 'bcon.cmd'

real*8 kut

dimension dfvm(4,4), dfvp(4,4)

ind(ki,kj)=ki+(kj-1)*4+(ij-1)*16

if (invisd.gt.0) return
if (i.eq.1 .or. i.eq.imax) return
if (j.eq.1 .and. wtbb(i).ne.2) return
if (j.eq.jm1 .and. wttb(i).ne.2) return

do kj=1,4
do ki=1,4
  dfvp(ki,kj)=0.0
  dfvm(ki,kj)=0.0
enddo
enddo

ipj=ij+1
ijp=ij+imax
ipjp=ijp+1
imj=ij-1
imjp=imj+imax

mut=2.0*mu(ij)*mu(ijp)/(mu(ij)+mu(ijp))
kut=2.0*ku(ij)*ku(ijp)/(ku(ij)+ku(ijp))
uc=u(ij)
vc=v(ij)
ut=(u(ij)+u(ijp))/2.0
vt=(v(ij)+v(ijp))/2.0
utt=u(ijp)
vtt=v(ijp)
utr=(u(ipj)+u(ipjp))/2.0
vtr=(v(ipj)+v(ipjp))/2.0
utl=(u(imj)+u(imjp))/2.0
vtl=(v(imj)+v(imjp))/2.0
dudx=(utr-utl)/2.0
dude=(utt-uc)
dvdx=(vtr-vtl)/2.0
dvde=(vtt-vc)

fv2=mut*(b5(ij)*dudx+b1(ij)*dude+b6(ij)*dvdx+b2(ij)*dvde+b7(ij)*vt)
fv3=mut*(b8(ij)*dudx+b2(ij)*dude+b9(ij)*dvdx+b3(ij)*dvde+b10(ij)*vt)

dudq1p=-u(ij)/q1(ij)
dudq2p=1.0/q1(ij)
dvdq1p=-v(ij)/q1(ij)
dvdq3p=1.0/q1(ij)
dpdq1=0.5*gm1*(u(ij)*u(ij)+v(ij)*v(ij))
dpdq2=-gm1*u(ij)
dpdq3=-gm1*v(ij)
dpdq4=gm1
dtdq1p=(dpdq1-p(ij)/q1(ij))/q1(ij)/rgas
dtdq2p=dpdq2/q1(ij)/rgas
dtdq3p=dpdq3/q1(ij)/rgas
dtdq4p=dpdq4/q1(ij)/rgas
```

```
 dudq1m=-u(ijp)/q1(ijp)
 dudq2m=1.0/q1(ijp)
 dvdq1m=-v(ijp)/q1(ijp)
 dvdq3m=1.0/q1(ijp)
 dpdq1=0.5*gm1*(u(ijp)*u(ijp)+v(ijp)*v(ijp))
 dpdq2=-gm1*u(ijp)
 dpdq3=-gm1*v(ijp)
 dpdq4=gm1
 dtdq1m=(dpdq1-p(ijp)/q1(ijp))/q1(ijp)/rgas
 dtdq2m=dpdq2/q1(ijp)/rgas
 dtdq3m=dpdq3/q1(ijp)/rgas
 dtdq4m=dpdq4/q1(ijp)/rgas

 dfvp(2,1)=-mut*(b1(ij)*dudq1p+b2(ij)*dvdq1p-b7(ij)*dvdq1p/2.0)
 dfvp(2,2)=-mut*b1(ij)*dudq2p
 dfvp(2,3)=-mut*(b2(ij)*dvdq3p-b7(ij)*dvdq3p/2.0)

 dfvp(3,1)=-mut*(b2(ij)*dudq1p+b3(ij)*dvdq1p-b10(ij)*dvdq1p/2.0)
 dfvp(3,2)=-mut*b2(ij)*dudq2p
 dfvp(3,3)=-mut*(b3(ij)*dvdq3p-b10(ij)*dvdq3p/2.0)

 dfvp(4,1)=-kut*b4(ij)*dtdq1p+fv2*dudq1p/2.0+ut*dfvp(2,1)+fv3*dvdq1p/2.0 &
        +vt*dfvp(3,1)
 dfvp(4,2)=-kut*b4(ij)*dtdq2p+fv2*dudq2p/2.0+ut*dfvp(2,2)+vt*dfvp(3,2)
 dfvp(4,3)=-kut*b4(ij)*dtdq3p+ut*dfvp(2,3)+fv3*dvdq3p/2.0+vt*dfvp(3,3)
 dfvp(4,4)=-kut*b4(ij)*dtdq4p

 dfvm(2,1)=mut*(b1(ij)*dudq1m+b2(ij)*dvdq1m+b7(ij)*dvdq1m/2.0)
 dfvm(2,2)=mut*b1(ij)*dudq2m
 dfvm(2,3)=mut*(b2(ij)*dvdq3m+b7(ij)*dvdq3m/2.0)

 dfvm(3,1)=mut*(b2(ij)*dudq1m+b3(ij)*dvdq1m+b10(ij)*dvdq1m/2.0)
 dfvm(3,2)=mut*b2(ij)*dudq2m
 dfvm(3,3)=mut*(b3(ij)*dvdq3m+b10(ij)*dvdq3m/2.0)

 dfvm(4,1)=kut*b4(ij)*dtdq1m+fv2*dudq1m/2.0+ut*dfvm(2,1)+fv3*dvdq1m/2.0 &
        +vt*dfvm(3,1)
 dfvm(4,2)=kut*b4(ij)*dtdq2m+fv2*dudq2m/2.0+ut*dfvm(2,2)+vt*dfvm(3,2)
 dfvm(4,3)=kut*b4(ij)*dtdq3m+ut*dfvm(2,3)+fv3*dvdq3m/2.0+vt*dfvm(3,3)
 dfvm(4,4)=kut*b4(ij)*dtdq4m

 do kj=1,4
 do ki=1,4
   bp(ind(ki,kj))=bp(ind(ki,kj))-dfvp(ki,kj)
   bm(ind(ki,kj))=bm(ind(ki,kj))-dfvm(ki,kj)
 enddo
 enddo

 return
 end

 subroutine caldfi
!
!******************************************************************************
! form the jacobian matrices b+ = df+/dq and b- = df-/dq for the split flux in
! eta-direction at specified cell wall based on a variety of schemes.
!
! iflux=1 does the van leer scheme
! iflux=2 does the steger-warming scheme
! iflux=3 does the roe scheme
!******************************************************************************
```

```
!
 include 'params.cmd'
 include 'precs.cmd'
 include 'arrays.cmd'
 include 'param.cmd'
 include 'bcon.cmd'

 data del1 /1.0d-10/

 ind(ki,kj)=ki+(kj-1)*4+(ij-1)*16

 if (i.eq.1 .or. i.eq.imax) return

 unxw=unxt(ij)
 unyw=unyt(ij)
 arew=aret(ij)
!
! find cell wall interpolated values
!
 call calqpm ('t',inoret)

 vbm=unxw*um+unyw*vm
 vbp=unxw*up+unyw*vp

 if ((j.eq.1 .and. wtbb(i).le.2) .or. (j.eq.jm1 .and. wttb(i).le.2)) then
   vbm=dampi*vbm
   vbp=dampi*vbp
 endif

 if (iflux.eq.1) then
!
! evaluate the flux jacobian based on van leer's flux-vector splitting scheme
!
   qsqm=0.5*(um*um+vm*vm)
   qsqp=0.5*(up*up+vp*vp)
   gqsqm=gm1*qsqm
   gqsqp=gm1*qsqp
!
! calculate bp based on fp
!
   cm=ssqrt(gamma*rgas*tm)
   met=vbm/cm

   if (sabs(met).lt.clone) then
     metp=met+1.0
     f1p=arew*0.25*rm*cm*metp*metp
     df1pdr=f1p/rm
     df1pdu=arew*0.5*rm*unxw*metp
     df1pdv=arew*0.5*rm*unyw*metp
     df1pdc=arew*0.25*rm*(1.0-met)*metp

     dudq1=-um/rm
     dudq2=1.0/rm

     dvdq1=-vm/rm
     dvdq3=1.0/rm

     fact=0.5*gm1*cm/pm
     dcdq1=fact*(-em/rm+um*um+vm*vm)
     dcdq2=-fact*um
     dcdq3=-fact*vm
     dcdq4=fact
```

```
  b2p=unxw/gamma*(-vbm+2.0*cm)+um
  db2pdu=-unxw*unxw/gamma+1.0
  db2pdv=-unxw*unyw/gamma
  db2pdc=2.0/gamma*unxw

  b3p=unyw/gamma*(-vbm+2.0*cm)+vm
  db3pdu=-unxw*unyw/gamma
  db3pdv=-unyw*unyw/gamma+1.0
  db3pdc=2.0/gamma*unyw

  b4p=hm
  db4pdu=um
  db4pdv=vm
  db4pdc=2.0*cm/gm1

  bp(ind(1,1))=df1pdr+df1pdu*dudq1+df1pdv*dvdq1+df1pdc*dcdq1
  bp(ind(1,2))=df1pdu*dudq2+df1pdc*dcdq2
  bp(ind(1,3))=df1pdv*dvdq3+df1pdc*dcdq3
  bp(ind(1,4))=df1pdc*dcdq4

  bp(ind(2,1))=b2p*bp(ind(1,1))+f1p*(db2pdu*dudq1+db2pdv*dvdq1 &
         +db2pdc*dcdq1)
  bp(ind(2,2))=b2p*bp(ind(1,2))+f1p*(db2pdu*dudq2+db2pdc*dcdq2)
  bp(ind(2,3))=b2p*bp(ind(1,3))+f1p*(db2pdv*dvdq3+db2pdc*dcdq3)
  bp(ind(2,4))=b2p*bp(ind(1,4))+f1p*(db2pdc*dcdq4)

  bp(ind(3,1))=b3p*bp(ind(1,1))+f1p*(db3pdu*dudq1+db3pdv*dvdq1 &
         +db3pdc*dcdq1)
  bp(ind(3,2))=b3p*bp(ind(1,2))+f1p*(db3pdu*dudq2+db3pdc*dcdq2)
  bp(ind(3,3))=b3p*bp(ind(1,3))+f1p*(db3pdv*dvdq3+db3pdc*dcdq3)
  bp(ind(3,4))=b3p*bp(ind(1,4))+f1p*(db3pdc*dcdq4)

  bp(ind(4,1))=b4p*bp(ind(1,1))+f1p*(db4pdu*dudq1+db4pdv*dvdq1 &
         +db4pdc*dcdq1)
  bp(ind(4,2))=b4p*bp(ind(1,2))+f1p*(db4pdu*dudq2+db4pdc*dcdq2)
  bp(ind(4,3))=b4p*bp(ind(1,3))+f1p*(db4pdv*dvdq3+db4pdc*dcdq3)
  bp(ind(4,4))=b4p*bp(ind(1,4))+f1p*(db4pdc*dcdq4)
elseif (met.ge.clone) then
  unxw=arew*unxw
  unyw=arew*unyw
  vbm=arew*vbm

  bp(ind(1,1))=0.0
  bp(ind(1,2))=unxw
  bp(ind(1,3))=unyw
  bp(ind(1,4))=0.0

  bp(ind(2,1))=unxw*gqsqm-um*vbm
  bp(ind(2,2))=vbm-gm2*unxw*um
  bp(ind(2,3))=unyw*um-gm1*unxw*vm
  bp(ind(2,4))=unxw*gm1

  bp(ind(3,1))=unyw*gqsqm-vm*vbm
  bp(ind(3,2))=unxw*vm-gm1*unyw*um
  bp(ind(3,3))=vbm-gm2*unyw*vm
  bp(ind(3,4))=unyw*gm1

  bp(ind(4,1))=vbm*(gqsqm-hm)
  bp(ind(4,2))=unxw*hm-gm1*um*vbm
  bp(ind(4,3))=unyw*hm-gm1*vm*vbm
  bp(ind(4,4))=gamma*vbm
```

```
      unxw=unxt(ij)
      unyw=unyt(ij)
    else
      do kj=1,4
      do ki=1,4
        bp(ind(ki,kj))=0.0
      enddo
      enddo
    endif
!
! calculate bm based on fm
!
    cp=ssqrt(gamma*rgas*tp)
    met=vbp/cp

    if (sabs(met).lt.clone) then
      metm=met-1.0
      f1m=-arew*0.25*rp*cp*metm*metm
      df1mdr=f1m/rp
      df1mdu=-arew*0.5*rp*unxw*metm
      df1mdv=-arew*0.5*rp*unyw*metm
      df1mdc=-arew*0.25*rp*(1.0-met)*(1.0+met)

      dudq1=-up/rp
      dudq2=1.0/rp

      dvdq1=-vp/rp
      dvdq3=1.0/rp

      fact=0.5*gm1*cp/pp
      dcdq1=fact*(-ep/rp+up*up+vp*vp)
      dcdq2=-fact*up
      dcdq3=-fact*vp
      dcdq4=fact

      b2m=unxw/gamma*(-vbp-2.0*cp)+up
      db2mdu=-unxw*unxw/gamma+1.0
      db2mdv=-unxw*unyw/gamma
      db2mdc=-2.0/gamma*unxw

      b3m=unyw/gamma*(-vbp-2.0*cp)+vp
      db3mdu=-unxw*unyw/gamma
      db3mdv=-unyw*unyw/gamma+1.0
      db3mdc=-2.0/gamma*unyw

      b4m=hp
      db4mdu=up
      db4mdv=vp
      db4mdc=2.0*cp/gm1

      bm(ind(1,1))=df1mdr+df1mdu*dudq1+df1mdv*dvdq1+df1mdc*dcdq1
      bm(ind(1,2))=df1mdu*dudq2+df1mdc*dcdq2
      bm(ind(1,3))=df1mdv*dvdq3+df1mdc*dcdq3
      bm(ind(1,4))=df1mdc*dcdq4

      bm(ind(2,1))=b2m*bm(ind(1,1))+f1m*(db2mdu*dudq1+db2mdv*dvdq1 &
             +db2mdc*dcdq1)
      bm(ind(2,2))=b2m*bm(ind(1,2))+f1m*(db2mdu*dudq2+db2mdc*dcdq2)
      bm(ind(2,3))=b2m*bm(ind(1,3))+f1m*(db2mdv*dvdq3+db2mdc*dcdq3)
      bm(ind(2,4))=b2m*bm(ind(1,4))+f1m*(db2mdc*dcdq4)
```

```
   bm(ind(3,1))=b3m*bm(ind(1,1))+f1m*(db3mdu*dudq1+db3mdv*dvdq1 &
          +db3mdc*dcdq1)
   bm(ind(3,2))=b3m*bm(ind(1,2))+f1m*(db3mdu*dudq2+db3mdc*dcdq2)
   bm(ind(3,3))=b3m*bm(ind(1,3))+f1m*(db3mdv*dvdq3+db3mdc*dcdq3)
   bm(ind(3,4))=b3m*bm(ind(1,4))+f1m*(db3mdc*dcdq4)

   bm(ind(4,1))=b4m*bm(ind(1,1))+f1m*(db4mdu*dudq1+db4mdv*dvdq1 &
          +db4mdc*dcdq1)
   bm(ind(4,2))=b4m*bm(ind(1,2))+f1m*(db4mdu*dudq2+db4mdc*dcdq2)
   bm(ind(4,3))=b4m*bm(ind(1,3))+f1m*(db4mdv*dvdq3+db4mdc*dcdq3)
   bm(ind(4,4))=b4m*bm(ind(1,4))+f1m*(db4mdc*dcdq4)
  elseif (met.le.-clone) then
   unxw=arew*unxw
   unyw=arew*unyw
   vbp=arew*vbp

   bm(ind(1,1))=0.0
   bm(ind(1,2))=unxw
   bm(ind(1,3))=unyw
   bm(ind(1,4))=0.0

   bm(ind(2,1))=unxw*gqsqp-up*vbp
   bm(ind(2,2))=vbp-gm2*unxw*up
   bm(ind(2,3))=unyw*up-gm1*unxw*vp
   bm(ind(2,4))=unxw*gm1

   bm(ind(3,1))=unyw*gqsqp-vp*vbp
   bm(ind(3,2))=unxw*vp-gm1*unyw*up
   bm(ind(3,3))=vbp-gm2*unyw*vp
   bm(ind(3,4))=unyw*gm1

   bm(ind(4,1))=vbp*(gqsqp-hp)
   bm(ind(4,2))=unxw*hp-gm1*up*vbp
   bm(ind(4,3))=unyw*hp-gm1*vp*vbp
   bm(ind(4,4))=gamma*vbp
  else
   do kj=1,4
   do ki=1,4
     bm(ind(ki,kj))=0.0
   enddo
   enddo
  endif
 elseif (iflux.eq.2) then
!
! evaluate the flux jacobian based on eigen-value (steger-warming) flux-vector
! splitting scheme
!
   qsqm=0.5*(um*um+vm*vm)
   qsqp=0.5*(up*up+vp*vp)
   gqsqm=gm1*qsqm
   gqsqp=gm1*qsqp
!
! calculate bp based on fp
!
   cm=ssqrt(gamma*rgas*tm)
   met=vbm/cm

   if (met.le.-clone) then
    do kj=1,4
    do ki=1,4
      bp(ind(ki,kj))=0.0
    enddo
```

```
  enddo
elseif (met.gt.-clone .and. met.le.zero) then
  dudq1=-um/rm
  dudq2=1.0/rm

  dvdq1=-vm/rm
  dvdq3=1.0/rm

  fact=0.5*gm1*cm/pm
  dcdq1=fact*(-em/rm+um*um+vm*vm)
  dcdq2=-fact*um
  dcdq3=-fact*vm
  dcdq4=fact

  ev3=vbm+cm
  ev3=(ev3+ssqrt(ev3*ev3+del1))/2.0
  fact=arew*ev3*rm/2.0/gamma
  dfdq1=arew*(cm+rm*dcdq1)/2.0/gamma
  dfdq2=arew*(unxw+rm*dcdq2)/2.0/gamma
  dfdq3=arew*(unyw+rm*dcdq3)/2.0/gamma
  dfdq4=arew*(rm*dcdq4)/2.0/gamma

  term1=um+cm*unxw
  term2=vm+cm*unyw
  term3=vbm*cm+qsqm+cm*cm/gm1

  bp(ind(1,1))=dfdq1
  bp(ind(1,2))=dfdq2
  bp(ind(1,3))=dfdq3
  bp(ind(1,4))=dfdq4

  bp(ind(2,1))=fact*(dudq1+dcdq1*unxw)+term1*dfdq1
  bp(ind(2,2))=fact*(dudq2+dcdq2*unxw)+term1*dfdq2
  bp(ind(2,3))=fact*dcdq3*unxw+term1*dfdq3
  bp(ind(2,4))=fact*dcdq4*unxw+term1*dfdq4

  bp(ind(3,1))=fact*(dvdq1+dcdq1*unyw)+term2*dfdq1
  bp(ind(3,2))=fact*dcdq2*unyw+term2*dfdq2
  bp(ind(3,3))=fact*(dvdq3+dcdq3*unyw)+term2*dfdq3
  bp(ind(3,4))=fact*dcdq4*unyw+term2*dfdq4

  bp(ind(4,1))=fact*((dudq1*unxw+dvdq1*unyw)*cm+vbm*dcdq1+um*dudq1 &
         +vm*dvdq1+2.0*cm*dcdq1/gm1)+term3*dfdq1
  bp(ind(4,2))=fact*(dudq2*unxw*cm+vbm*dcdq2+um*dudq2+2.0*cm*dcdq2/gm1) &
         +term3*dfdq2
  bp(ind(4,3))=fact*(dvdq3*unyw*cm+vbm*dcdq3+vm*dvdq3+2.0*cm*dcdq3/gm1) &
         +term3*dfdq3
  bp(ind(4,4))=fact*(vbm*dcdq4+2.0*cm*dcdq4/gm1)+term3*dfdq4
elseif (met.gt.zero .and. met.lt.clone) then
  unxw=arew*unxw
  unyw=arew*unyw
  vbm=arew*vbm

  bp(ind(1,1))=0.0
  bp(ind(1,2))=unxw
  bp(ind(1,3))=unyw
  bp(ind(1,4))=0.0

  bp(ind(2,1))=unxw*gqsqm-um*vbm
  bp(ind(2,2))=vbm-gm2*unxw*um
  bp(ind(2,3))=unyw*um-gm1*unxw*vm
  bp(ind(2,4))=unxw*gm1
```

```
bp(ind(3,1))=unyw*gqsqm-vm*vbm
bp(ind(3,2))=unxw*vm-gm1*unyw*um
bp(ind(3,3))=vbm-gm2*unyw*vm
bp(ind(3,4))=unyw*gm1

bp(ind(4,1))=vbm*(gqsqm-hm)
bp(ind(4,2))=unxw*hm-gm1*um*vbm
bp(ind(4,3))=unyw*hm-gm1*vm*vbm
bp(ind(4,4))=gamma*vbm

unxw=unxt(ij)
unyw=unyt(ij)
vbm=unxw*um+unyw*vm

dudq1=-um/rm
dudq2=1.0/rm

dvdq1=-vm/rm
dvdq3=1.0/rm

fact=0.5*gm1*cm/pm
dcdq1=fact*(-em/rm+um*um+vm*vm)
dcdq2=-fact*um
dcdq3=-fact*vm
dcdq4=fact

ev1=vbm-cm
ev1=(ev1-ssqrt(ev1*ev1+del1))/2.0
fact=arew*ev1*rm/2.0/gamma
dfdq1=-arew*(cm+rm*dcdq1)/2.0/gamma
dfdq2=arew*(unxw-rm*dcdq2)/2.0/gamma
dfdq3=arew*(unyw-rm*dcdq3)/2.0/gamma
dfdq4=-arew*(rm*dcdq4)/2.0/gamma

term1=um-cm*unxw
term2=vm-cm*unyw
term3=-vbm*cm+qsqm+cm*cm/gm1

bp(ind(1,1))=bp(ind(1,1))-dfdq1
bp(ind(1,2))=bp(ind(1,2))-dfdq2
bp(ind(1,3))=bp(ind(1,3))-dfdq3
bp(ind(1,4))=bp(ind(1,4))-dfdq4

bp(ind(2,1))=bp(ind(2,1))-(fact*(dudq1-dcdq1*unxw)+term1*dfdq1)
bp(ind(2,2))=bp(ind(2,2))-(fact*(dudq2-dcdq2*unxw)+term1*dfdq2)
bp(ind(2,3))=bp(ind(2,3))-(-fact*dcdq3*unxw+term1*dfdq3)
bp(ind(2,4))=bp(ind(2,4))-(-fact*dcdq4*unxw+term1*dfdq4)

bp(ind(3,1))=bp(ind(3,1))-(fact*(dvdq1-dcdq1*unyw)+term2*dfdq1)
bp(ind(3,2))=bp(ind(3,2))-(-fact*dcdq2*unyw+term2*dfdq2)
bp(ind(3,3))=bp(ind(3,3))-(fact*(dvdq3-dcdq3*unyw)+term2*dfdq3)
bp(ind(3,4))=bp(ind(3,4))-(-fact*dcdq4*unyw+term2*dfdq4)

bp(ind(4,1))=bp(ind(4,1))-(fact*(-(dudq1*unxw+dvdq1*unyw)*cm &
        -vbm*dcdq1+um*dudq1+vm*dvdq1+2.0*cm*dcdq1/gm1)+term3*dfdq1)
bp(ind(4,2))=bp(ind(4,2))-(fact*(-dudq2*unxw*cm-vbm*dcdq2+um*dudq2 &
        +2.0*cm*dcdq2/gm1)+term3*dfdq2)
bp(ind(4,3))=bp(ind(4,3))-(fact*(-dvdq3*unyw*cm-vbm*dcdq3+vm*dvdq3 &
        +2.0*cm*dcdq3/gm1)+term3*dfdq3)
bp(ind(4,4))=bp(ind(4,4))-(fact*(-vbm*dcdq4+2.0*cm*dcdq4/gm1) &
        +term3*dfdq4)
```

```
    else
      unxw=arew*unxw
      unyw=arew*unyw
      vbm=arew*vbm

      bp(ind(1,1))=0.0
      bp(ind(1,2))=unxw
      bp(ind(1,3))=unyw
      bp(ind(1,4))=0.0

      bp(ind(2,1))=unxw*gqsqm-um*vbm
      bp(ind(2,2))=vbm-gm2*unxw*um
      bp(ind(2,3))=unyw*um-gm1*unxw*vm
      bp(ind(2,4))=unxw*gm1

      bp(ind(3,1))=unyw*gqsqm-vm*vbm
      bp(ind(3,2))=unxw*vm-gm1*unyw*um
      bp(ind(3,3))=vbm-gm2*unyw*vm
      bp(ind(3,4))=unyw*gm1

      bp(ind(4,1))=vbm*(gqsqm-hm)
      bp(ind(4,2))=unxw*hm-gm1*um*vbm
      bp(ind(4,3))=unyw*hm-gm1*vm*vbm
      bp(ind(4,4))=gamma*vbm

      unxw=unxt(ij)
      unyw=unyt(ij)
    endif
!
! calculate bm based on fm
!
    cp=ssqrt(gamma*rgas*tp)
    met=vbp/cp
    if (met.le.-clone) then
      unxw=arew*unxw
      unyw=arew*unyw
      vbp=arew*vbp

      bm(ind(1,1))=0.0
      bm(ind(1,2))=unxw
      bm(ind(1,3))=unyw
      bm(ind(1,4))=0.0

      bm(ind(2,1))=unxw*gqsqp-up*vbp
      bm(ind(2,2))=vbp-gm2*unxw*up
      bm(ind(2,3))=unyw*up-gm1*unxw*vp
      bm(ind(2,4))=unxw*gm1

      bm(ind(3,1))=unyw*gqsqp-vp*vbp
      bm(ind(3,2))=unxw*vp-gm1*unyw*up
      bm(ind(3,3))=vbp-gm2*unyw*vp
      bm(ind(3,4))=unyw*gm1

      bm(ind(4,1))=vbp*(gqsqp-hp)
      bm(ind(4,2))=unxw*hp-gm1*up*vbp
      bm(ind(4,3))=unyw*hp-gm1*vp*vbp
      bm(ind(4,4))=gamma*vbp
    elseif (met.gt.-clone .and. met.le.zero) then
      unxw=arew*unxw
      unyw=arew*unyw
      vbp=arew*vbp
```

```
bm(ind(1,1))=0.0
bm(ind(1,2))=unxw
bm(ind(1,3))=unyw
bm(ind(1,4))=0.0

bm(ind(2,1))=unxw*gqsqp-up*vbp
bm(ind(2,2))=vbp-gm2*unxw*up
bm(ind(2,3))=unyw*up-gm1*unxw*vp
bm(ind(2,4))=unxw*gm1

bm(ind(3,1))=unyw*gqsqp-vp*vbp
bm(ind(3,2))=unxw*vp-gm1*unyw*up
bm(ind(3,3))=vbp-gm2*unyw*vp
bm(ind(3,4))=unyw*gm1

bm(ind(4,1))=vbp*(gqsqp-hp)
bm(ind(4,2))=unxw*hp-gm1*up*vbp
bm(ind(4,3))=unyw*hp-gm1*vp*vbp
bm(ind(4,4))=gamma*vbp

unxw=unxt(ij)
unyw=unyt(ij)
vbp=unxw*up+unyw*vp

dudq1=-up/rp
dudq2=1.0/rp

dvdq1=-vp/rp
dvdq3=1.0/rp

fact=0.5*gm1*cp/pp
dcdq1=fact*(-ep/rp+up*up+vp*vp)
dcdq2=-fact*up
dcdq3=-fact*vp
dcdq4=fact

ev3=vbp+cp
ev3=(ev3+ssqrt(ev3*ev3+del1))/2.0
fact=ev3*rp/2.0/gamma
dfdq1=arew*(cp+rp*dcdq1)/2.0/gamma
dfdq2=arew*(unxw+rp*dcdq2)/2.0/gamma
dfdq3=arew*(unyw+rp*dcdq3)/2.0/gamma
dfdq4=arew*(rp*dcdq4)/2.0/gamma

term1=up+cp*unxw
term2=vp+cp*unyw
term3=vbp*cp+qsqp+cp*cp/gm1

bm(ind(1,1))=bm(ind(1,1))-dfdq1
bm(ind(1,2))=bm(ind(1,2))-dfdq2
bm(ind(1,3))=bm(ind(1,3))-dfdq3
bm(ind(1,4))=bm(ind(1,4))-dfdq4

bm(ind(2,1))=bm(ind(2,1))-(fact*(dudq1+dcdq1*unxw)+term1*dfdq1)
bm(ind(2,2))=bm(ind(2,2))-(fact*(dudq2+dcdq2*unxw)+term1*dfdq2)
bm(ind(2,3))=bm(ind(2,3))-(fact*dcdq3*unxw+term1*dfdq3)
bm(ind(2,4))=bm(ind(2,4))-(fact*dcdq4*unxw+term1*dfdq4)

bm(ind(3,1))=bm(ind(3,1))-(fact*(dvdq1+dcdq1*unyw)+term2*dfdq1)
bm(ind(3,2))=bm(ind(3,2))-(fact*dcdq2*unyw+term2*dfdq2)
bm(ind(3,3))=bm(ind(3,3))-(fact*(dvdq3+dcdq3*unyw)+term2*dfdq3)
bm(ind(3,4))=bm(ind(3,4))-(fact*dcdq4*unyw+term2*dfdq4)
```

```
  bm(ind(4,1))=bm(ind(4,1))-(fact*((dudq1*unxw+dvdq1*unyw)*cp+vbp*dcdq1 &
          +up*dudq1+vp*dvdq1+2.0*cp*dcdq1/gm1)+term3*dfdq1)
  bm(ind(4,2))=bm(ind(4,2))-(fact*(dudq2*unxw*cp+vbp*dcdq2+up*dudq2 &
          +2.0*cp*dcdq2/gm1)+term3*dfdq2)
  bm(ind(4,3))=bm(ind(4,3))-(fact*(dvdq3*unyw*cp+vbp*dcdq3+vp*dvdq3 &
          +2.0*cp*dcdq3/gm1)+term3*dfdq3)
  bm(ind(4,4))=bm(ind(4,4))-(fact*(vbp*dcdq4+2.0*cp*dcdq4/gm1) &
          +term3*dfdq4)
elseif (met.gt.zero .and. met.lt.clone) then
  dudq1=-up/rp
  dudq2=1.0/rp

  dvdq1=-vp/rp
  dvdq3=1.0/rp

  fact=0.5*gm1*cp/pp
  dcdq1=fact*(-ep/rp+up*up+vp*vp)
  dcdq2=-fact*up
  dcdq3=-fact*vp
  dcdq4=fact

  ev1=vbp-cp
  ev1=(ev1-ssqrt(ev1*ev1+del1))/2.0
  fact=arew*ev1*rp/2.0/gamma
  dfdq1=-arew*(cp+rp*dcdq1)/2.0/gamma
  dfdq2=arew*(unxw-rp*dcdq2)/2.0/gamma
  dfdq3=arew*(unyw-rp*dcdq3)/2.0/gamma
  dfdq4=-arew*(rp*dcdq4)/2.0/gamma

  term1=up-cp*unxw
  term2=vp-cp*unyw
  term3=-vbp*cp+qsqp+cp*cp/gm1

  bm(ind(1,1))=dfdq1
  bm(ind(1,2))=dfdq2
  bm(ind(1,3))=dfdq3
  bm(ind(1,4))=dfdq4

  bm(ind(2,1))=fact*(dudq1-dcdq1*unxw)+term1*dfdq1
  bm(ind(2,2))=fact*(dudq2-dcdq2*unxw)+term1*dfdq2
  bm(ind(2,3))=-fact*dcdq3*unxw+term1*dfdq3
  bm(ind(2,4))=-fact*dcdq4*unxw+term1*dfdq4

  bm(ind(3,1))=fact*(dvdq1-dcdq1*unyw)+term2*dfdq1
  bm(ind(3,2))=-fact*dcdq2*unyw+term2*dfdq2
  bm(ind(3,3))=fact*(dvdq3-dcdq3*unyw)+term2*dfdq3
  bm(ind(3,4))=-fact*dcdq4*unyw+term2*dfdq4

  bm(ind(4,1))=fact*(-(dudq1*unxw+dvdq1*unyw)*cp-vbp*dcdq1+up*dudq1 &
          +vp*dvdq1+2.0*cp*dcdq1/gm1)+term3*dfdq1
  bm(ind(4,2))=fact*(-dudq2*unxw*cp-vbp*dcdq2+up*dudq2 &
          +2.0*cp*dcdq2/gm1)+term3*dfdq2
  bm(ind(4,3))=fact*(-dvdq3*unyw*cp-vbp*dcdq3+vp*dvdq3 &
          +2.0*cp*dcdq3/gm1)+term3*dfdq3
  bm(ind(4,4))=fact*(-vbp*dcdq4+2.0*cp*dcdq4/gm1)+term3*dfdq4
else
  do kj=1,4
  do ki=1,4
    bm(ind(ki,kj))=0.0
  enddo
  enddo
```

```
    endif
  elseif (iflux.eq.3) then
!
! evaluate the flux jacobian based on roe's flux-difference splitting scheme
!
    gqsqm=0.5*gm1*(um*um+vm*vm)
    gqsqp=0.5*gm1*(up*up+vp*vp)
    hare=0.5*arew
    unxw=hare*unxw
    unyw=hare*unyw
    vbm=hare*vbm
    vbp=hare*vbp
!
! calculate bp based on fp
!
    bp(ind(1,1))=0.0
    bp(ind(1,2))=unxw
    bp(ind(1,3))=unyw
    bp(ind(1,4))=0.0

    bp(ind(2,1))=unxw*gqsqm-um*vbm
    bp(ind(2,2))=vbm-gm2*unxw*um
    bp(ind(2,3))=unyw*um-gm1*unxw*vm
    bp(ind(2,4))=unxw*gm1

    bp(ind(3,1))=unyw*gqsqm-vm*vbm
    bp(ind(3,2))=unxw*vm-gm1*unyw*um
    bp(ind(3,3))=vbm-gm2*unyw*vm
    bp(ind(3,4))=unyw*gm1

    bp(ind(4,1))=vbm*(gqsqm-hm)
    bp(ind(4,2))=unxw*hm-gm1*um*vbm
    bp(ind(4,3))=unyw*hm-gm1*vm*vbm
    bp(ind(4,4))=gamma*vbm
!
! calculate bm based on fm
!
    bm(ind(1,1))=0.0
    bm(ind(1,2))=unxw
    bm(ind(1,3))=unyw
    bm(ind(1,4))=0.0

    bm(ind(2,1))=unxw*gqsqp-up*vbp
    bm(ind(2,2))=vbp-gm2*unxw*up
    bm(ind(2,3))=unyw*up-gm1*unxw*vp
    bm(ind(2,4))=unxw*gm1

    bm(ind(3,1))=unyw*gqsqp-vp*vbp
    bm(ind(3,2))=unxw*vp-gm1*unyw*up
    bm(ind(3,3))=vbp-gm2*unyw*vp
    bm(ind(3,4))=unyw*gm1

    bm(ind(4,1))=vbp*(gqsqp-hp)
    bm(ind(4,2))=unxw*hp-gm1*up*vbp
    bm(ind(4,3))=unyw*hp-gm1*vp*vbp
    bm(ind(4,4))=gamma*vbp
  endif

  return
  end

  subroutine caldev
```

```
!
!*******************************************************************************
! form the jacobian matrix dev/dq of the viscous flux in xi-direction at right
! wall of cell (i,j).
!*******************************************************************************
!
 include 'params.cmd'
 include 'precs.cmd'
 include 'arrays.cmd'
 include 'param.cmd'
 include 'bcon.cmd'

 real*8 kur

 dimension devm(4,4), devp(4,4)

 ind(ki,kj)=ki+(kj-1)*4+(ij-1)*16

 if (invisd.gt.0) return
 if (j.eq.1 .or. j.eq.jmax) return
 if (i.eq.1 .and. wtlb(j).ne.2) return
 if (i.eq.im1 .and. wtrb(j).ne.2) return

 do kj=1,4
 do ki=1,4
   devp(ki,kj)=0.0
   devm(ki,kj)=0.0
 enddo
 enddo

 ipj=ij+1
 ijp=ij+imax
 ipjp=ijp+1
 ijm=ij-imax
 ipjm=ijm+1

 mur=2.0*mu(ij)*mu(ipj)/(mu(ij)+mu(ipj))
 kur=2.0*ku(ij)*ku(ipj)/(ku(ij)+ku(ipj))
 uc=u(ij)
 vc=v(ij)
 ur=(u(ij)+u(ipj))/2.0
 vr=(v(ij)+v(ipj))/2.0
 urr=u(ipj)
 vrr=v(ipj)
 urt=(u(ijp)+u(ipjp))/2.0
 vrt=(v(ijp)+v(ipjp))/2.0
 urbs=(u(ijm)+u(ipjm))/2.0
 vrbs=(v(ijm)+v(ipjm))/2.0
 dudx=(urr-uc)
 dude=(urt urbs)/2.0
 dvdx=(vrr-vc)
 dvde=(vrt-vrbs)/2.0

 ev2=mur*(a1(ij)*dudx+a5(ij)*dude+a2(ij)*dvdx+a6(ij)*dvde+a7(ij)*vr)
 ev3=mur*(a2(ij)*dudx+a8(ij)*dude+a3(ij)*dvdx+a9(ij)*dvde+a10(ij)*vr)

 dudq1p=-u(ij)/q1(ij)
 dudq2p=1.0/q1(ij)
 dvdq1p=-v(ij)/q1(ij)
 dvdq3p=1.0/q1(ij)
 dpdq1=0.5*gm1*(u(ij)*u(ij)+v(ij)*v(ij))
 dpdq2=-gm1*u(ij)
```

```
dpdq3=-gm1*v(ij)
dpdq4=gm1
dtdq1p=(dpdq1-p(ij)/q1(ij))/q1(ij)/rgas
dtdq2p=dpdq2/q1(ij)/rgas
dtdq3p=dpdq3/q1(ij)/rgas
dtdq4p=dpdq4/q1(ij)/rgas

dudq1m=-u(ipj)/q1(ipj)
dudq2m=1.0/q1(ipj)
dvdq1m=-v(ipj)/q1(ipj)
dvdq3m=1.0/q1(ipj)
dpdq1=0.5*gm1*(u(ipj)*u(ipj)+v(ipj)*v(ipj))
dpdq2=-gm1*u(ipj)
dpdq3=-gm1*v(ipj)
dpdq4=gm1
dtdq1m=(dpdq1-p(ipj)/q1(ipj))/q1(ipj)/rgas
dtdq2m=dpdq2/q1(ipj)/rgas
dtdq3m=dpdq3/q1(ipj)/rgas
dtdq4m=dpdq4/q1(ipj)/rgas

devp(2,1)=-mur*(a1(ij)*dudq1p+a2(ij)*dvdq1p-a7(ij)*dvdq1p/2.0)
devp(2,2)=-mur*a1(ij)*dudq2p
devp(2,3)=-mur*(a2(ij)*dvdq3p-a7(ij)*dvdq3p/2.0)

devp(3,1)=-mur*(a2(ij)*dudq1p+a3(ij)*dvdq1p-a10(ij)*dvdq1p/2.0)
devp(3,2)=-mur*a2(ij)*dudq2p
devp(3,3)=-mur*(a3(ij)*dvdq3p-a10(ij)*dvdq3p/2.0)

devp(4,1)=-kur*a4(ij)*dtdq1p+ev2*dudq1p/2.0+ur*devp(2,1)+ev3*dvdq1p/2.0 &
        +vr*devp(3,1)
devp(4,2)=-kur*a4(ij)*dtdq2p+ev2*dudq2p/2.0+ur*devp(2,2)+vr*devp(3,2)
devp(4,3)=-kur*a4(ij)*dtdq3p+ur*devp(2,3)+ev3*dvdq3p/2.0+vr*devp(3,3)
devp(4,4)=-kur*a4(ij)*dtdq4p

devm(2,1)=mur*(a1(ij)*dudq1m+a2(ij)*dvdq1m+a7(ij)*dvdq1m/2.0)
devm(2,2)=mur*a1(ij)*dudq2m
devm(2,3)=mur*(a2(ij)*dvdq3m+a7(ij)*dvdq3m/2.0)

devm(3,1)=mur*(a2(ij)*dudq1m+a3(ij)*dvdq1m+a10(ij)*dvdq1m/2.0)
devm(3,2)=mur*a2(ij)*dudq2m
devm(3,3)=mur*(a3(ij)*dvdq3m+a10(ij)*dvdq3m/2.0)

devm(4,1)=kur*a4(ij)*dtdq1m+ev2*dudq1m/2.0+ur*devm(2,1)+ev3*dvdq1m/2.0 &
        +vr*devm(3,1)
devm(4,2)=kur*a4(ij)*dtdq2m+ev2*dudq2m/2.0+ur*devm(2,2)+vr*devm(3,2)
devm(4,3)=kur*a4(ij)*dtdq3m+ur*devm(2,3)+ev3*dvdq3m/2.0+vr*devm(3,3)
devm(4,4)=kur*a4(ij)*dtdq4m

do kj=1,4
do ki=1,4
  ap(ind(ki,kj))=ap(ind(ki,kj))-devp(ki,kj)
  am(ind(ki,kj))=am(ind(ki,kj))-devm(ki,kj)
enddo
enddo

return
end

subroutine caldei
!
!****************************************************************************
! form the jacobian matrices a+ = de+/dq and a- = de-/dq for the split flux in
```

```
! xi-direction at specified cell wall based on a variety of schemes.
!
! iflux=1 does the van leer scheme
! iflux=2 does the steger-warming scheme
! iflux=3 does the roe scheme
!******************************************************************************
!
  include 'params.cmd'
  include 'precs.cmd'
  include 'arrays.cmd'
  include 'param.cmd'
  include 'bcon.cmd'

  data del1 /1.0d-10/

  ind(ki,kj)=ki+(kj-1)*4+(ij-1)*16

  if (j.eq.1 .or. j.eq.jmax) return

  unxw=unxr(ij)
  unyw=unyr(ij)
  arew=arer(ij)
!
! find cell wall interpolated values
!
  call calqpm ('r',inorxi)

  ubm=unxw*um+unyw*vm
  ubp=unxw*up+unyw*vp

  if ((i.eq.1 .and. wtlb(j).le.2) .or. (i.eq.im1 .and. wtrb(j).le.2)) then
    ubm=dampi*ubm
    ubp=dampi*ubp
  endif

  if (iflux.eq.1) then
!
! evaluate the flux jacobian based on van leer's flux-vector splitting scheme
!
    qsqm=0.5*(um*um+vm*vm)
    qsqp=0.5*(up*up+vp*vp)
    gqsqm=gm1*qsqm
    gqsqp=gm1*qsqp
!
! calculate ap based on ep
!
    cm=ssqrt(gamma*rgas*tm)
    mxi=ubm/cm

    if (sabs(mxi).lt.clone) then
      mxip=mxi+1.0
      e1p=arew*0.25*rm*cm*mxip*mxip
      de1pdr=e1p/rm
      de1pdu=arew*0.5*rm*unxw*mxip
      de1pdv=arew*0.5*rm*unyw*mxip
      de1pdc=arew*0.25*rm*(1.0-mxi)*mxip

      dudq1=-um/rm
      dudq2=1.0/rm

      dvdq1=-vm/rm
      dvdq3=1.0/rm
```

```
  fact=0.5*gm1*cm/pm
  dcdq1=fact*(-em/rm+um*um+vm*vm)
  dcdq2=-fact*um
  dcdq3=-fact*vm
  dcdq4=fact

  a2p=unxw/gamma*(-ubm+2.0*cm)+um
  da2pdu=-unxw*unxw/gamma+1.0
  da2pdv=-unxw*unyw/gamma
  da2pdc=2.0/gamma*unxw

  a3p=unyw/gamma*(-ubm+2.0*cm)+vm
  da3pdu=-unxw*unyw/gamma
  da3pdv=-unyw*unyw/gamma+1.0
  da3pdc=2.0/gamma*unyw

  a4p=hm
  da4pdu=um
  da4pdv=vm
  da4pdc=2.0*cm/gm1

  ap(ind(1,1))=de1pdr+de1pdu*dudq1+de1pdv*dvdq1+de1pdc*dcdq1
  ap(ind(1,2))=de1pdu*dudq2+de1pdc*dcdq2
  ap(ind(1,3))=de1pdv*dvdq3+de1pdc*dcdq3
  ap(ind(1,4))=de1pdc*dcdq4

  ap(ind(2,1))=a2p*ap(ind(1,1))+e1p*(da2pdu*dudq1+da2pdv*dvdq1 &
         +da2pdc*dcdq1)
  ap(ind(2,2))=a2p*ap(ind(1,2))+e1p*(da2pdu*dudq2+da2pdc*dcdq2)
  ap(ind(2,3))=a2p*ap(ind(1,3))+e1p*(da2pdv*dvdq3+da2pdc*dcdq3)
  ap(ind(2,4))=a2p*ap(ind(1,4))+e1p*(da2pdc*dcdq4)

  ap(ind(3,1))=a3p*ap(ind(1,1))+e1p*(da3pdu*dudq1+da3pdv*dvdq1 &
         +da3pdc*dcdq1)
  ap(ind(3,2))=a3p*ap(ind(1,2))+e1p*(da3pdu*dudq2+da3pdc*dcdq2)
  ap(ind(3,3))=a3p*ap(ind(1,3))+e1p*(da3pdv*dvdq3+da3pdc*dcdq3)
  ap(ind(3,4))=a3p*ap(ind(1,4))+e1p*(da3pdc*dcdq4)

  ap(ind(4,1))=a4p*ap(ind(1,1))+e1p*(da4pdu*dudq1+da4pdv*dvdq1 &
         +da4pdc*dcdq1)
  ap(ind(4,2))=a4p*ap(ind(1,2))+e1p*(da4pdu*dudq2+da4pdc*dcdq2)
  ap(ind(4,3))=a4p*ap(ind(1,3))+e1p*(da4pdv*dvdq3+da4pdc*dcdq3)
  ap(ind(4,4))=a4p*ap(ind(1,4))+e1p*(da4pdc*dcdq4)
elseif (mxi.ge.clone) then
  unxw=arew*unxw
  unyw=arew*unyw
  ubm=arew*ubm

  ap(ind(1,1))=0.0
  ap(ind(1,2))=unxw
  ap(ind(1,3))=unyw
  ap(ind(1,4))=0.0

  ap(ind(2,1))=unxw*gqsqm-um*ubm
  ap(ind(2,2))=ubm-gm2*unxw*um
  ap(ind(2,3))=unyw*um-gm1*unxw*vm
  ap(ind(2,4))=unxw*gm1

  ap(ind(3,1))=unyw*gqsqm-vm*ubm
  ap(ind(3,2))=unxw*vm-gm1*unyw*um
  ap(ind(3,3))=ubm-gm2*unyw*vm
```

```
      ap(ind(3,4))=unyw*gm1

      ap(ind(4,1))=ubm*(gqsqm-hm)
      ap(ind(4,2))=unxw*hm-gm1*um*ubm
      ap(ind(4,3))=unyw*hm-gm1*vm*ubm
      ap(ind(4,4))=gamma*ubm

      unxw=unxr(ij)
      unyw=unyr(ij)
    else
      do kj=1,4
      do ki=1,4
        ap(ind(ki,kj))=0.0
      enddo
      enddo
    endif
!
! calculate am based on em
!
    cp=ssqrt(gamma*rgas*tp)
    mxi=ubp/cp

    if (sabs(mxi).lt.clone) then
      mxim=mxi-1.0
      e1m=-arew*0.25*rp*cp*mxim*mxim
      de1mdr=e1m/rp
      de1mdu=-arew*0.5*rp*unxw*mxim
      de1mdv=-arew*0.5*rp*unyw*mxim
      de1mdc=-arew*0.25*rp*(1.0-mxi)*(1.0+mxi)

      dudq1=-up/rp
      dudq2=1.0/rp

      dvdq1=-vp/rp
      dvdq3=1.0/rp

      fact=0.5*gm1*cp/pp
      dcdq1=fact*(-ep/rp+up*up+vp*vp)
      dcdq2=-fact*up
      dcdq3=-fact*vp
      dcdq4=fact

      a2m=unxw/gamma*(-ubp-2.0*cp)+up
      da2mdu=-unxw*unxw/gamma+1.0
      da2mdv=-unxw*unyw/gamma
      da2mdc=-2.0/gamma*unxw

      a3m=unyw/gamma*(-ubp-2.0*cp)+vp
      da3mdu=-unxw*unyw/gamma
      da3mdv=-unyw*unyw/gamma+1.0
      da3mdc=-2.0/gamma*unyw

      a4m=hp
      da4mdu=up
      da4mdv=vp
      da4mdc=2.0*cp/gm1

      am(ind(1,1))=de1mdr+de1mdu*dudq1+de1mdv*dvdq1+de1mdc*dcdq1
      am(ind(1,2))=de1mdu*dudq2+de1mdc*dcdq2
      am(ind(1,3))=de1mdv*dvdq3+de1mdc*dcdq3
      am(ind(1,4))=de1mdc*dcdq4
```

```
      am(ind(2,1))=a2m*am(ind(1,1))+elm*(da2mdu*dudq1+da2mdv*dvdq1 &
              +da2mdc*dcdq1)
      am(ind(2,2))=a2m*am(ind(1,2))+elm*(da2mdu*dudq2+da2mdc*dcdq2)
      am(ind(2,3))=a2m*am(ind(1,3))+elm*(da2mdv*dvdq3+da2mdc*dcdq3)
      am(ind(2,4))=a2m*am(ind(1,4))+elm*(da2mdc*dcdq4)

      am(ind(3,1))=a3m*am(ind(1,1))+elm*(da3mdu*dudq1+da3mdv*dvdq1 &
              +da3mdc*dcdq1)
      am(ind(3,2))=a3m*am(ind(1,2))+elm*(da3mdu*dudq2+da3mdc*dcdq2)
      am(ind(3,3))=a3m*am(ind(1,3))+elm*(da3mdv*dvdq3+da3mdc*dcdq3)
      am(ind(3,4))=a3m*am(ind(1,4))+elm*(da3mdc*dcdq4)

      am(ind(4,1))=a4m*am(ind(1,1))+elm*(da4mdu*dudq1+da4mdv*dvdq1 &
              +da4mdc*dcdq1)
      am(ind(4,2))=a4m*am(ind(1,2))+elm*(da4mdu*dudq2+da4mdc*dcdq2)
      am(ind(4,3))=a4m*am(ind(1,3))+elm*(da4mdv*dvdq3+da4mdc*dcdq3)
      am(ind(4,4))=a4m*am(ind(1,4))+elm*(da4mdc*dcdq4)
    elseif (mxi.le.-clone) then
      unxw=arew*unxw
      unyw=arew*unyw
      ubp=arew*ubp

      am(ind(1,1))=0.0
      am(ind(1,2))=unxw
      am(ind(1,3))=unyw
      am(ind(1,4))=0.0

      am(ind(2,1))=unxw*gqsqp-up*ubp
      am(ind(2,2))=ubp-gm2*unxw*up
      am(ind(2,3))=unyw*up-gm1*unxw*vp
      am(ind(2,4))=unxw*gm1

      am(ind(3,1))=unyw*gqsqp-vp*ubp
      am(ind(3,2))=unxw*vp-gm1*unyw*up
      am(ind(3,3))=ubp-gm2*unyw*vp
      am(ind(3,4))=unyw*gm1

      am(ind(4,1))=ubp*(gqsqp-hp)
      am(ind(4,2))=unxw*hp-gm1*up*ubp
      am(ind(4,3))=unyw*hp-gm1*vp*ubp
      am(ind(4,4))=gamma*ubp
    else
      do kj=1,4
      do ki=1,4
        am(ind(ki,kj))=0.0
      enddo
      enddo
    endif
  elseif (iflux.eq.2) then
!
! evaluate the flux jacobian based on eigen-value
! (steger-warming) flux-vector splitting scheme
!
    qsqm=0.5*(um*um+vm*vm)
    qsqp=0.5*(up*up+vp*vp)
    gqsqm=gm1*qsqm
    gqsqp=gm1*qsqp
!
! calculate ap based on ep
!
    cm=ssqrt(gamma*rgas*tm)
    mxi=ubm/cm
```

```
 if (mxi.le.-clone) then
   do kj=1,4
   do ki=1,4
     ap(ind(ki,kj))=0.0
   enddo
   enddo
 elseif (mxi.gt.-clone .and. mxi.le.zero) then
   dudq1=-um/rm
   dudq2=1.0/rm

   dvdq1=-vm/rm
   dvdq3=1.0/rm

   fact=0.5*gm1*cm/pm
   dcdq1=fact*(-em/rm+um*um+vm*vm)
   dcdq2=-fact*um
   dcdq3=-fact*vm
   dcdq4=fact

   ev3=ubm+cm
   ev3=(ev3+ssqrt(ev3*ev3+del1))/2.0
   fact=arew*ev3*rm/2.0/gamma
   dfdq1=arew*(cm+rm*dcdq1)/2.0/gamma
   dfdq2=arew*(unxw+rm*dcdq2)/2.0/gamma
   dfdq3=arew*(unyw+rm*dcdq3)/2.0/gamma
   dfdq4=arew*(rm*dcdq4)/2.0/gamma

   term1=um+cm*unxw
   term2=vm+cm*unyw
   term3=ubm*cm+qsqm+cm*cm/gm1

   ap(ind(1,1))=dfdq1
   ap(ind(1,2))=dfdq2
   ap(ind(1,3))=dfdq3
   ap(ind(1,4))=dfdq4

   ap(ind(2,1))=fact*(dudq1+dcdq1*unxw)+term1*dfdq1
   ap(ind(2,2))=fact*(dudq2+dcdq2*unxw)+term1*dfdq2
   ap(ind(2,3))=fact*dcdq3*unxw+term1*dfdq3
   ap(ind(2,4))=fat*dcdq4*unxw+term1*dfdq4

   ap(ind(3,1))=fact*(dvdq1+dcdq1*unyw)+term2*dfdq1
   ap(ind(3,2))=fact*dcdq2*unyw+term2*dfdq2
   ap(ind(3,3))=fact*(dvdq3+dcdq3*unyw)+term2*dfdq3
   ap(ind(3,4))=fact*dcdq4*unyw+term2*dfdq4

   ap(ind(4,1))=fact*((dudq1*unxw+dvdq1*unyw)*cm+ubm*dcdq1+um*dudq1 &
          +vm*dvdq1+2.0*cm*dcdq1/gm1)+term3*dfdq1
   ap(ind(4,2))=fact*(dudq2*unxw*cm+ubm*dcdq2+um*dudq2+2.0*cm*dcdq2/gm1) &
          +term3*dfdq2
   ap(ind(4,3))=fact*(dvdq3*unyw*cm+ubm*dcdq3+vm*dvdq3+2.0*cm*dcdq3/gm1) &
          +term3*dfdq3
   ap(ind(4,4))=fact*(ubm*dcdq4+2.0*cm*dcdq4/gm1)+term3*dfdq4
 elseif (mxi.gt.zero .and. mxi.lt.clone) then
   unxw=arew*unxw
   unyw=arew*unyw
   ubm=arew*ubm

   ap(ind(1,1))=0.0
   ap(ind(1,2))=unxw
   ap(ind(1,3))=unyw
```

```
ap(ind(1,4))=0.0

ap(ind(2,1))=unxw*gqsqm-um*ubm
ap(ind(2,2))=ubm-gm2*unxw*um
ap(ind(2,3))=unyw*um-gm1*unxw*vm
ap(ind(2,4))=unxw*gm1

ap(ind(3,1))=unyw*gqsqm-vm*ubm
ap(ind(3,2))=unxw*vm-gm1*unyw*um
ap(ind(3,3))=ubm-gm2*unyw*vm
ap(ind(3,4))=unyw*gm1

ap(ind(4,1))=ubm*(gqsqm-hm)
ap(ind(4,2))=unxw*hm-gm1*um*ubm
ap(ind(4,3))=unyw*hm-gm1*vm*ubm
ap(ind(4,4))=gamma*ubm

unxw=unxr(ij)
unyw=unyr(ij)
ubm=unxw*um+unyw*vm

dudq1=-um/rm
dudq2=1.0/rm

dvdq1=-vm/rm
dvdq3=1.0/rm

fact=0.5*gm1*cm/pm
dcdq1=fact*(-em/rm+um*um+vm*vm)
dcdq2=-fact*um
dcdq3=-fact*vm
dcdq4=fact

ev1=ubm-cm
ev1=(ev1-ssqrt(ev1*ev1+del1))/2.0
fact=arew*ev1*rm/2.0/gamma
dfdq1=-arew*(cm+rm*dcdq1)/2.0/gamma
dfdq2=arew*(unxw-rm*dcdq2)/2.0/gamma
dfdq3=arew*(unyw-rm*dcdq3)/2.0/gamma
dfdq4=-arew*(rm*dcdq4)/2.0/gamma

term1=um-cm*unxw
term2=vm-cm*unyw
term3=-ubm*cm+qsqm+cm*cm/gm1

ap(ind(1,1))=ap(ind(1,1))-dfdq1
ap(ind(1,2))=ap(ind(1,2))-dfdq2
ap(ind(1,3))=ap(ind(1,3))-dfdq3
ap(ind(1,4))=ap(ind(1,4))-dfdq4

ap(ind(2,1))=ap(ind(2,1))-(fact*(dudq1-dcdq1*unxw)+term1*dfdq1)
ap(ind(2,2))=ap(ind(2,2))-(fact*(dudq2-dcdq2*unxw)+term1*dfdq2)
ap(ind(2,3))=ap(ind(2,3))-(-fact*dcdq3*unxw+term1*dfdq3)
ap(ind(2,4))=ap(ind(2,4))-(-fact*dcdq4*unxw+term1*dfdq4)

ap(ind(3,1))=ap(ind(3,1))-(fact*(dvdq1-dcdq1*unyw)+term2*dfdq1)
ap(ind(3,2))=ap(ind(3,2))-(-fact*dcdq2*unyw+term2*dfdq2)
ap(ind(3,3))=ap(ind(3,3))-(fact*(dvdq3-dcdq3*unyw)+term2*dfdq3)
ap(ind(3,4))=ap(ind(3,4))-(-fact*dcdq4*unyw+term2*dfdq4)

ap(ind(4,1))=ap(ind(4,1))-(fact*(-(dudq1*unxw+dvdq1*unyw)*cm &
      -ubm*dcdq1+um*dudq1+vm*dvdq1+2.0*cm*dcdq1/gm1)+term3*dfdq1)
```

```
    ap(ind(4,2))=ap(ind(4,2))-(fact*(-dudq2*unxw*cm-ubm*dcdq2+um*dudq2 &
            +2.0*cm*dcdq2/gm1)+term3*dfdq2)
    ap(ind(4,3))=ap(ind(4,3))-(fact*(-dvdq3*unyw*cm-ubm*dcdq3+vm*dvdq3 &
            +2.0*cm*dcdq3/gm1)+term3*dfdq3)
    ap(ind(4,4))=ap(ind(4,4))-(fact*(-ubm*dcdq4+2.0*cm*dcdq4/gm1) &
            +term3*dfdq4)
  else
    unxw=arew*unxw
    unyw=arew*unyw
    ubm=arew*ubm

    ap(ind(1,1))=0.0
    ap(ind(1,2))=unxw
    ap(ind(1,3))=unyw
    ap(ind(1,4))=0.0

    ap(ind(2,1))=unxw*gqsqm-um*ubm
    ap(ind(2,2))=ubm-gm2*unxw*um
    ap(ind(2,3))=unyw*um-gm1*unxw*vm
    ap(ind(2,4))=unxw*gm1

    ap(ind(3,1))=unyw*gqsqm-vm*ubm
    ap(ind(3,2))=unxw*vm-gm1*unyw*um
    ap(ind(3,3))=ubm-gm2*unyw*vm
    ap(ind(3,4))=unyw*gm1

    ap(ind(4,1))=ubm*(gqsqm-hm)
    ap(ind(4,2))=unxw*hm-gm1*um*ubm
    ap(ind(4,3))=unyw*hm-gm1*vm*ubm
    ap(ind(4,4))=gamma*ubm

    unxw=unxr(ij)
    unyw=unyr(ij)
  endif
!
! calculate am based on em
!
  cp=ssqrt(gamma*rgas*tp)
  mxi=ubp/cp

  if (mxi.le.-clone) then
    unxw=arew*unxw
    unyw=arew*unyw
    ubp=arew*ubp

    am(ind(1,1))=0.0
    am(ind(1,2))=unxw
    am(ind(1,3))=unyw
    am(ind(1,4))=0.0

    am(ind(2,1))=unxw*gqsqp-up*ubp
    am(ind(2,2))=ubp-gm2*unxw*up
    am(ind(2,3))=unyw*up-gm1*unxw*vp
    am(ind(2,4))=unxw*gm1

    am(ind(3,1))=unyw*gqsqp-vp*ubp
    am(ind(3,2))=unxw*vp-gm1*unyw*up
    am(ind(3,3))=ubp-gm2*unyw*vp
    am(ind(3,4))=unyw*gm1

    am(ind(4,1))=ubp*(gqsqp-hp)
    am(ind(4,2))=unxw*hp-gm1*up*ubp
```

```
 am(ind(4,3))=unyw*hp-gm1*vp*ubp
 am(ind(4,4))=gamma*ubp
elseif (mxi.gt.-clone .and. mxi.le.zero) then
 unxw=arew*unxw
 unyw=arew*unyw
 ubp=arew*ubp

 am(ind(1,1))=0.0
 am(ind(1,2))=unxw
 am(ind(1,3))=unyw
 am(ind(1,4))=0.0

 am(ind(2,1))=unxw*gqsqp-up*ubp
 am(ind(2,2))=ubp-gm2*unxw*up
 am(ind(2,3))=unyw*up-gm1*unxw*vp
 am(ind(2,4))=unxw*gm1

 am(ind(3,1))=unyw*gqsqp-vp*ubp
 am(ind(3,2))=unxw*vp-gm1*unyw*up
 am(ind(3,3))=ubp-gm2*unyw*vp
 am(ind(3,4))=unyw*gm1

 am(ind(4,1))=ubp*(gqsqp-hp)
 am(ind(4,2))=unxw*hp-gm1*up*ubp
 am(ind(4,3))=unyw*hp-gm1*vp*ubp
 am(ind(4,4))=gamma*ubp

 unxw=unxr(ij)
 unyw=unyr(ij)
 ubp=unxw*up+unyw*vp

 dudq1=-up/rp
 dudq2=1.0/rp

 dvdq1=-vp/rp
 dvdq3=1.0/rp

 fact=0.5*gm1*cp/pp
 dcdq1=fact*(-ep/rp+up*up+vp*vp)
 dcdq2=-fact*up
 dcdq3=-fact*vp
 dcdq4=fact

 ev3=ubp+cp
 ev3=(ev3+ssqrt(ev3*ev3+del1))/2.0
 fact=arew*ev3*rp/2.0/gamma
 dfdq1=arew*(cp+rp*dcdq1)/2.0/gamma
 dfdq2=arew*(unxw+rp*dcdq2)/2.0/gamma
 dfdq3=arew*(unyw+rp*dcdq3)/2.0/gamma
 dfdq4=arew*(rp*dcdq4)/2.0/gamma

 term1=up+cp*unxw
 term2=vp+cp*unyw
 term3=ubp*cp+qsqp+cp*cp/gm1

 am(ind(1,1))=am(ind(1,1))-dfdq1
 am(ind(1,2))=am(ind(1,2))-dfdq2
 am(ind(1,3))=am(ind(1,3))-dfdq3
 am(ind(1,4))=am(ind(1,4))-dfdq4

 am(ind(2,1))=am(ind(2,1))-(fact*(dudq1+dcdq1*unxw)+term1*dfdq1)
 am(ind(2,2))=am(ind(2,2))-(fact*(dudq2+dcdq2*unxw)+term1*dfdq2)
```

```
  am(ind(2,3))=am(ind(2,3))-(fact*dcdq3*unxw+term1*dfdq3)
  am(ind(2,4))=am(ind(2,4))-(fact*dcdq4*unxw+term1*dfdq4)

  am(ind(3,1))=am(ind(3,1))-(fact*(dvdq1+dcdq1*unyw)+term2*dfdq1)
  am(ind(3,2))=am(ind(3,2))-(fact*dcdq2*unyw+term2*dfdq2)
  am(ind(3,3))=am(ind(3,3))-(fact*(dvdq3+dcdq3*unyw)+term2*dfdq3)
  am(ind(3,4))=am(ind(3,4))-(fact*dcdq4*unyw+term2*dfdq4)

  am(ind(4,1))=am(ind(4,1))-(fact*((dudq1*unxw+dvdq1*unyw)*cp+ubp*dcdq1 &
         +up*dudq1+vp*dvdq1+2.0*cp*dcdq1/gm1)+term3*dfdq1)
  am(ind(4,2))=am(ind(4,2))-(fact*(dudq2*unxw*cp+ubp*dcdq2+up*dudq2 &
         +2.0*cp*dcdq2/gm1)+term3*dfdq2)
  am(ind(4,3))=am(ind(4,3))-(fact*(dvdq3*unyw*cp+ubp*dcdq3+vp*dvdq3 &
         +2.0*cp*dcdq3/gm1)+term3*dfdq3)
  am(ind(4,4))=am(ind(4,4))-(fact*(ubp*dcdq4+2.0*cp*dcdq4/gm1) &
         +term3*dfdq4)
elseif (mxi.gt.zero .and. mxi.lt.clone) then
  dudq1=-up/rp
  dudq2=1.0/rp

  dvdq1=-vp/rp
  dvdq3=1.0/rp

  fact=0.5*gm1*cp/pp
  dcdq1=fact*(-ep/rp+up*up+vp*vp)
  dcdq2=-fact*up
  dcdq3=-fact*vp
  dcdq4=fact

  ev1=ubp-cp
  ev1=(ev1-ssqrt(ev1*ev1+del1))/2.0
  fact=arew*ev1*rp/2.0/gamma
  dfdq1=-arew*(cp+rp*dcdq1)/2.0/gamma
  dfdq2=arew*(unxw-rp*dcdq2)/2.0/gamma
  dfdq3=arew*(unyw-rp*dcdq3)/2.0/gamma
  dfdq4=-arew*(rp*dcdq4)/2.0/gamma

  term1=up-cp*unxw
  term2=vp-cp*unyw
  term3=-ubp*cp+qsqp+cp*cp/gm1

  am(ind(1,1))=dfdq1
  am(ind(1,2))=dfdq2
  am(ind(1,3))=dfdq3
  am(ind(1,4))=dfdq4

  am(ind(2,1))=fact*(dudq1-dcdq1*unxw)+term1*dfdq1
  am(ind(2,2))=fact*(dudq2-dcdq2*unxw)+term1*dfdq2
  am(ind(2,3))=-fact*dcdq3*unxw+term1*dfdq3
  am(ind(2,4))=-fact*dcdq4*unxw+term1*dfdq4

  am(ind(3,1))=fact*(dvdq1-dcdq1*unyw)+term2*dfdq1
  am(ind(3,2))=-fact*dcdq2*unyw+term2*dfdq2
  am(ind(3,3))=fact*(dvdq3-dcdq3*unyw)+term2*dfdq3
  am(ind(3,4))=-fact*dcdq4*unyw+term2*dfdq4

  am(ind(4,1))=fact*(-(dudq1*unxw+dvdq1*unyw)*cp-ubp*dcdq1+up*dudq1 &
         +vp*dvdq1+2.0*cp*dcdq1/gm1)+term3*dfdq1
  am(ind(4,2))=fact*(-dudq2*unxw*cp-ubp*dcdq2+up*dudq2+2.0*cp*dcdq2/gm1)&
         +term3*dfdq2
  am(ind(4,3))=fact*(-dvdq3*unyw*cp-ubp*dcdq3+vp*dvdq3+2.0*cp*dcdq3/gm1)&
         +term3*dfdq3
```

```
      am(ind(4,4))=fact*(-ubp*dcdq4+2.0*cp*dcdq4/gm1)+term3*dfdq4
    else
      do kj=1,4
      do ki=1,4
        am(ind(ki,kj))=0.0
      enddo
      enddo
    endif
  elseif (iflux.eq.3) then
!
! evaluate the flux jacobian based on
! roe's flux-difference splitting scheme
!
    gqsqm=0.5*gm1*(um*um+vm*vm)
    gqsqp=0.5*gm1*(up*up+vp*vp)
    hare=0.5*arew
    unxw=hare*unxw
    unyw=hare*unyw
    ubm=hare*ubm
    ubp=hare*ubp
!
! calculate ap based on ep
!
    ap(ind(1,1))=0.0
    ap(ind(1,2))=unxw
    ap(ind(1,3))=unyw
    ap(ind(1,4))=0.0

    ap(ind(2,1))=unxw*gqsqm-um*ubm
    ap(ind(2,2))=ubm-gm2*unxw*um
    ap(ind(2,3))=unyw*um-gm1*unxw*vm
    ap(ind(2,4))=unxw*gm1

    ap(ind(3,1))=unyw*gqsqm-vm*ubm
    ap(ind(3,2))=unxw*vm-gm1*unyw*um
    ap(ind(3,3))=ubm-gm2*unyw*vm
    ap(ind(3,4))=unyw*gm1

    ap(ind(4,1))=ubm*(gqsqm-hm)
    ap(ind(4,2))=unxw*hm-gm1*um*ubm
    ap(ind(4,3))=unyw*hm-gm1*vm*ubm
    ap(ind(4,4))=gamma*ubm
!
! calculate am based on em
!
    am(ind(1,1))=0.0
    am(ind(1,2))=unxw
    am(ind(1,3))=unyw
    am(ind(1,4))=0.0

    am(ind(2,1))=unxw*gqsqp-up*ubp
    am(ind(2,2))=ubp-gm2*unxw*up
    am(ind(2,3))=unyw*up-gm1*unxw*vp
    am(ind(2,4))=unxw*gm1

    am(ind(3,1))=unyw*gqsqp-vp*ubp
    am(ind(3,2))=unxw*vp-gm1*unyw*up
    am(ind(3,3))=ubp-gm2*unyw*vp
    am(ind(3,4))=unyw*gm1

    am(ind(4,1))=ubp*(gqsqp-hp)
    am(ind(4,2))=unxw*hp-gm1*up*ubp
```

```
   am(ind(4,3))=unyw*hp-gm1*vp*ubp
   am(ind(4,4))=gamma*ubp
  endif

  return
  end

  block data blkdat
!
!*****************************************************************************
! block dat file
!*****************************************************************************
!
  include 'params.cmd'
  include 'precs.cmd'
  include 'arrays.cmd'
  include 'param.cmd'
  include 'bcon.cmd'
!
! set initial values for parameters
!
  data autot/1.0/, cycle/0/, cyl/0.0/, delt/1.0e-10/, itcon/0/, ttime/0.0/, &
      prtdt/1.0e10/, twprt/0.0/, pltdt/1.0e10/, twplt/0.0/, twfin/0.0/, &
      twsta/0.0/, gx/0.0/, gy/0.0/, ui/0.0/, vi/0.0/, pi/1.0/, ti/1.0/, &
      velmx/1.0/, isympl/0/, fcycle/0/, rdxi/1.0/, rdet/1.0/, ivisc/0/, &
      invisd/1/, iplot/0/, ianim/0/, iprint/0/, implct/0/, epsi/0.001/, &
      rstdt/1.0e10/, isplit/2/, scaleg/1.0/, limit/1/, lgs/0/, epsig/1.0/, &
      rlxm/0.0/, rlx/1.0/, jupdat/1/, rlxq/0.95/, inorxi/1/, inoret/1/, &
      iflux/3/, iflip/2/, itord/1/, isavfm/0/, irstfm/0/, ifreq/25/, &
      lspeed/1/, iturb/0/, isweep/0/, iflag/0/, damp/0.0/, dampi/1.0/, &
      dlim/0.95/
!
! set initial values for boundary conditions
!
  data wtlb/jdim*1/, ptlb/jdim*1.0/, ttlb/jdim*1.0/, uxlb/jdim*1.0/, &
      uylb/jdim*0.0/, pslb/jdim*1.0/, tslb/jdim*1.0/, ulb/jdim*0.0/, &
      vlb/jdim*0.0/, rplb/jdim*0.0/, brlb/jdim*0.0/, wtrb/jdim*1/, &
      ptrb/jdim*1.0/, ttrb/jdim*1.0/, uxrb/jdim*1.0/, uyrb/jdim*0.0/, &
      psrb/jdim*1.0/, tsrb/jdim*1.0/, urb/jdim*0.0/, vrb/jdim*0.0/, &
      rprb/jdim*0.0/, brrb/jdim*0.0/, wtbb/idim*1/, ptbb/idim*1.0/, &
      ttbb/idim*1.0/, uxbb/idim*0.0/, uybb/idim*1.0/, psbb/idim*1.0/, &
      tsbb/idim*1.0/, ubb/idim*0.0/, vbb/idim*0.0/, rpbb/idim*0.0/, &
      brbb/idim*0.0/, wttb/idim*1/, pttb/idim*1.0/, tttb/idim*1.0/, &
      uxtb/idim*0.0/, uytb/idim*1.0/, pstb/idim*1.0/, tstb/idim*1.0/, &
      utb/idim*0.0/, vtb/idim*0.0/, rptb/idim*0.0/, brtb/idim*0.0/, impbc/0/
!
! set initial values for constants
!
  data em6/1.0e-6/, em10/1.0e-10/, em20/1.0e-20/, ep10/1.0e10/, ep20/1.0e20/, &
      c1/0.333333/, c2/0.666667/, zero/0.0/, one/1.0/, clone/0.999999/, &
      cfl/5.0/
!
! set physical properties of the gas
!
  data rgas/1.0/, gamma/1.4/, cmu/0.0/, cku/0.0/, sc1/0.0/, sc2/0.0/, &
      cvg/1.0/, cpg/1.0/, prt/1.0/, mulm/ijdim*0.0/, mu/ijdim*0.0/, &
      kulm/ijdim*0.0/, ku/ijdim*0.0/
!
! set logical variables
!
  data restrt/.false./, savers/.false./, steady/.false./, icheck/.false./, &
      source/.false./, wshear/.false./
```

```
!
! set the names
!
 data name/'flow simulation using the << sharp >> program'/
 data fngrd/'sharp.grd'/, fninp/'sharp.inp'/, fnout/'sharp.out'/
 data fnplt/'sharp.dat'/, fnrst/'sharp.rst'/, fnsav/'sharp.sav'/
 data fndlt/'delta.dat'/, fnrs1/'dnrsd.dat'/, fnrs2/'xmrsd.dat'/
 data fnrs3/'ymrsd.dat'/, fnrs4/'enrsd.dat'/, fnrs5/'ttrsd.dat'/
 data dfgrd/' '/, dfinp/' '/, dfout/' '/, dfplt/' '/, dfrst/' '/, dfsav/' '/
 data dfdlt/' '/, dfrs1/' '/, dfrs2/' '/, dfrs3/' '/, dfrs4/' '/, dfrs5/' '/
!
! zero all primitive variables
!
 data p/ijdim*1.0/, u/ijdim*0.0/, v/ijdim*0.0/, t/ijdim*1.0/, &
     dt/ijdim*1.0e-10/, mach/ijdim*0.0/, rm/0.0/, rum/0.0/, rvm/0.0/, &
     em/0.0/, um/0.0/, vm/0.0/, pm/0.0/, tm/0.0/, rp/0.0/, rup/0.0/, &
     rvp/0.0/, ep/0.0/, up/0.0/, vp/0.0/, pp/0.0/, tp/0.0/
!
! zero all conserved variables
!
 data q1/ijdim*1.0/, q2/ijdim*0.0/, q3/ijdim*0.0/, q4/ijdim*1.0/, &
     dq1/ijdim*0.0/,dq2/ijdim*0.0/, dq3/ijdim*0.0/, dq4/ijdim*0.0/, &
     dqn1/ijdim*0.0/, dqn2/ijdim*0.0/, dqn3/ijdim*0.0/, dqn4/ijdim*0.0/
!
! zero all coefficients needed in viscous flux terms
!
 data a1/ijdim*0.0/, a2/ijdim*0.0/, a3/ijdim*0.0/, a4/ijdim*0.0/, &
     a5/ijdim*0.0/, a6/ijdim*0.0/, a7/ijdim*0.0/, a8/ijdim*0.0/, &
     a9/ijdim*0.0/, a10/ijdim*0.0/, a11/ijdim*0.0/, b1/ijdim*0.0/, &
     b2/ijdim*0.0/, b3/ijdim*0.0/, b4/ijdim*0.0/, b5/ijdim*0.0/, &
     b6/ijdim*0.0/, b7/ijdim*0.0/, b8/ijdim*0.0/, b9/ijdim*0.0/, &
     b10/ijdim*0.0/, b11/ijdim*0.0/
!
! zero all geometry
!
 data x/ijdim*0.0/, y/ijdim*0.0/, yc/ijdim*1.0/, unxr/ijdim*0.0/, &
     unyr/ijdim*0.0/, arer/ijdim*0.0/, unxt/ijdim*0.0/, unyt/ijdim*0.0/, &
     aret/ijdim*0.0/, jac/ijdim*0.0/
!
! zero all sources
!
 data omg1/ijdim*0.0/, omg2/ijdim*0.0/, omg3/ijdim*0.0/, omg4/ijdim*0.0/, &
     avea/nvsq*0.0/, terr/nvar*0.0/, errl/nvar*0.0/
!
! zero all jacobians
!
 data ap/ijnvs*0.0/, am/ijnvs*0.0/, bp/ijnvs*0.0/, bm/ijnvs*0.0/, &
     dhdq/ijnvs*0.0/
!
! zero all other stuff
!
 data i/2/, j/2/, ij/2/, im1/3/, jm1/3/, im2/2/, jm2/2/, im2jm2/4/, imax/4/, &
     jmax/4/, ei1/0.0/, ei2/0.0/, ei3/0.0/, ei4/0.0/, ev1/0.0/, ev2/0.0/, &
     ev3/0.0/, ev4/0.0/, fi1/0.0/, fi2/0.0/, fi3/0.0/, fi4/0.0/, fv1/0.0/, &
     fv2/0.0/, fv3/0.0/, fv4/0.0/, h1/0.0/, h2/0.0/, h3/0.0/, h4/0.0/, &
     ilow/2/, jlow/2/, ihgh/3/, jhgh/3/

 end

 subroutine bcimp
!
!******************************************************************************
```

```
! set boundary conditions
! option 1 : rigid free-slip boundary condition.
! option 2 : rigid no-slip boundary condition.
! option 3 : subsonic inlet boundary condition.
! option 4 : supersonic inlet boundary condition.
! option 5 : outlet boundary condition.
! option 6 : mass injection boundary condition.
!*********************************************************************************
!
 include 'params.cmd'
 include 'precs.cmd'
 include 'arrays.cmd'
 include 'param.cmd'
 include 'bcon.cmd'

 return
 end
```

ضمیمه ج

مقالات ارائه شده در کنفرانسهای داخلی و خارجی

Paper No: M061

18th National & 7th ISHMT-ASME
Heat and Mass Transfer Conference
January 4 - 6, 2006
IIT Guwahati, India

Cross Wind and Natural Draft Dry Cooling Towers

M.H. Kayhani
Mechanical EngineeringDepartment
Shahrood University of Technology
h_kayhani@shahrood.ac.ir

M. Shahsavan
Energy Group
MOSHANIR co.
mshahsavan@yahoo.com

A. Abbasnejad
Mechanical EngineeringDepartment
Shahrood University of Technology
abbasnejadali@gmail.com

1. Abstract

The major objective of this research work is to study the impact of special type of wind break walls on performance improvement of natural draft dry cooling towers under cross wind condition. Using the finite volume method, the fluid flow and temperature distribution around and in a tower are simulated. $k-\varepsilon$ model was employed for turbulent flow modeling. A new parameter called "mass efficiency" was introduced to compare different cases. The results have indicated improvement in the performance of cooling tower when wind break walls are used. Losses due to the cross wind is reduced and electricity production is increased. The predicted numerical results were validated in forced convection case with Su et al. results.

2. Nomenclature

V	velocity vector
T	temperature
K	turbulent kinetic energy
P	kinetic energy production due to turbulence

Greek Letters

σ	stress tensor
β	volume expansion coeff.
ε	dissipation rate
μ	viscosity
η	mass efficiency

Subscripts

t	turbulence
k	due to kinetic energy
ε	due to dissipation rate

3. Introduction

Natural draft dry cooling towers are a type of cooling towers which are used under certain conditions such as high water temperature and insufficient water supplies. Dry cooling towers that depend on convection and use air as the transport medium may be preferable under aforementioned conditions.

The performance of all air-cooled heat exchangers and cooling towers are affected by changes in ambient conditions. Changes in temperature, humidity, winds, inversions and rain all influence the performance of cooling towers.

In summer and wind seasons, the cooling efficiency of the tower is evidently reduced and the electricity produced by the power plant is decreased. Because of importance of wind effect on dry cooling towers performance, many investigations have been done either numerically and experimentally.

Bergstom et al. [4] did the first 2D simulation of fluid flow in cooling tower. Demoren and Rodi [5] modeled the cooling tower as an empty tube with heat fluid from its bottom. Using PHOENICS software, Du Preez and Kroger [7] studied the effect of wind on the homon type cooling tower performance. They used a porous wall at the center of tower as wind break wall for reducing the wind effect.

Su et al. [8] numerically studied the performance of the Heller dry cooling tower under cross wind. Using FLUENT, Al-Waked and Behnia [9] studied the effect of the cross wind on thermal performance of Homon-type cooling towers and employed wind break walls to reduce this effect.

The aim of this research work is to study the impact of wind break walls on the performance of towers under cross wind condition. Finite volume method is used to simulate the fluid flow and temperature distribution around and in a Heller type dry cooling tower. In absence of high circulating, k–ε method was employed for turbulent flow modeling. A new parameter, "mass efficiency" or "hydrodynamic efficiency" was introduced to compare different cases. The fluid flow in three phases was considered.

In the first phase, a CFD code was developed by authors to simulate natural convection in dry cooling tower in the absence of wind. In the second phase, forced convection due to cross wind effect was studied by utilizing FLUENT.

At the last phase, the effect of a special type of wind break wall on the performance improvement of cooling towers was studied.

4. Governing Equations

The following assumptions were considered to derive the governing equations:

The flow is steady, three dimensional and incompressible. The variation of density due to pressure can be neglected and thus the density varies only with temperature and the buoyancy term is considered. The principal equations in a vector form are:

$$\nabla \cdot V = 0 \tag{1}$$

$$(V \cdot \nabla)V = -\frac{1}{\rho}\nabla P + \nabla \cdot \left(\frac{\sigma}{\rho}\right) - \beta(T - T_a)g + F \tag{2}$$

$$\rho(V \cdot \nabla)T = -\nabla \cdot \left[(\Gamma + \Gamma_t)\nabla T\right] + Q \tag{3}$$

When air flows through the radiators, heat is added to air from the radiators and therefore a heat source term is added in the energy equation. In the k–ε turbulence model the dynamic viscous coefficient is written as:

$$\upsilon_t = \frac{1}{\rho}\mu_t = c_\mu \frac{k^2}{\varepsilon} \tag{4}$$

Where equations for turbulence kinetic energy, k, and dissipation rate,ε are:

$$(V \cdot \nabla)k = \nabla . \left[(\upsilon + \upsilon_t/\sigma_k)\nabla k\right] + P + G - \varepsilon \tag{5}$$

$$(V \cdot \nabla)\varepsilon = \nabla . \left[(\upsilon + \upsilon_t/\sigma_\varepsilon)\nabla k\right] + c_{1\varepsilon}\frac{\varepsilon}{k}(P + G) - c_{2\varepsilon}\frac{\varepsilon^2}{k} \tag{6}$$

5. Numerical Models

In this study the fluid flow was studied numerically in three models: natural convection (nominal condition), forced convection due to wind and a model including wind break walls.

5.1 Natural Convection Model

In natural convection case, the fluid flow in the cooling tower is simulated by using a CFD code which has been developed by authors. The code has ability to generate 2D and 3D meshes and ability to solve 2D and axisymmetric continuity, momentum and energy equations. The flow and temperature fields in natural convection case are axially symmetric.

In this case, firstly, the grid is generated by isoparametric coordinates method. This two dimensional grid generation method was introduced by Zienkiewicz & Philips [2].

The generated grid cell number using this technique for current geometry is 11×36 which depicted in fig. 1. In this case, the atmospheric pressure was considered to define the inlet pressure.

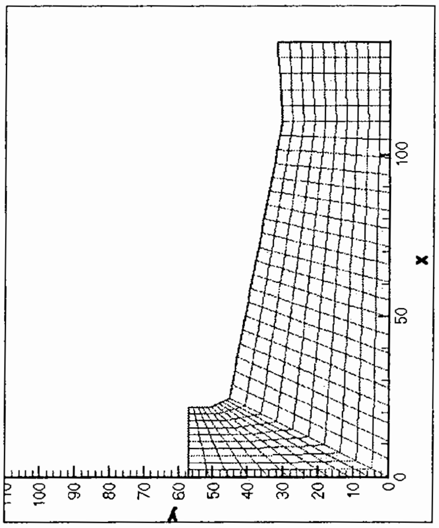


Fig. 1: Grid for natural convection model.

5.2 Forced Convection Model

In the forced convection model the algebraic equations of discredited governing equations are discrete and solved by an implicit method and a segregated solver. The SIMPLE algorithm is used for the calculation of the pressure and therefore the velocity field.

The wind velocity was used to define the inlet boundary condition and the air static pressure at the flow outlet boundary for forced convection case (windy condition). Three dimensional fluid flow has a symmetric plane, parallel to the wind direction.

5.3 Wind Break Walls Model

Wind break walls first were introduced by du Preez and Kroger [7], and they applied a porous wall in the centre of a Homon-type cooling tower (inside the tower) as a wind break wall.

Another type of wind break walls was applied by Al-Waked & Behnia [9]. They used eight rectangular shaped walls on the outside of the tower. Their work shows the good improvement in the performance of the tower under cross wind condition.

In this paper, two types of wind break walls are introduced. First type as shown in fig. 2 is two curve shaped walls and another (fig. 3) is four curve shaped walls.

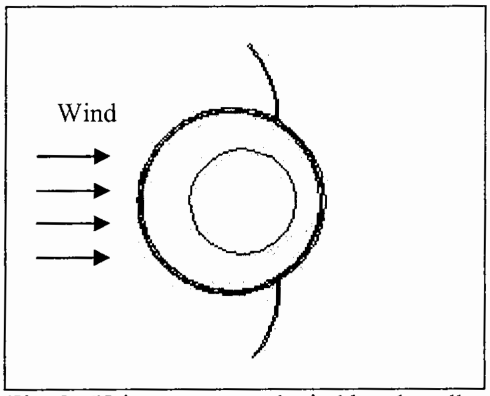


Fig. 2: Using two curved wind break walls.

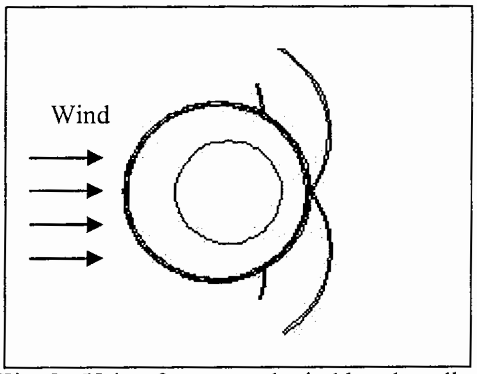


Fig. 3: Using four curved wind break walls.

The no slip principle was used for solid regions such as wind break walls, tower shell and ground. The pressure drop through radiators is assumed to be proportional to the dynamic head of air and for modeling radiators; a porous zone with thermal source term is selected at the inlet boundary of the tower.

$$\Delta P = K_L \frac{1}{2} \rho v^2 \qquad (7)$$

6. Results and Discussions

To investigate the performance of cooling towers in aforementioned models a real industrial scale cooling tower was used with following specifications is used.
Bottom diameter: 110 m
Tower height: 130 m
Throat diameter: 62 m
Radiator height: 20 m
Ejected heat: 404 MW

Fig. 4 shows the calculation domain dimensions and boundary types. Domain is consists of a rectangular parallelepiped with 700×400×400 m dimensions, around the tower.

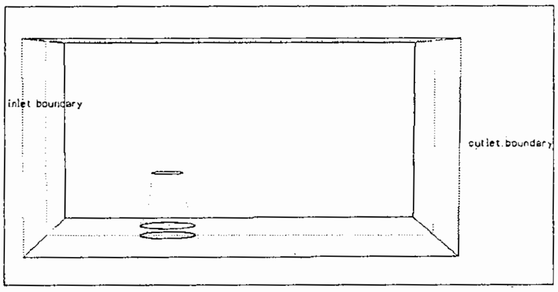


Fig. 4: Physical domain.

Figs 5 and 6 show the velocity vectors, contours of temperature respectively for natural convection model. In this model the flow field and temperature distribution are axisymmetric and so a half of the tower was considered.

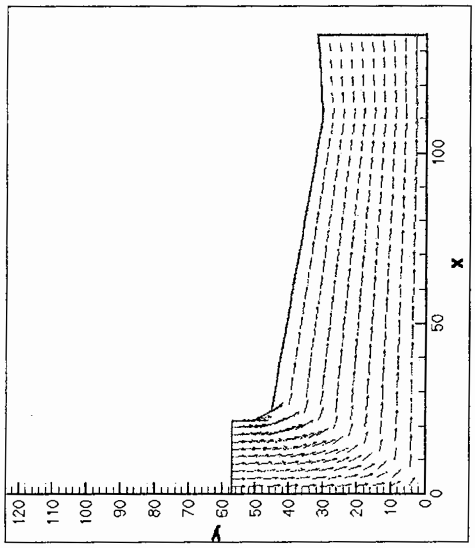


Fig. 5: Velocity vectors for natural convection model.

Fig. 7 shows the velocity vectors in symmetric plane for 5 m/s wind velocity for forced convection model. Contours of temperature in symmetric plane are shown in fig. 8. Fig. 9 shows the velocity vectors in the horizontal plane at the level of 10 m and in this fig. there is a couple of vortexes in the tower, which are formed due to the convergence of two flows through the front part and back part of the tower with different intensities.

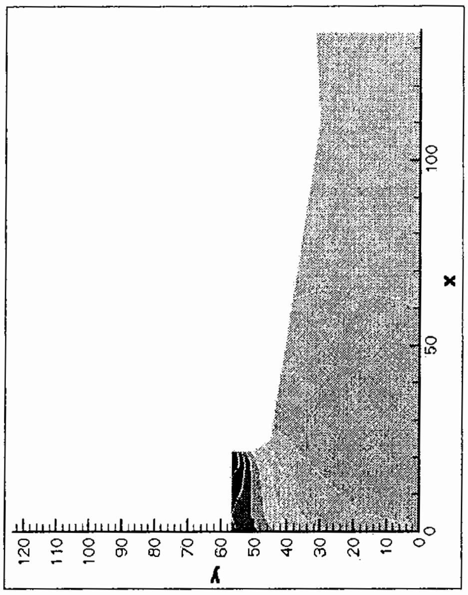


Fig. 6: Contours of temperature for natural convection model.

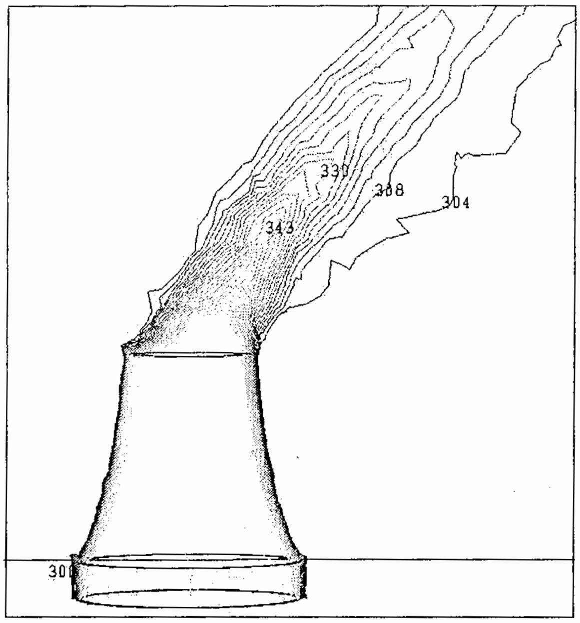


Fig. 8: Contours of temperature in symmetry plane.

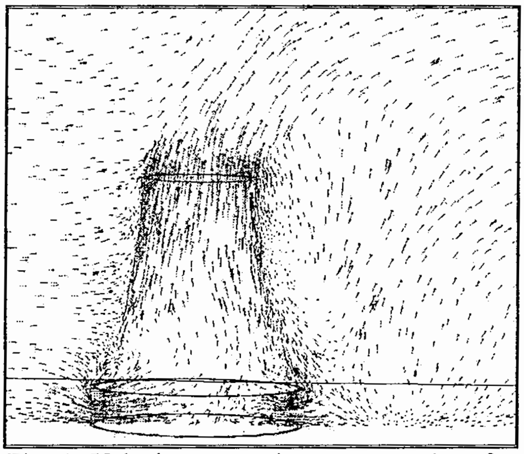


Fig. 7: Velocity vectors in symmetry plane for 5 m/s wind speed.

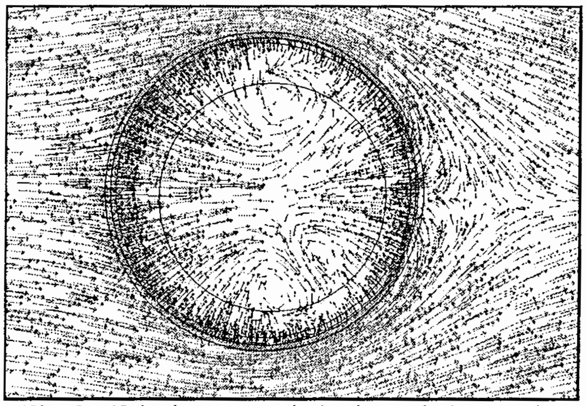


Fig. 9: Velocity vectors in horizontal plane at the level of 10 m.

The flux of entering air into the tower is reduced greatly in the side part of the tower (outside the radiators) and so the cooling efficiency decreases because the air tangential velocity is lower similar to flow over a cylinder. According to fig. 6, the flux of air entering the radiators increases in the front part of the tower (facing the wind).

Fig. 10 shows the velocity vectors in the symmetric plane for wind velocity equal to 10 m/s and it is clear that there is a "wind-cover" over the tower. This phenomenon is occurred because the air exit velocity of tower is less than the wind velocity moving over the tower. Wind-cover acts as a cap over the tower. With increasing wind velocity, this phenomenon becomes more and more strong.

To validate the predicted numerical results, these results were compared with Su et al. [4] work and comparison was showed good agreement for these results.

The flux of air moving through radiators can be an important factor for heat transfer from radiators and thus cooling efficiency. So the new parameter, η, named "mass efficiency" or "hydrodynamic efficiency" that is the ratio of air flux at different conditions such as different wind speeds and high ambient temperature to nominal condition air flux (natural convection), is a suitable parameter for cooling performance investigation.

$$\eta = \frac{\dot{m}}{\dot{m}_{nat.convection}} \tag{8}$$

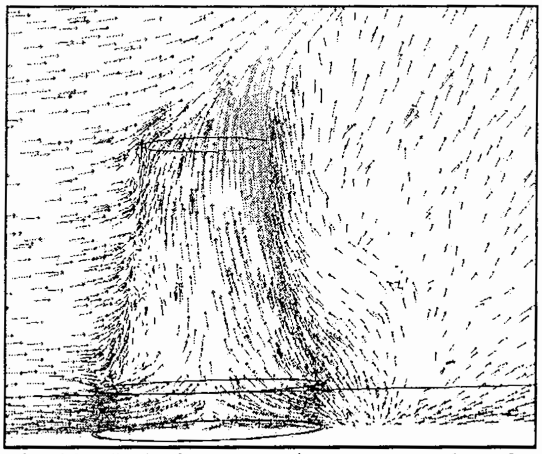


Fig. 10: Velocity vectors in symmetry plane for 10 m/s wind speed.

Fig. 11 shows the value of η in different wind velocities for two cases without and with using wind break walls. Without using wind break walls, it can be seen that η decreases when the wind velocity increases. When the value of η decreases, the air flux moving through radiators decreases and the radiators can not work in nominal condition, so the cooling efficiency and finally the electricity production reduces.

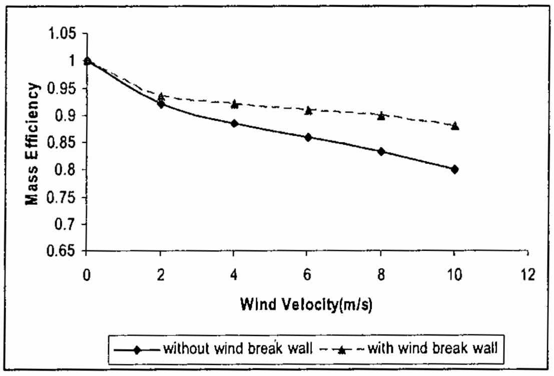


Fig. 11: Value of Mass Efficiency in Different Wind Velocities.

As mentioned in forced convection part, the air flux decreases in the side part of the tower and thus the walls located at the $\theta = 120°$ where θ measured from wind direction. We introduce a new parameter to compare the performance of tower at different conditions such as with or without wind break walls.

Table 1 shows the value of mass efficiency for different cases at 10 m/s wind velocity. As seen in table 1, using both types of wind break walls improve the tower performance because the value of η shows increasing in comparison with no wind break wall condition.

Table 1: Values of Mass Efficiency for Different Cases.

State	Mass Efficiency (η)
Natural Convection	1
10 m/s Wind Speed	0.8
Using Two Wind Break Walls	0.86
Using Four Wind Break Walls	0.89

Velocity vectors in horizontal plane for 10 m/s wind velocity was shown in fig. 12, when four wind break walls was used. It is clear that the effect of vortexes mentioned in forced convection case is weaker.

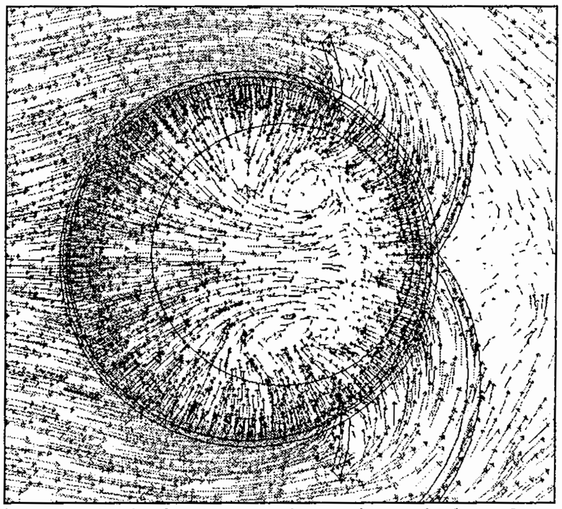


Fig. 12: Velocity Vectors in Horizontal Plane for 10 m/s Wind Speed..

Increasing in value of η and so the radiators air flux can also improve the wind-cover effect because the air exit velocity at the top of tower will increase. This effect is shown in fig. 13 for 10 m/s wind speed.

Also fig.11 shows increasing the value of η for all wind velocities with using wind break walls and thus increasing in η causes the performance improvement of tower. Therefore using wind break walls improve the tower performance in different wind velocities.

Comparison between fig. 13 and fig. 8 shows that the wind-cover without using wind break walls is stronger than using them.

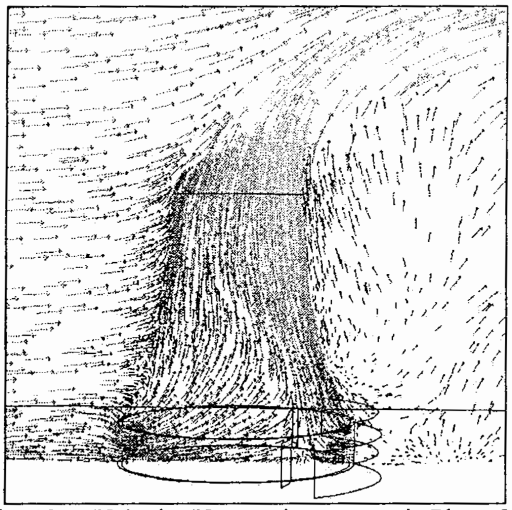


Fig. 13: Velocity Vectors in symmetric Plane for 10 m/s Wind Speed with Wind Break Walls.

7. Conclusion

According to the predicted numerical results mentioned in this study the reasons for loss in performance of natural draft dry cooling towers are:

1) Vortex production in the bottom of tower.

2) Decreasing in flux of air entering the tower from side part.

3) Difference between intensities of the air flow exits from cooling tower and wind flow at top of the tower that causes "wind-cover".

As mentioned in last section, change in the external shape of the tower such as installing wind break walls can improve the cooling efficiency under cross wind condition.

8. References

1. C. Hirsch, Numerical Computation of Internal and External Flows, Vol.2, John Wily & sons, 1990.
2. O.C Zienkiewicz and D.V Philips, An Automatic Mesh Generation Scheme for Plane and Curved Surfaces by Isoparametric coordinates, International Journal for Numerical Method in engineering, Vol.3, pp. 319-328, 1971.
3. J.C. Tannehill, D.A. Anderson and Pletcher R.H., Computational Fluid Mechanics and Heat Transfer, Taylor & Francis, 2nd Edition, 1997.
4. D. Bergetrom, D. Derkson and K.Rezkallah, Numerical Study of Wind Flow Over a Cooling Tower, Journal of Wind Engineering and Industrial Aerodynamics, Vol. 46-47, pp. 657-664, 1993.
5. D. Demoren and W. Rodi, Three Dimensional Numerical Calculations of Flow and Plume Spreading Past Cooling Towers, Journal of Heat Transfer, Vol. 109, pp. 113-119, 1987.
6. FLUENT, User's Guide, FLUENT Incorporated, Lebanon, NH, 1999.
7. A.F. Du Preez and D. Kroger, the Effect of The Heat Exchanger Arrangement and Wind Break Walls on The Performance of Natural Draft Dry-Cooling Towers Subjected to Cross-Winds, journal of Wind Engineering and Industrial aerodynamics, Vol. 58, pp. 293-303, 1995.
8. M. Su, G. Tang and S. Fu, Numerical simulation of fluid Flow and Thermal Performance of a Dry-Cooling Tower Under Cross Wind Condition, Journal of Wind Engineering and Industrial Aerodynamics, Vol. 79, No. 3, pp. 289-306, 1999.
9. R. Al-Waked and M. Behnia, The Performance of Natural Draft Dry Cooling Towers under Cross wind: CFD Study, International Journal of energy Research, Vol. 28, pp. 147-161, 2004.
10. D.G. Kroger, Air-Cooled Heat Exchanger and Cooling Towers, Thermal Flow Performance Evaluation and Design, Begell House, Inc, New York, 1998.

HEFAT2005
4th International Conference on Heat Transfer, Fluid Mechanics and Thermodynamics
Cairo, Egypt
Paper number: KM1

CROSS WIND AND NATURAL DRAFT DRY COOLING TOWERS: EFFECTS OF HIGH AMBIENT TEMPERATURE AND WIND BREAK WALLS

Kayhani M.H, Shahsavan M and Abbasnejad A*
*Author for correspondence
Mechanical Engineering Faculty,
Shahrood University of Technology,
Shahrood,
Iran,
E-mail: abbasnejadali@yahoo.com

ABSTRACT

The purpose of this paper is to study the effect of wind, different types of wind break walls and ambient temperature on the performance of natural draft dry cooling towers. The performance of cooling towers under above mentioned cases was investigated numerically. Using a general purpose CFD code, a three dimensional fluid flow and temperature distribution around and in a cooling tower were simulated. In absence of high circulating, k-ε turbulence model was used for turbulent flow modeling. The performance losses reasons was discussed in brief and then, the performance of cooling tower under cross wind with using different types of wind break walls was investigated. The results have indicated a good improvement in reducing the performance losses due to cross wind. Finally, the effect of high ambient temperature was discussed.

INTRODUCTION

Natural draft dry cooling towers are known for their advantages as water resource saving, protecting the environment, pollution free. But it is found that the performance and cooling efficiency of cooling towers are seriously dependent on the environmental conditions, such as ambient temperature, cross wind speed, humidity, inversions and rain.

In summer and wind seasons the cooling efficiency of the tower is evidently reduced and the electricity production by power plant reduced to a large extent due to condenser vacuum decreasing. Because of importance of wind effect on dry cooling towers performance, many investigations have been done either numerically and experimentally.

Bergstom et al. [1] did the first 2D simulation of fluid flow in cooling tower. Demoren and Rodi [2] modeled the cooling tower as an empty tube with heat fluid from its bottom. Using PHOENICS software, Du Preez and Kroger [4] studied the effect of wind on the homon type cooling tower performance. They used a porous wall at the center of tower as wind break wall for reducing the wind effect.

Su et al. [5] numerically studied the performance of the Heller dry cooling tower under cross wind. Using FLUENT, Al-Waked and Behnia [6] studied the effect of the cross wind on thermal performance of Homon-type cooling towers and employed wind break walls to reduce this effect.

The aim of this research work is to study the impact of wind break walls on the performance of towers under cross wind condition. Finite volume method is used to simulate the fluid flow and temperature distribution around and in a Heller type dry cooling tower. In absence of high circulating, k-ε method was employed for turbulent flow modeling. A new parameter, "mass efficiency" or "hydrodynamic efficiency" was introduced to compare different cases.

The major objective of this research work is to study the impact of different types of wind break walls on the natural draft dry cooling tower performance under cross wind.

Firstly the reasons for performance losses were discussed briefly and then different arrangements of wind break walls and advantages and disadvantages were studied. Then the performance improvement using these walls was discussed. Finally the effect of ambient temperature was studied.

NOMENCLATURE

V	velocity vector
T	temperature
K	turbulent kinetic energy
P	kinetic energy production due to turbulence

Greek Letters

σ	stress tensor
β	volume expansion coeff.
ε	dissipation rate

μ	viscosity
η	mass efficiency

Subscripts

t	turbulence
w	windy condition
wo	water outlet
a	ambient
ai	air inlet
k	due to kinetic energy
ε	due to dissipation rate

GOVERNING EQUATIONS

The following assumptions were considered to derive the governing equations:

The flow is steady, three dimensional and incompressible. The variation of density due to pressure can be neglected and thus the density varies only with temperature and the buoyancy term is considered. The principal equations in a vector form are:

$$\nabla \cdot V = 0 \qquad (1)$$

$$(V \cdot \nabla)V = -\frac{1}{\rho}\nabla P + \nabla \cdot (\frac{\sigma}{\rho}) - \beta(T - T_a)g + F \qquad (2)$$

$$\rho(V \cdot \nabla)T = -\nabla \cdot [(\Gamma + \Gamma_t)\nabla T] + Q \qquad (3)$$

When air flows through the radiators, heat add to air from the radiators and therefore a heat source term is added in the energy equation. In the k–ε turbulence model the dynamic viscous coefficient is written as:

$$\upsilon_t = \frac{1}{\rho}\mu_t = c_\mu \frac{k^2}{\varepsilon} \qquad (4)$$

Where equations for turbulence kinetic energy, k, and dissipation rate,ε are:

$$(V \cdot \nabla)k = \nabla.[(\upsilon + \upsilon_t/\sigma_k)\nabla k] + P + G - \varepsilon \qquad (5)$$

$$(V \cdot \nabla)\varepsilon = \nabla.[(\upsilon + \upsilon_t/\sigma_\varepsilon)\nabla k] + c_{1\varepsilon}\frac{\varepsilon}{k}(P + G) - c_{2\varepsilon}\frac{\varepsilon^2}{k} \qquad (6)$$

EFFECT OF WIND

Measurements performed on natural draft cooling towers subject to cross winds indicate a rise in outlet water temperature with increasing wind speed for a given heat rejection rate. In a dry cooling tower change in approach temperature (difference between outlet water temperature and air entering the tower) expressed as follows:

$$\Delta T_{WO} = \Delta(T_{WO} - T_{ai}) = (T_{WO} - T_{ai})_W - (T_{WO} - T_{ai}) \qquad (7)$$

Fig 1 shows the rise in cooling water temperature (approach temperature) for six power plants.

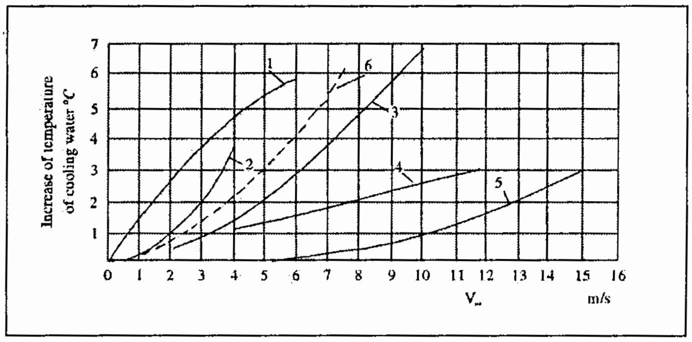


Figure 1: Rise in Cooling Temperature for different power plants: 1) Lazdain in Russia; 2) Ibenbiron in Germany; 3) Kakalin in Hungary; 4) Grud fry No.5 in South Africa; 5) Grud fry No.6 in South Africa.

It is clear that for all aforementioned power plants, the approach temperature increases with wind speed increasing. Kroger [7] mentioned that cooling towers with horizontally heat exchanger arrangement are less sensitive to wind.

For determining the tower performance losses under cross wind condition, using FLUENT the fluid flow of a tower simulated numerically. In this case the algebraic equations of discredited governing equations are discrete and solved by an implicit method and a segregated solver. The SIMPLE algorithm is used for the calculation of the pressure and therefore the velocity field.

Fig. 2 shows the velocity vectors in symmetric plane for 10 m/s wind velocity for forced convection model. Contours of temperature in symmetry plane are shown in fig. 3. Fig. 4 shows the velocity vectors in the horizontal plane at the level of 10 m and in this fig. there is a couple of vortexes in the tower, which are formed due to the convergence of two flows through the front part and back part of the tower with different intensities.

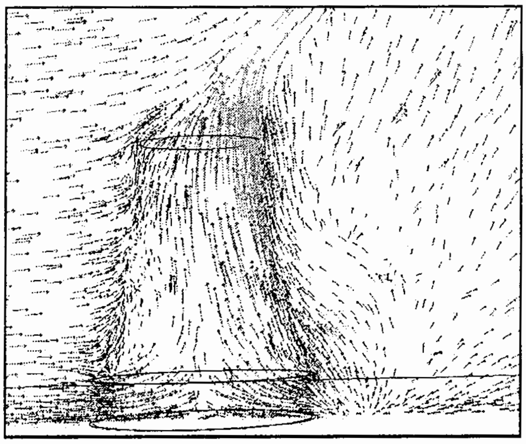


Figure 2: Velocity Vectors in Symmetry Plane.

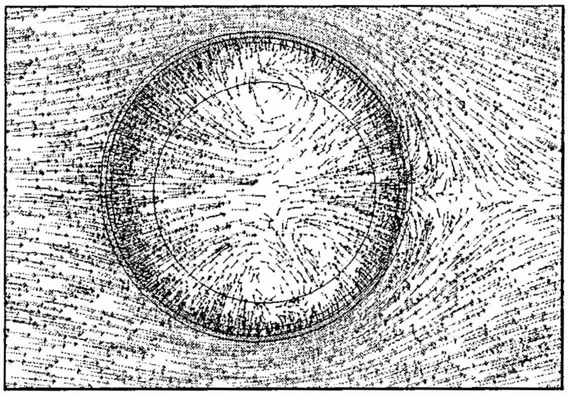


Figure 4: Velocity Vectors in Horizontal Plane at the Level of 10 m.

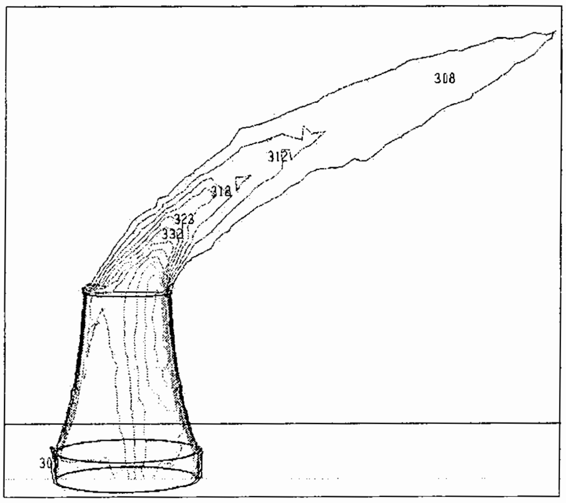


Figure 3: Contours of Temperature in Symmetry Plane.

The flux of entering air into the tower is reduced greatly in the side part of the tower (outside the radiators) and so the cooling efficiency decreases because the air tangential velocity is lower similar to flow over a cylinder. Fig. 5 shows the pressure contours in horizontal plane at the level of 10 m. Distribution of static pressure at the tower inlet is shown in Fig. 6. According to Figs 5, 6 it is clear that pressure distribution at the tower inlet is similar to the flow over a cylinder.

According to fig. 4, the flux of air entering the radiators increases in the front part of the tower (facing the wind).

From fig. 2 it is clear that there is a "wind-cover" over the tower. This phenomenon is occurred because the air exit velocity of tower is less than the wind velocity moving over the tower. Wind-cover acts as a cap over the tower. With increasing wind velocity, this phenomenon becomes more and more strong.

The flux of air moving through radiators can be an important factor for heat transfer from radiators and thus cooling efficiency. So the new parameter, η, named "mass efficiency" or "hydrodynamic efficiency" that is the ratio of air flux at different conditions such as different wind speeds and high ambient temperature to nominal condition air flux (natural convection and 288 K ambient temperature), is a suitable parameter for cooling performance investigation.

$$\eta = \frac{\dot{m}}{\dot{m}_{nat.convection}} \tag{8}$$

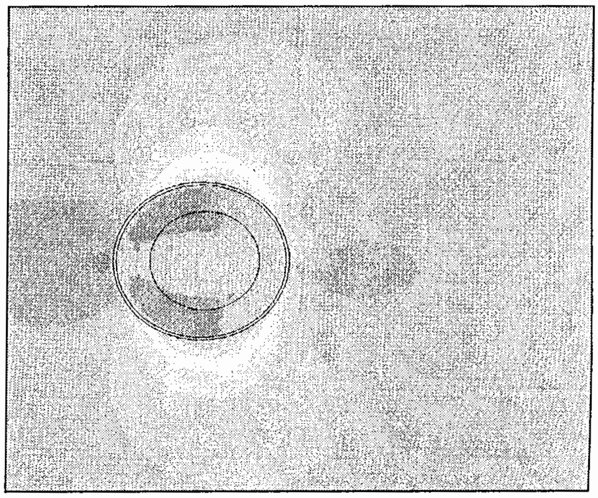


Figure 5: Contours of Static Pressure in Horizontal Plane.

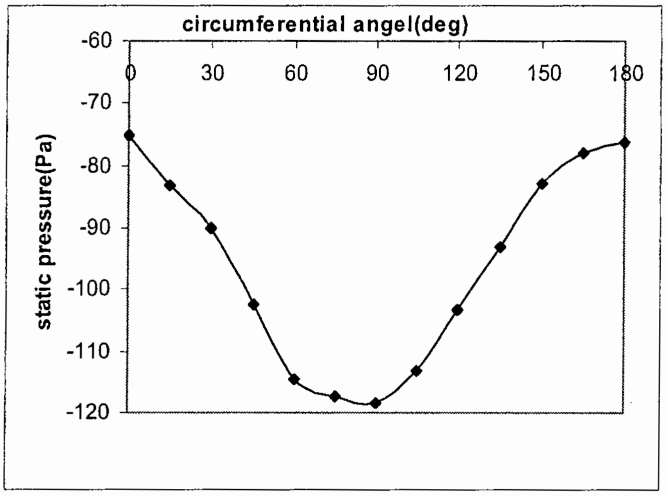


Figure 6: Distribution of Static Pressure at the Tower Inlet.

Fig. 7 indicates the values of η for different wind velocities. It can be seen that η decreases when the wind velocity increases. When the value of η decreases, the air flux moving through radiators decreases and the radiators can not work in nominal condition, so the cooling efficiency and finally the electricity production reduces.

According to the predicted numerical results the reasons for loss in performance of natural draft dry cooling towers are:
1) Vortex production in the bottom of tower.
2) Decreasing in flux of air entering the tower from side part.
3) Difference between intensities of the air flow exits from cooling tower and wind flow at top of the tower that causes "wind-cover".

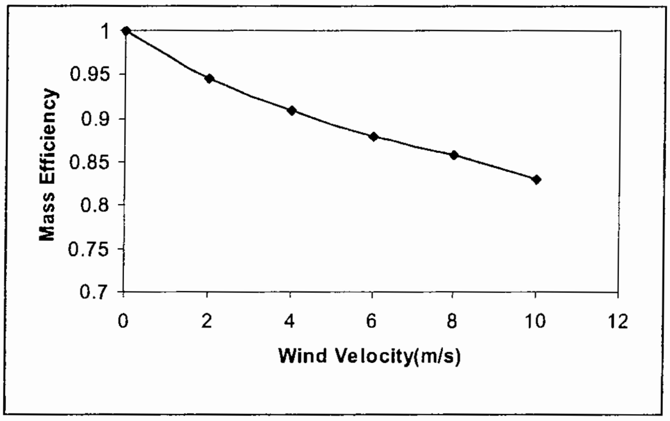


Figure 7: Mass Efficiency Values in Different Wind Velocities (Ta=288 K).

WIND BREAK WALL TYPES

Wind break walls first were introduced by du Preez and Kroger [4], and they applied a porous wall in the centre of a Homon-type cooling tower (inside the tower) as a wind break wall.

Another type of wind break walls was applied by Al-Waked & Behnia [6]. They used eight rectangular shaped walls on the outside of the Homon type cooling tower. Their work shows the good improvement in the performance of the tower under cross wind condition.

In this paper five different types of wind break walls were introduced and employed for cooling tower performance improvement these wind break wall types are shown in figs 8 to 12. Figures 8 and 9 show using two and four curve shaped wind break walls. As mentioned in last part, the air flux decreases in the side part of the tower and thus the walls located at the θ = 120° where θ measured from wind direction. So the walls were considered in θ=120°.

In fig. 10 the walls are normal to the inlet boundary of the tower. All aforementioned walls are dependent to wind direction. But we apply other two wall types that are not dependent on wind direction. Fig. 11 shows two walls which installed at θ=90.

As shown in fig 12 eight wind break walls are located radially at the tower inlet. This walls arrangement is not depend on wind direction and it is an advantage for this arrangement.

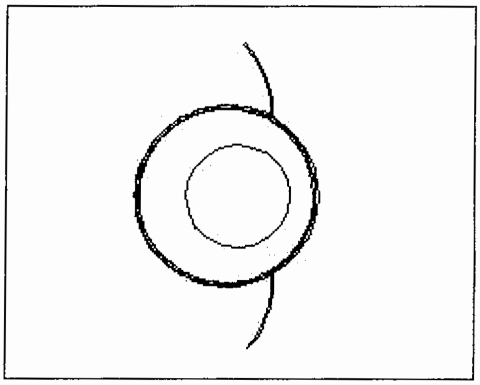


Figure 8: Two Curve Shaped Wind Break Walls.

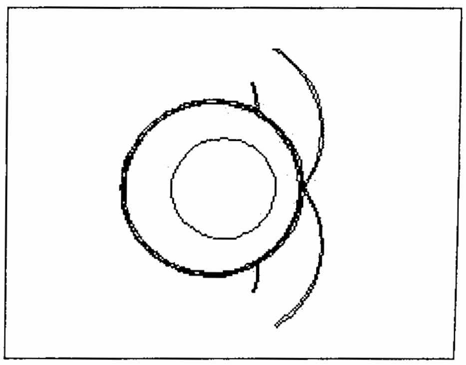


Figure 9: Four Curve Shaped Wind Break Walls.

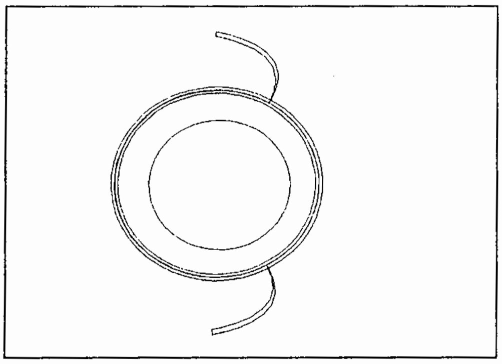


Figure 10: Two Curve Shaped Wind Break Walls (normal to tower inlet).

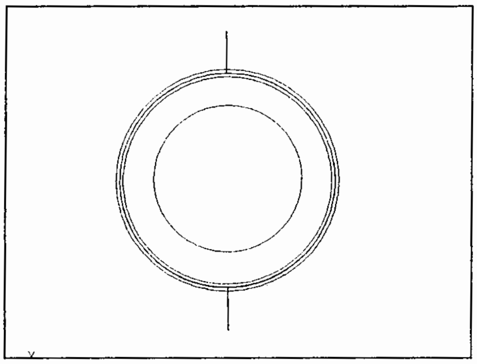


Figure 11: Two Radial Wind Break Walls.

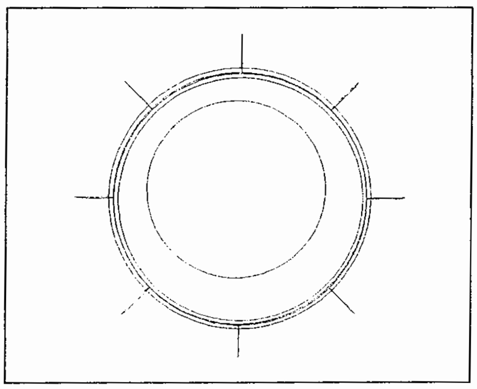


Figure 12: Eight Radial Wind Break Walls.

RESULTS AND DISCUSSIONS

To investigate the performance of cooling towers in aforementioned models a real industrial scale cooling tower was used with following specifications is used.

Bottom diameter: 110 m

Tower height: 130 m

Throat diameter: 62 m

Radiator height: 20 m

Ejected heat: 404 MW

Domain is consists of a rectangular parallelepiped with 700×400×400 m dimensions, around the tower. The wind velocity was used to define the inlet boundary condition and the air static pressure at the flow outlet boundary for forced convection case (windy condition).

The no slip principle was used for solid regions such as wind break walls, tower shell and ground. The pressure drop through radiators is assumed to be proportional to the dynamic head of air and for modelling radiators; a porous zone with thermal source term is selected at the inlet boundary of the tower. Three dimensional fluid flow has a symmetric plane, parallel to the wind direction.

The velocity vectors in horizontal plane at the level of 10 m were shown in figures 13 to 16. According to these figs, it is clear that the aforementioned couple of vortexes become weaker, so the air flux entering the tower and moving through radiators increases at the side part of the tower.

Fig. 17 shows the velocity vectors in symmetry plane when eight radial wind break walls were used. It can be seen that the wind cover effect is weaker too.

Therefore the air flux entering the tower and moving through radiators increases and then the value of η increases. Different values of η were shown in figs 18 and 19 for two types of wind break walls.

According to these figs the tower performance improves with using wind break walls.

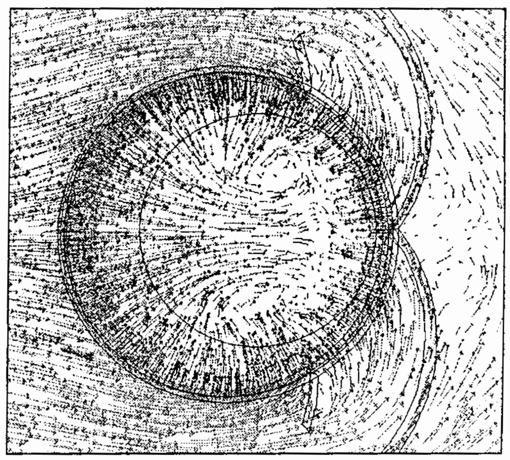


Figure 13: Velocity Vectors in Horizontal Plane Using Four Curved Wind Break Walls.

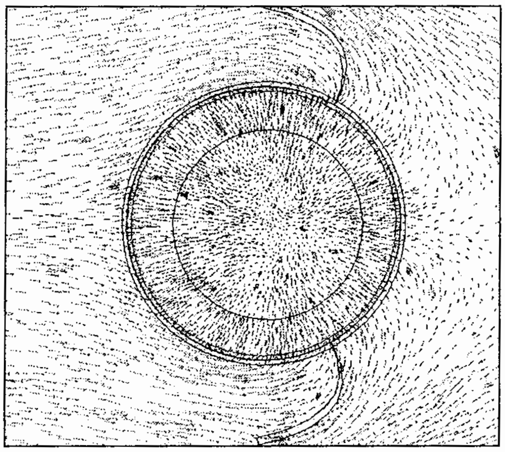


Figure 14: Velocity Vectors in Horizontal Plane Using Two Normal Wind Break Walls.

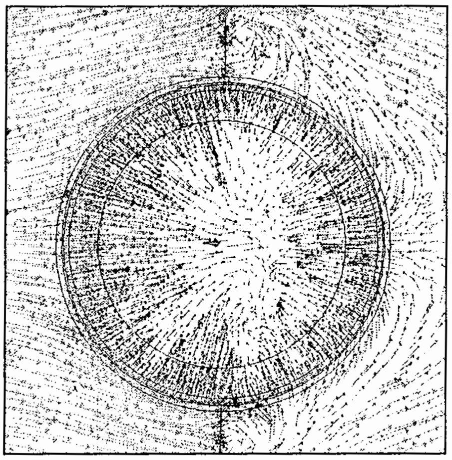


Figure 15: Velocity Vectors in Horizontal Plane Using Two Radial Wind Break Walls.

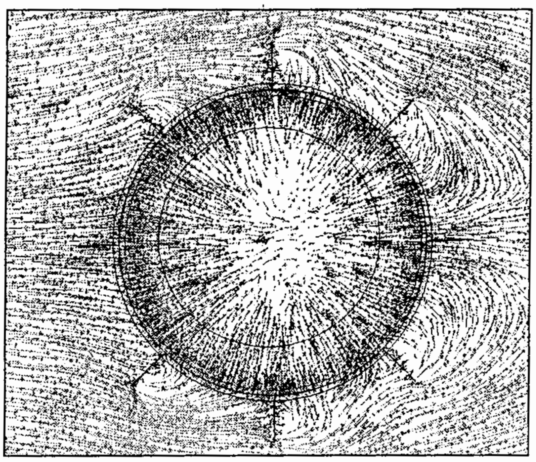


Figure 16: Velocity Vectors in Horizontal Plane Using Eight Radial Wind Break Walls.

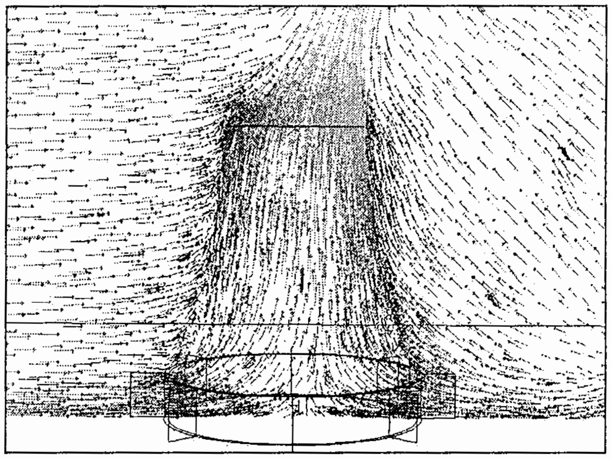


Figure 17: Velocity Vectors in Symmetry Plane Using Eight Radial Wind Break Walls

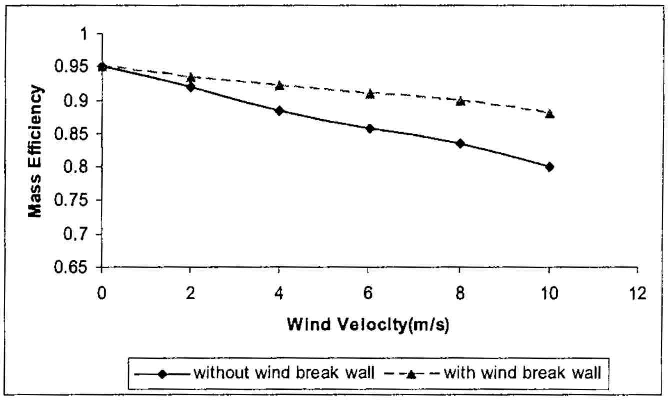


Figure 18: Mass Efficiency Values Using Four Curved Walls (Ta=300 K).

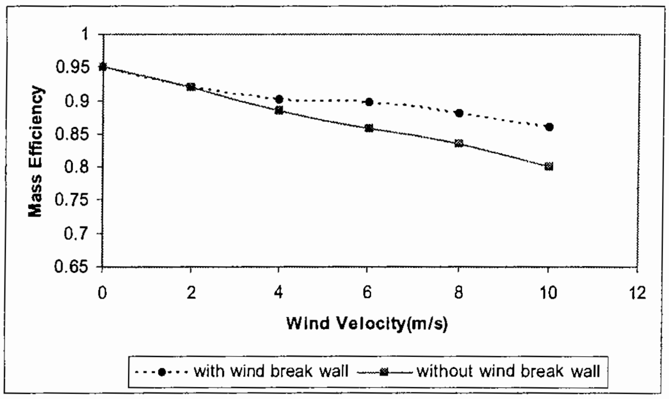


Figure 19: Mass Efficiency Values Using Eight Radial Walls (Ta=300 K).

For all wind velocities the mass efficiency was increased and so the tower performance was improved using any wind break walls. But using eight radial wind break walls is recommended because these walls are not sensitive to wind orientation.

HIGH AMBIENT TEMPERATURE

Another parameter that influence on the cooling tower performance is the ambient dry bulb temperature, because the heat transfer across the heat exchangers occurs via sensible heat transfer only.

As expected, when the air temperature increases the thermal performance and so mass efficiency decreases. Fig. 20 shows the decrement in mass efficiency for natural convection case (no wind) at different ambient temperatures.

As it is shown in Fig. 21 for different wind velocities and ambient temperatures, the mass efficiency follows the same trend at the different wind speeds.

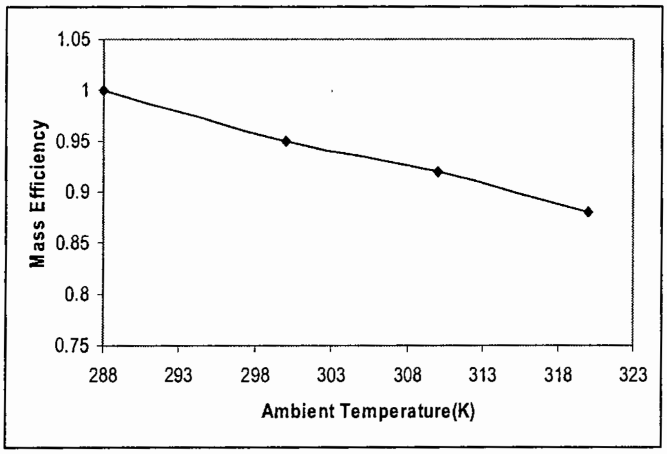


Figure 20: Mass Efficiency Values at Different Ambient Temperatures.

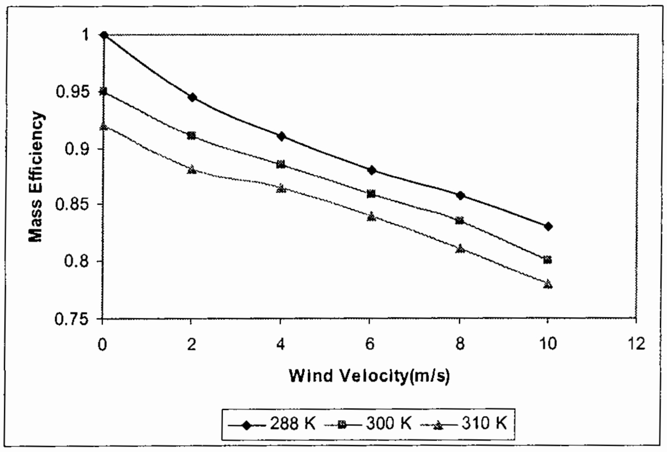


Figure 21: Mass Efficiency Values at Different Wind Velocities and Ambient Temperature.

CONCLUSION

According to the predicted numerical results, changes in the external shape of natural draft dry cooling towers will improve their performance under cross wind condition by removing the vortexes at the tower bottom and increasing the air flux moving through radiators.

Five types of wind break walls were considered and showed that using all of them has a positive impact on tower performance subjected to cross wind. But some of these walls are wind direction dependent and useful for power plants located in regions that have one wind orientation. For regions with different wind orientation using eight radial walls is recommended because this arrangement is not depend on wind direction.

Using wind break walls improved the wind-cover effect too, but already this phenomenon exists at tower outlet. Also our researches showed that changes in the shape of tower outlet or installing special devices at the top of the tower will improve the wind-cover effect and so the cooling efficiency.

As predicted, numerical results showed that when the ambient temperature increases, the cooling tower efficiency decreases. To improve this effect, redundant heat exchangers called "peak cooler" are used in dry cooling towers with water spray systems.

REFERENCES

[1] D. Bergetrom, D. Derkson and K.Rezkallah, Numerical Study of Wind Flow Over a Cooling Tower, Journal of Wind Engineering and Industrial Aerodynamics, Vol. 46-47, pp. 657-664, 1993.

[2] D. Demoren and W. Rodi, Three Dimensional Numerical Calculations of Flow and Plume Spreading Past Cooling Towers, Journal of Heat Transfer, Vol. 109, pp. 113-119, 1987.

[3] FLUENT, User's Guide, FLUENT Incorporated, Lebanon, NH, 1999.

[4] A.F. Du Preez and D. Kroger, the Effect of The Heat Exchanger Arrangement and Wind Break Walls on The Performance of Natural Draft Dry-Cooling Towers Subjected to Cross-Winds, journal of Wind Engineering and Industrial aerodynamics, Vol. 58, pp. 293-303, 1995.

[5] M. Su, G. Tang and S. Fu, Numerical simulation of fluid Flow and Thermal Performance of a Dry-Cooling Tower Under Cross Wind Condition, Journal of Wind Engineering and Industrial Aerodynamics, Vol. 79, No. 3, pp. 289-306, 1999.

[6] R. Al-Waked and M. Behnia, The Performance of Natural Draft Dry Cooling Towers under Cross wind: CFD Study, International Journal of energy Research, Vol. 28, pp. 147-161, 2004.

[7] D.G. Kroger, Air-Cooled Heat Exchanger and Cooling Towers, Thermal Flow Performance Evaluation and Design, Begell House, Inc, New York, 1998.

بهبود عملکرد برجهای خنک کن خشک تحت شرایط باد متقاطع با استفاده از دیوارهای باد شکن (Wind Break Walls)

محمد حسن کیهانی [۱ *]، محمد شاهسون [۲ **]، علی عباس نژاد [۳ *]

***دانشگاه صنعتی شاهرود-دانشکده مکانیک**

****شرکت خدمات مهندسی برق (مشانیر)**

h_kayhani@shahrood.ac.ir

چکیده

برج های خنک کن خشک یکی از متداول ترین برج ها در نیروگاههای موجود در مناطق کم آب می باشد . یکی از مشکلاتی که در رابطه با عملکرد این برج ها مطرح است اثرات شرایط محیطی بر عملکرد آنهاست . یکی از مؤثرترین این عوامل ، وزش باد است . در این مقاله با استفاده از روش حجم محدود ، جریان سیال ، داخل و اطراف این برجها در دو حالت جابجایی طبیعی (بدون وزش باد) و جابجایی اجباری (با وزش باد) مدلسازی شده است وعواملی که باعث افت عملکرد این برجها ، تحت اثر باد می شوند ، معرفی شده و در نهایت تأثیر دیواره های باد شکن *Wind Break Walls* برای جلوگیری از این اثرات مورد ارزیابی قرار خواهد گرفت .

واژه‌های کلیدی: برج خنک کن خشک، باد متقاطع، دینامیک سیالات محاسباتی، *Wind Cover* ، دیوار باد شکن

مقدمه

سیستم های خنک کننده در نیروگاه ها متناسب با چگالش بخار خروجی از توربین، توسط آب یا هوا می توانند به صورت مستقیم یا غیر مستقیم عمل کنند. استفاده از برج های خنک کننده جریان طبیعی خشک تحت شرایط معین از جمله کافی نبودن آب و مشکلاتی همچون جلوگیری از تلفات آب از اهمیت فوق العاده ای برخوردار است. با توجه به مسأله کمبود آب در محل های دورتر از رودخانه ها و دریاچه ها و منابع طبیعی آب، لزوم استفاده از برج های خنک کننده خشک مشهودتر می شود. کشور ایران نیز از نظر در اختیار داشتن رودخانه ها و منابع طبیعی آب در زمره کشورهای نسبتاً خشک قرار دارد. به همین جهت لزوم استفاده از برج های خنک کننده خشک بیش از پیش نمایان می گردد، به ویژه اینکه طرح تبدیل برج های خنک کن تر به خشک برای بعضی از نیروگاه ها در حال بررسی است .

با توجه به موارد فوق، مطالعه و بررسی عملکرد برج های خنک کن و عوامل تأثیر گذار بر آن در طراحی نیروگاه ها حائز اهمیت است. یکی از عوامل بسیار مهم در عملکرد برج های خنک کننده خشک، عوامل محیطی از قبیل دمای هوا و سرعت وزش باد می باشد. این دو عامل گاه اثرات بسیار مخربی بر عملکرد برجها گذاشته و باعث افت شدید راندمان

۱- استادیار .

۲- استادیار .

۳- دانشجوی کارشناسی ارشد .

نیروگاه می شوند. بنابراین شناخت نحوه تأثیرات شرایط محیطی و میزان و شدت این تأثیرات از اهمیت فوق العاده ای برخوردار است. تحقیقات بسیاری در باب تأثیر شرایط فوق انجام شده است، ولی آنچه مسلم است این است که هنوز راه حلی جامع و کامل برای رفع این مشکل ارائه نشده است.

افراد بسیاری تأثیر باد بر روی کارایی برج های خنک کن خشک را مورد بررسی قرار داده اند. در کارهای پیشین، در زمینه شرایط محیطی و خصوصیات هندسی، ساده سازی های زیادی انجام شده است. به عنوان مثال *Bergstrom [4]* جریان اطراف برج خنک کن را به صورت دو بعدی مدل کرده است و یا *Rodi & Demuren [5]* به صورت یک لوله تو خالی که سیال از انتهای آن به برج وارد می شود، در نظر گرفته اند. تمامی این مدل ها تفاوت های زیادی با واقعیت داشتند و به همین جهت جواب های به دست آمده از این مدل ها را نمی توان برای تمام شرایط صادق دانست .

اما مدل های ارائه شده بعدی، مدل های کامل تری بودند که از آن جمله *Spalding & Radoslavjevic [7]* با استفاده از نرم افزار *PHOENICS* شبیه سازی سه بعدی جریان سیال در برج را تحت شرایط باد متقاطع کامل تر کردند. *Kroeger & Du Preez [6]* نیز با استفاده از همان نرم افزار عملکرد برج های خشک از نوع *Hamon-Type* را بررسی کردند. *M.D. SU [7]* و همکارانش جریان سیال در برج های خنک کن نوع هلر *Heller* را تحت شرایط باد متقاطع بررسی کردند. آنها از مدل توربولانسی $k-\varepsilon$ و فرضیه *Boussinesq Eddy Viscosity* برای مدل سازی جریان استفاده کرده اند . *Al-Waked & Behina [8]* نیز استفاده از *Wind Break Wall* را برای بهبود عملکرد برج ها پیشنهاد کرده اند.

در این مقاله عملکرد برج های خنک کن خشک در سه حالت مورد بررسی قرار می گیرد :

الف) جابجایی طبیعی: در این حالت عملکرد برج با استفاده از یک کد که توسط نگارندگان مقاله توسعه یافته است، مورد بررسی قرار می گیرد.

ب) جابجایی اجباری: در این حالت عملکرد برج با استفاده از یک کد تجاری مناسب (*FLUENT*) تحلیل می شود . در این حالت ضمن بررسی جریان جابجایی طبیعی داخل برج ، تأثیر جریان اجباری باد نیز بر آن بررسی می شود .

ج) استفاده از دیوارهای باد شکن .

شبیه سازی جریان

برای مدل سازی جریان به علت وجود جابجایی طبیعی از فرض *Boussinesq* استفاده شده است. به دلیل عدم وجود جریان چرخشی بالا از مدل توربولانس $k-\varepsilon$ استاندارد استفاده شده است . با توجه به ارتفاع مبدل ها از سطح زمین (۲۰ متر) اثرات سطح زمین بر جریان باد نادیده گرفته شده و پروفیل سرعت باد به طور یکنواخت در نظر گرفته شده است. همچنین از تغییرات لحظه ای سرعت باد صرفنظر شده است .

معادلات حاکم

معادلات حاکم بر جریان داخل و اطراف برج خنک کن با توجه به فرض *Boussinesq* و به شکل برداری به صورت زیرساده شده است.

$$\nabla \cdot V = 0 \qquad (۱)$$

$$(V \cdot \nabla)V = -\frac{1}{\rho}\nabla P + \nabla \cdot (\frac{\sigma}{\rho}) - \beta(T - T_a)g + F \qquad (۲)$$

$$\rho(V \cdot \nabla)T = -\nabla \cdot \left[(\Gamma + \Gamma_t)\nabla T\right] + Q \qquad (۳)$$

V، نمایشگر بردار سرعت، T دما و σ تانسور تنش است که به وسیله فرمول زیر بیان می شود:

$$\sigma_{ij} = (\mu + \mu_t) \cdot S_{ij} \qquad (۴)$$

$$S_{ij} = \frac{1}{2}(\frac{\partial V_i}{\partial x_j} + \frac{\partial V_i}{\partial x_i}) \qquad (۵)$$

μ و μ_t به ترتیب ار جهت مولکولی و توربولانس هستند . β ضریب انبساط حجمی و T_a دمای محیط هستند. ترم F در معادله ممنتوم دربرگیرنده افت فشار سیال در هنگام

عبور از رادیاتورها بوده که این ترم فقط در ورودی برج ظاهر می شود .

ترم Q در معادله انرژی ، مقدار حرارات منتقل شده به سیال از رادیاتورهاست . میزان Q در حالت جابجایی طبیعی (بدون وزش باد) دارای مقدار ثابتی بوده و به طور یکنواخت در ناحیه ورود به برج توزیع می شود . اما در هنگام وزش باد ، توزیع Q در ورودی برج دیگر یکنواخت نیست و حرارت داده شده به سیال در نقاط مختلف در ورودی برج متفاوت است . اما به منظور مدل سازی صرفاً جریان داخل برج و نه از نقطه نظر انتقال حرارت ، از این اثر اغماض نموده و یک مقدار متوسط برای Q به طور یکنواخت در ورودی برج منظور شده است . چون با توجه به بررسی های انجام شده آنچه که باعث افت عملکرد و در نتیجه توزیع غیر یکنواخت Q می شود ، اختلال در جریان ورودی برج است . همچنین به دلیل مخلوط شدن هوا بلافاصله پس از ناحیه ورودی (رادیاتورها) وضعیت جریان داخل برج تا حوالی ناحیه خروجی آن تقریباً به صورت متقارن محوری در می آید .

Γ و Γ_t در معادله انرژی بیانگر ضریب انتقال حرارت هدایتی مولکولی و توربولانس هستند .

$$\Gamma = \frac{\mu}{\Pr} \qquad \Gamma_t = \frac{\mu}{\Pr_t}$$

Pr و $\Pr_t$ به ترتیب اعداد پرانتل سیال و توربولانس می باشند .

جریان داخل و اطراف برج ، یک جریان مغشوش است و با استفاده از مدل $k-\varepsilon$ استاندارد معادلات توربولانس به صورت زیر بیان می شوند:

$$(V\cdot\nabla)k = \nabla.\left[(\upsilon + \upsilon_t/\sigma_k)\nabla k\right] + P + G - \varepsilon \qquad (۶)$$

$$(V\cdot\nabla)\varepsilon = \nabla.\left[(\upsilon + \upsilon_t/\sigma_\varepsilon)\nabla k\right] +$$

$$c_{1\varepsilon}\frac{\varepsilon}{k}(P+G) - c_{2\varepsilon}\frac{\varepsilon^2}{k} \qquad (۷)$$

$$\upsilon_t = \frac{1}{\rho}\mu_t = c_\mu \frac{k^2}{\varepsilon} \qquad (۸)$$

P انرژی سینتیک تولید شده به وسیله توربولانس و G به وسیله نیروی غوطه وری هستند و ضرایب ثابت در مدل توربولانس به شرح زیر هستند:

$$c_\mu = 0.09 \qquad c_{1\varepsilon} = 1.44 \qquad c_{2\varepsilon} = 1.92$$

$$\sigma_k = 1 \qquad \sigma_\varepsilon = 1.3$$

فضای محاسباتی و شرایط مرزی

مدل مورد استفاده ، مدل یک برج واقعی در مقیاس صنعتی با ابعاد زیر است :

قطر پائین برج : ۱۱۰ متر.

ارتفاع برج : ۱۳۰ متر.

قطر گلوگاه : ۶۲ متر.

حرارت دفع شده : ۴۰۴ مگاوات .

فضای اطراف برج همان طور که در شکل (۱) ملاحظه می شود به صورت یک فضا به شکل مکعب مستطیل در اطراف برج است که دارای یک مرز ورود سرعت و یک مرز خروجی *Pressure Outlet* است که سرعت مرز ورودی همان سرعت باد است . شرط مرزی دیوار (عدم لغزش) برای دیوار بدنه برج در نظر گرفته شده است . شرط مرزی برای رادیاتورها که یک مرز داخلی هستند به صورت یک محیط متخلخل (*Porous Media*) در نظر گرفته شده است . افت فشار در رادیاتورها متناسب با هد دینامیکی هوا در نظر گرفته شده است که به صورت زیر بیان می شود :

$$\Delta P = K_L \frac{1}{2}\rho v^2 \qquad (۹)$$

ρ دانسیته سیال و k_L یک ضریب افت بدون بعد است . برای محاسبه k_L در محیط *Porous* از نتایج تجربی مشابه موجود بهره گرفته شده است .

نتایج

برای مشاهده نتایج سه حالت در نظر می گیریم :

الف) جابجایی طبیعی .

ب) جابجایی اجباری .

ج) استفاده از دیوارهای باد شکن .

الف) جابجایی طبیعی :

در حالت جابجایی طبیعی ، جریان داخل برج با استفاده از یک کد که توسط نگارندگان مقاله توسعه داده شده حل شده است . جریان در این حالت متقارن محوری است .

در این حالت ، ابتدا شبکه لازم با استفاده از روش تبدیلات ایزوپارامتریک تولید می شود . شبکه سازی دو بعدی اشاره شده توسط زینکوویچ و فیلیپس *[2]* معرفی شده است . شبکه سازی شامل مراحل تعریف نواحی ، تقسیم بندی بلوکی نواحی ، تقسیم بندی بلوک ها ، ایجاد شماره گرها ، ایجاد مختصات گره ای و تشکیل جدول اتصال اجزاء می باشد . شبکه تولید شده توسط این روش یک شبکه ۳۶×۱۱ می باشد که در شکل (۲) مشاهده می شود .

سپس مختصات شبکه به دست آمده به صورت ورودی به کد حل کننده که بر اساس روش حجم محدود و به زبان *FORTRAN* نوشته شده است ، داده می شود . بردارهای سرعت و کانتورهای دما و فشار در این حالت در شکل های (۳) و (۴) و (۵) مشاهده می شوند .

ب) جابجایی اجباری :

در این حالت از روش حجم محدود برای گسسته سازی معادلات و روش ضمنی برای حل معادلات گسسته شده جبری استفاده شده است . از الگوریتم *SIMPLE* برای محاسبه فشار و بنابراین میدان جریان که برای حل معادله انرژی مورد نیاز است ، استفاده می شود .

در این حالت ، جریان سه بعدی بوده و دارای یک صفحه تقارن موازی با جهت باد است . شکل (۶) بردارهای سرعت در صفحه تقارن را تحت شرایط باد با سرعت *5 m/s* نشان می دهد . شکل (۷) نیز کانتورهای دما در این حالت را در صفحه تقارن نشان می دهد .

با ملاحظه شکل (۶) مشاهده می شود که دبی ورودی هوا در رادیاتورهایی که در جهت وزش باد قرار گرفته اند افزایش یافته و در رادیاتورهایی که باد مماس بر آنها حرکت می کند کاهش یافته و در پشت برج حالت سکون به وجود می آید . همان طور که مشاهده می شود و نیز *Al-Waked* و *Behina [8]* اشاره کرده اند در سرعت های کم باد ، ایجاد جدایش در بالای برج باعث کاهش فشار در بالای برج می شود و این عامل باعث افزایش دبی عبوری از برج شده و یک اثر مثبت در عملکرد آن محسوب می شود .

اما در سرعت های بالا (مثلاً *10 m/s*) ، پدیده درپوش (*Wind Cover*) ایجاد می شود .

علت ایجاد پدیده درپوش، تفاوت اندازه حرکت و جهت سیال خروجی از برج و سیالی است که با سرعت باد در بالای برج به طور موازی با افق حرکت می کنند. این پدیده نیز باعث کاهش دبی هوای خروجی از برج شده و در نهایت باعث افت عملکرد برج می شود .

بردارهای سرعت در صفحه افقی و در ارتفاع ۱۰ متری از سطح زمین در شکل (۸) نشان داده شده اند. ملاحظه می شود که یک جفت ورتکس و جریان چرخشی در این صفحه وجود دارد که ناشی از برخورد جریانی است که از سمت روبروی باد و سمت پشت برج وارد می شوند . به علت شدت های متفاوت این دو جریان در برخورد آنها ورتکس هایی ایجاد می شوند . جریان خارج از رادیاتورها مانند جریان روی یک استوانه توپر است .

هنگام عبور از جریان روی قسمت های کناری برج (مماس بر جهت باد)، فشار استاتیکی کاهش یافته و این عامل باعث کاهش اختلاف فشار بین بالا و پایین برج شده و در نتیجه دبی هوای ورودی به برج کم شده که این عامل نیز باعث افت عملکرد برج می شود .

بردارهای سرعت در صفحه تقارن و و کانتورهای دما وبردارهای سرعت در صفحه افقی در ارتفاع ۱۰ متری در حالت سرعت باد *10 m/s* به ترتیب در شکل های (۹) ، (۱۰) و (۱۱) مشاهده می شوند .

با مقایسه بردارهای سرعت در دو حالت سرعت باد *5 m/s* و *10 m/s* ملاحظه می شود که با افزایش سرعت باد ، پدیده چرخش جریان در پایین برج و پدیده درپوش در بالای برج شدت بیشتری یافته و در نتیجه با افزایش سرعت باد ، عملکرد حرارتی برج شدیداً افت می کند .

ج) استفاده از دیوارهای بادشکن(*Wind Break Wall*):

استفاده از دیوارهای باد شکن ابتدا توسط *Kroeger[6]* معرفی شد، بدینگونه که وی و همکارانش ، دیوار باد شکن

را به صورت یک دیوار متخلخل در مرکز برجهایی که رادیاتورها به طور افقی قرار گرفته اند بکار بردند.طبق گزارش آنها نتایج بدست آمده تطابق بسیار خوبی با نتایج واقعی اندازه گیری شده داشت واستفاده از این نوع دیوارها را روندی مثبت در عملکرد برج تحت شرایط باد متقاطع دانسته اند، همچنین گونه ای دیگر از دیوارهای باد شکن توسط *Al-waked & Behnia[8]* بکار برده شده است.

آنها دیوارهای بادشکن را به تعداد ۸ دیوار مستطیل شکل و با فواصل یکسان در محیط خارجی برج و در ورودی آن نصب کردند که طبق اظهارات آنها ، استفاده از این نوع دیوارها نیز اثرات مثبتی بر عملکرد برجها داشته است.

در این مقاله دو نوع دیوار بادشکن مورد بررسی قرار خواهد گرفت. نوع اول که در شکل (۱۲) مشاهده می شود به صورت دو دیوار منحنی شکل در نظر گرفته شده است. همانطور که در شکل (۱۱) مشاهده می شود و *M.D.Su[7]* و همکارانش نیز اشاره کرده هاند در زوایای ۱۲۰-۹۰ درجه به علت کاهش فشار استاتیکی در ورودی برج ، اختلاف فشار بین پایین و بالای برج کم شده و این عامل باعث کاهش مکش و در نتیجه کاهش انتقال حرارت می شود.لذا به همین دلیل دو دیوار ذکر شده در زاویه ۱۲۰ درجه نصب شده اند.

با توجه به این که میزان دبی جرمی هوایی که از داخل رادیاتورها و در نتیجه از داخل برج عبور می کند، (در یک مقدار حرارت ثابت در ورودی برج) می تواند به عنوان شاخصی مناسب از انتقال حرارت و در نتیجه عملکرد برج باشد ، با معرفی یک ضریب تاثیر ε که برابر با نسبت دبی جرمی هوای عبوری از داخل برج در حالتهای مختلف (به عنوان مثال تحت شرایط باد با سرعت *10m/s* و یا نصب دیوار باد شکن) به دبی جرمی هوای عبوری از برج تحت شرایط جابجایی طبیعی (بدون وزش باد) است ، حالتهای مختلف مورد بررسی قرار خواهند گرفت.

$$\varepsilon = \frac{\dot{m}}{\dot{m}_{nat.convection}} \qquad (۱۰)$$

علاوه بر دیوارهای بادشکن که ابتدا معرفی شد ، دو دیوار منحنی شکل دیگر نیز بطوریکه در شکل (۱۳) مشاهده می شود در پشت برج نصب شده است . جدول (۱) مقدار ضریب تاثیر برای حالتهای مختلف را نشان می دهد ، لازم به ذکر است که تاثیر دیوارهای باد شکن درسرعت باد *10m/s* بررسی شده است . با ملاحظه مقادیر ضریب تاثیر در حالتهای مختلف نتیجه می شود که این تغییر در سطح خارجی برج در بهبود عملکرد آن موثر بوده است .

نتیجه گیری

با توجه به نتایجی که ذکر شد، افت عملکرد برج ناشی از عوامل زیر می باشد :

۱- ایجاد ورتکس در پایین برج به دلیل تفاوت سرعت های جریان های ورودی از سمت مقابل باد و پشت آن .

۲- کاهش دبی ورودی هوا در قسمت کناری برج .

۳- ایجاد پدیده *Wind Cover* به علت تفاوت اندازه حرکت جریان خروجی از برج و جریان باد .

همانگونه که ملاحظه شد ایجاد تغییراتی در شکل سطح خارجی برج می تواند عاملی جهت بهبود عملکرد آن تحت شرایط باد متقاطع باشد .

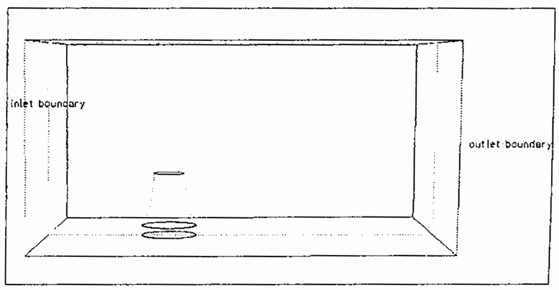
گرچه برای حل برخی از مشکلات ذکر شده راه حلهایی مانند استفاده از دیوارهای باد شکن ارائه شده است ، لیکن اولاً استفاده از این راه حل ها محدود بوده و تحقیقات بر روی شکل ، اندازه ، جنس و محل نصب این دیوارها هنوز ادامه دارد . ثانیاً هیچ راه حلی برای پدیده درپوش *Wind Cover* که در برج های بلندتر محسوس تر است ، پیش بینی نشده است . بررسی های اولیه انجام شده در این پژوهش حکایت از آن دارد که ایجاد تغییرات در شکل سطح خارجی برج می تواند در جلوگیری از این پدیده مؤثر باشد . لیکن نتیجه گیری قطعی در این مورد ، نیازمند بررسی های بسیار گسترده تر می باشد .

منابع

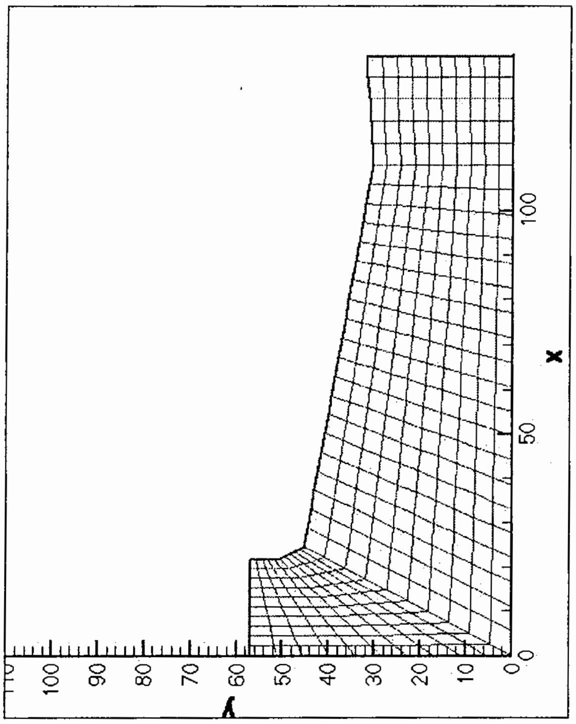
1. C. Hirsch , Numerical Computation of Internal and Extrenal Flows , Vol .2 , JOHN WILY & SONS , 1990 .

2. O.C . Zienkiewicz, and D.V.Philips , " An Automatic Mesh Generation Scheme for Plane and Curved Surface by Isoparametric Coordinates " , International Journal for Numerical Methode in Engineering , Vol.3,519-328 (1971) .

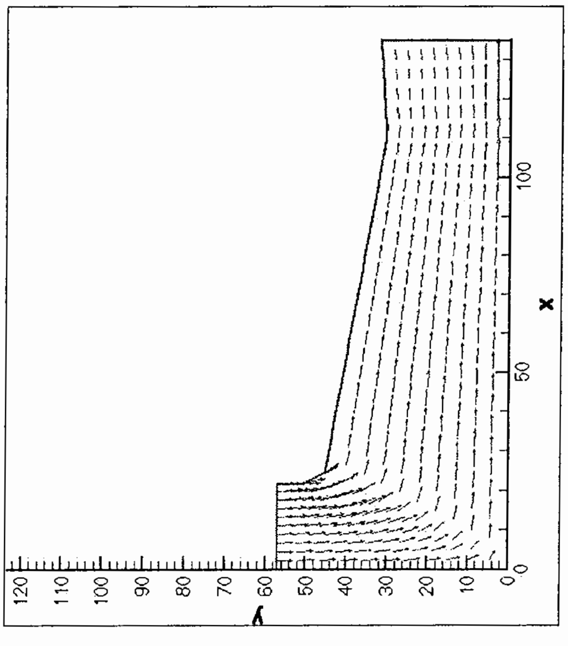
3. J.C. Tennhill , D. A. Anderson , R.H. Pletcher , Computational Fluids Mechanics and Heat Transfer , Taylor & Francis , 2nd Edition , 1997 .
4. D. Bergestrom,D. Derkson,K. Rezkallah, " Numerical Study of Wind Flow Over a Cooling Tower " , Journal of Wind Engieering and Indusrial Aerodynamics , 46-47: 657-664,1993 .
5. D.Demuren.,W.Rodi,"Three Dimensional Numerical Calculations of Flow and Plume Spreading Post Colling Towers " J. Heat Transfer , 109,113-119,1987.
6. A.F. Du Preez,D.G. Kroeger., " The Effect of The Heat Exchanger Arrangement and Wind Break Walls on The Performance of Natural Draft Dry-Cooling Towers Subjected to Cross-Winds " Journal of Wind Engineering and Industrial Aerodynamics " , 58:293-303 , 1995 .
7.M.Su,G.Tang,S.Fu,"Numerical Simulation of Fluid Flow and Thermal Performance of a Dry-Cooling Tower Under Cross Wind Condition " , J. of Wind Engineering and Industrial Aerodynamics , Vol.79 , No.3 , 289-306 , 1999 .
8.R.AL-Waked,M.Behina,"The Performance of Natural Draft Cooling Towers Under Cross Wind : CFD Study " , International Journal of Energy Research , 28: 147-161,2004 .
8. D.G.Kroeger,Air-Cooled heat Exchanger and Cooling Towers , Thermal-Flow Performance Evaluation and Design , Begell House , Inc : New York , 1198 .



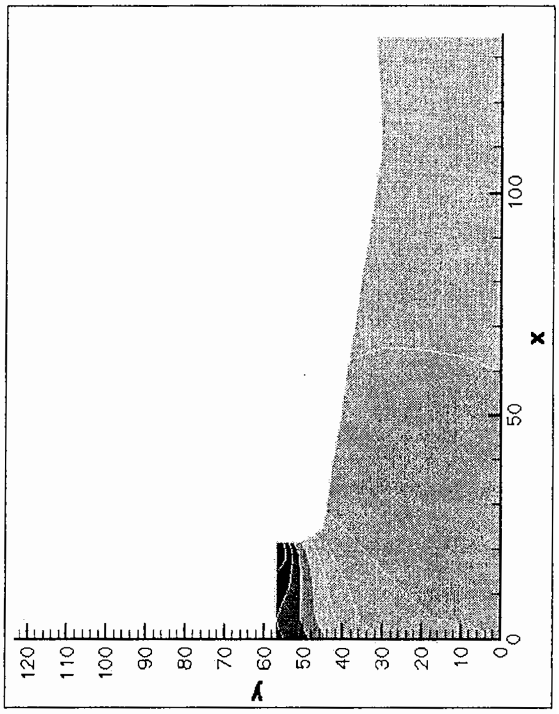
شکل ۱- فضای محاسباتی



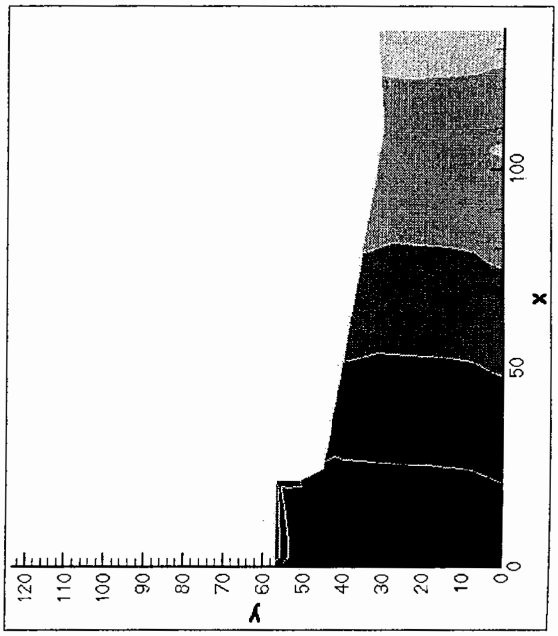
شکل ۲- شبکه تولید شده در حالت جابجایی طبیعی



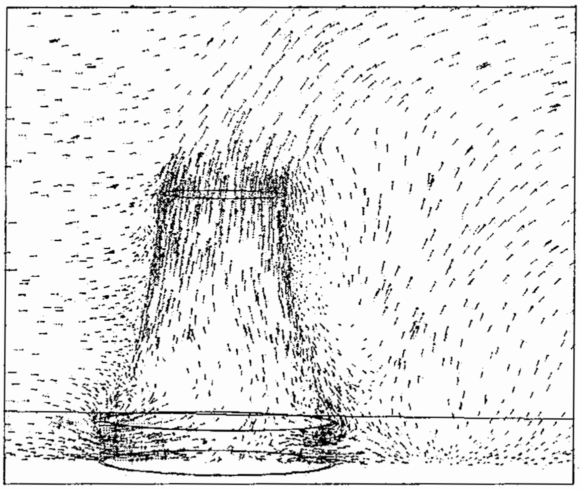
شکل ۳ - بردارهای سرعت در حالت جابجایی طبیعی



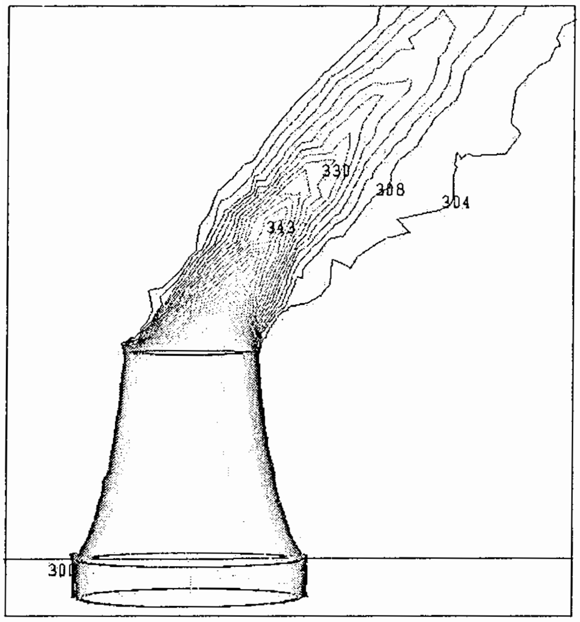
شکل ۴ - کانتورهای دما در حالت جابجایی طبیعی



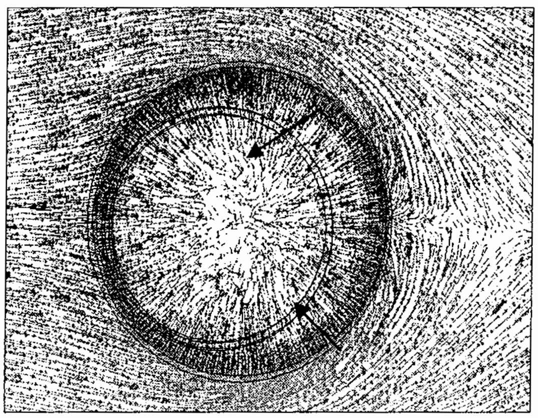
شکل ۵ - کانتورهای فشار در حالت جابجایی طبیعی



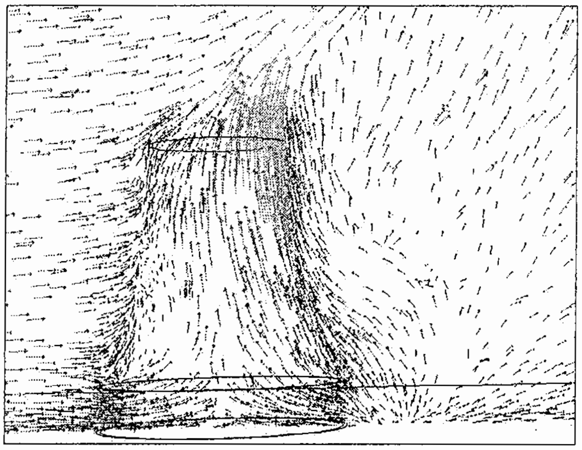
شکل ۶ - بردارهای سرعت در صفحه تقارن ، سرعت باد *5 m/s*



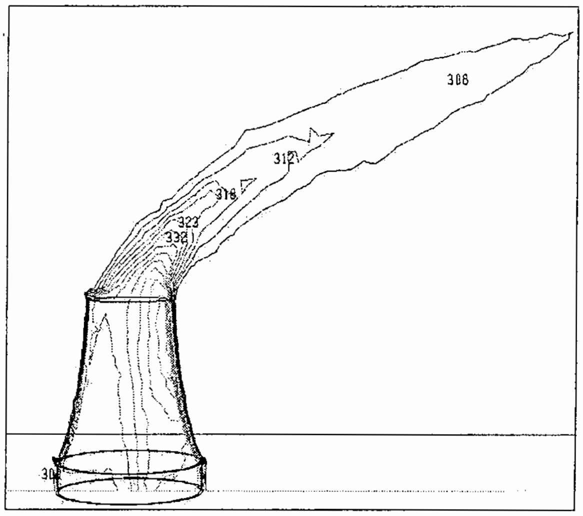
شکل ۷ - کانتورهای دما در صفحه تقارن ، سرعت باد *5 m/s*



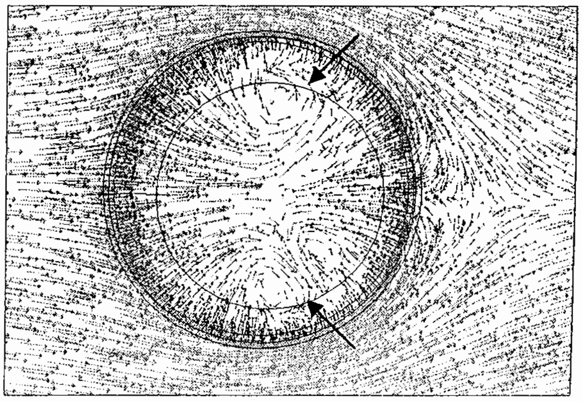
شکل ۸ - بردارهای سرعت در صفحه افقی ، سرعت باد *5 m/s*



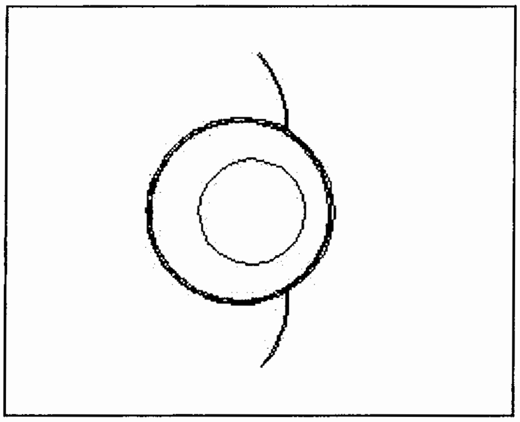
شکل ۹ -بردارهای سرعت در صفحه تقارن ، سرعت باد *10 m/s*



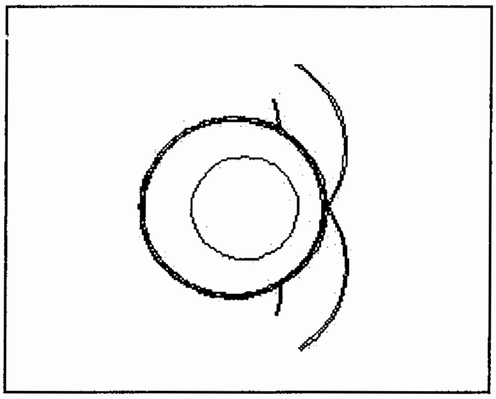
شکل ۱۰ - کانتورهای دما در صفحه تقارن ، سرعت باد *10 m/s*



شکل ۱۱ - بردارهای سرعت در صفحه افقی ، سرعت باد *10m/s*



شکل ۱۲- استفاده از دو دیوار باد شکن در زاویه ۱۲۰ درجه



شکل ۱۳ – استفاده از چهار دیوار باد شکن

جدول ۱ – میزان ضریب تاثیر در حالتهای مختلف

ضریب تاثیر	وضعیت
۱	جابجایی طبیعی
۰/۸	باد متقاطع با سرعت 10m/s
۰/۸۶	استفاده از دو دیوار باد شکن
۰/۸۸	استفاده از چهار دیوار باد شکن

EUROTHERM 82
Numerical Heat Transfer 2005
September 13-16, 2005, Gliwice-Cracow, Poland
Eds.: A. Nowak, R.A. Białecki

Cross Wind and Natural Draft Dry Cooling Towers: Effects of Different Wind Break Walls

M. H. Kayhani * , M. Shahsavan , A. Abbasnejad

Shahrood University of Technology, Shahrood, Iran
e-mail: h_kayhani@shahrood.ac.ir

Key words: cross wind, Dry cooling tower, CFD, wind break wall.

Abstract
The purpose of this paper is to study the effect of different types of wind break walls on the performance of natural draft dry cooling towers. The performance of cooling towers under cross wind was investigated numerically. Using a general purpose CFD code, a three dimensional fluid flow and temperature distribution around and in a cooling tower were simulated. In absence of high circulating, k-ε turbulence model was used for turbulent flow modeling. The performance losses reasons was discussed in brief and then, the performance of cooling tower under cross wind with using different types of wind break walls was investigated. The results have indicated a good improvement in reducing the performance losses due to cross wind.

1 Introduction

Natural draft dry cooling towers are known for their advantages as water resource saving, protecting the environment, pollution free. But it is found that the performance and cooling efficiency of cooling towers are seriously dependent on the environmental conditions, such as ambient temperature, cross wind speed, humidity, inversions and rain.

In summer and wind seasons the cooling efficiency of the tower is evidently reduced and the electricity production by power plant reduced to a large extent due to condenser vacuum decrease. Because of importance of wind effect on dry cooling towers performance, many investigations have been done either numerically and experimentally.

Bergstom et al. [1] did the first 2D simulation of fluid flow in cooling tower. Demoren and Rodi [2] modeled the cooling tower as an empty tube with injecting heated fluid from its bottom. Using PHOENICS software, Du Preez and Kroger [4] studied the effect of wind on the homon type cooling tower performance. They used a porous wall at the center of tower as wind break wall for reducing the unfavorable wind effect.

Su et al. [5] numerically studied the performance of the Heller dry cooling tower under cross wind. Al-Waked and Behnia [6] studied the effect of the cross wind on thermal performance of Homon-type cooling towers, using FLUENT code. They employed wind break walls to improve the cooling tower performance under cross wind condition.
The aim of this research work is to study the impact of wind break walls on the performance of towers under cross wind condition. Finite volume method is used to simulate the fluid flow and temperature distribution around and in a Heller type dry cooling tower. In absence of high swirling flow, k-ε model was employed for turbulent flow modeling. A new parameter, "mass efficiency" or "hydrodynamic efficiency" was introduced to compare different cases.
The major objective of this research is to study the impact of different types of wind break walls on the natural draft dry cooling tower performance under cross wind. Firstly the reasons of performance losses are discussed briefly and then different arrangements of wind break walls and their advantages and disadvantages were studied. Finally the performance improvement using these walls is discussed.

2 Effect of Wind

Measurements performed on natural draft cooling towers subjected to cross winds indicate a rise in outlet water temperature with increasing wind speed for a given heat rejection rate. In a dry cooling tower change in approach temperature (difference between outlet water temperature and air entering the tower) reads:

$$\Delta T_{WO} = \Delta(T_{WO} - T_{ai}) = (T_{WO} - T_{ai})_W - (T_{WO} - T_{ai}) \tag{1}$$

Fig 1 shows the rise in cooling water temperature (approach temperature) versus cross wind speed for six power plants.

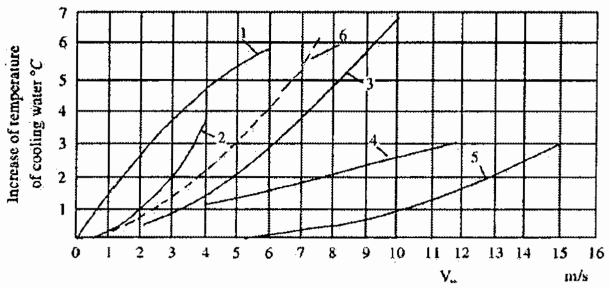


Figure 1: Rise in Cooling Temperature for different power plants: 1) Lazdain in Russia; 2) Ibenbiron in Germany; 3) Kakalin in Hungary; 4) Grud fry No.5 in South Africa; 5) Grud fry No.6 in South Africa.

It is clear that for all above mentioned power plants, the approach temperature increases with increasing wind speed. Kroger [7] mentioned that cooling towers with horizontally heat exchanger arrangement are less sensitive to wind.
For determining the tower performance losses under cross wind condition, numerical simulation of the fluid flow has been performed, using FLUENT code. In this case the algebraic equations of physical phenomena are discretized and solved by an implicit method and a segregated solver. The SIMPLE algorithm is used for the calculation of the pressure and therefore the velocity field.

To investigate the performance of cooling towers in aforementioned models a real industrial scale cooling tower was used with following specifications is used.
Domain is consists of a rectangular parallelepiped with 700×400×400 m dimensions, around the tower. The geometry is shown in Fig. 2.
Fig. 3 shows the velocity vectors in symmetric plane for 10 m/s cross wind velocity for forced convection model. Contours of temperature in symmetry plane are shown in Fig. 4. Fig. 5 shows the velocity vectors in the horizontal plane at the level of 10 m and in this Fig. there is a couple of vortexes in the tower, which are formed due to the convergence of two flows through the front part and back part of the tower with different intensities.

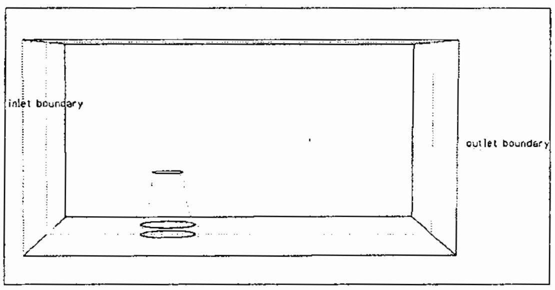


Figure 2: Geometry of Analyzed Tower.

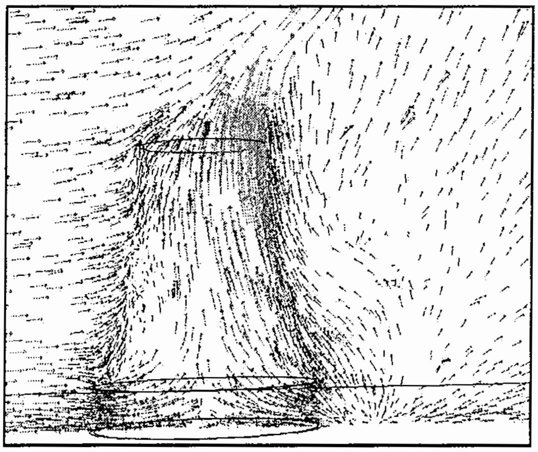


Figure 3: Velocity Vectors in Symmetry Plane.

The flux of the air entering into the tower is reduced greatly in the side part of the tower (outside the radiators) and so the cooling efficiency decreases because the air tangential velocity is lower similar to flow over a cylinder. According to Fig. 5, the flux of air entering the radiators increases in the front part of the tower (facing the wind).
From Fig. 3 it is clear that there is a "wind-cover" over the tower. This phenomenon is occurred because the air exit velocity of tower is less than the wind velocity moving over the tower. Wind-

cover acts as a cap over the tower. With increasing wind velocity, this phenomenon becomes more and more strong.

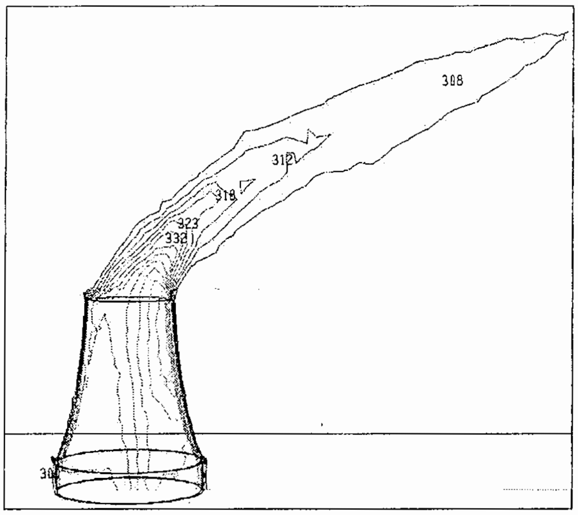


Figure 4: Contours of Temperature in Symmetry Plane.

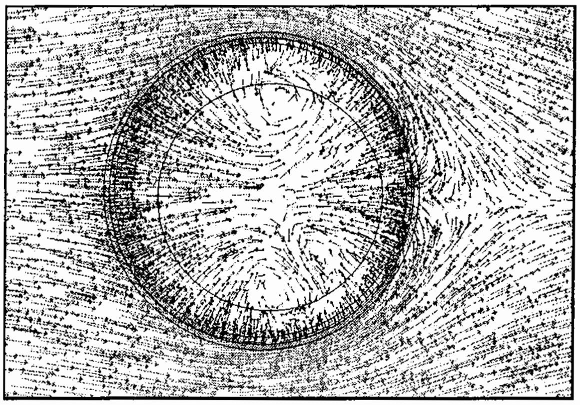


Figure 5: Velocity Vectors in Horizontal Plane at the Level of 10 m.

The flux of air moving through radiators can be an important factor for heat transfer from radiators and thus cooling efficiency. So the new parameter, η, named "mass efficiency" or "hydrodynamic efficiency" that is the ratio of air flux at different conditions such as different wind speeds and high ambient temperature to nominal condition air flux (natural convection), is a suitable parameter for cooling performance investigation.

$$\eta = \frac{\dot{m}}{\dot{m}_{nat.convection}} \quad (2)$$

Fig. 6 indicates the values of η for different wind velocities. It can be seen that η decreases when the wind velocity increases. When the value of η decreases, the air flux moving through radiators

decreases and the radiators can not work in nominal condition, so the cooling efficiency and finally the electricity production reduces.
According to the predicted numerical results the reasons for loss in performance of natural draft dry cooling towers are:
1) Vortex production in the bottom of tower.
2) Decreasing in flux of air entering the tower from side part.
3) Difference between intensities of the air flow exits from cooling tower and wind flow at top of the tower that causes "wind-cover".

3 Wind Break Walls

Wind break walls first were introduced by du Preez and Kroger [4], and they applied a porous wall in the centre of a Homon-type cooling tower (inside the tower) as a wind break wall.

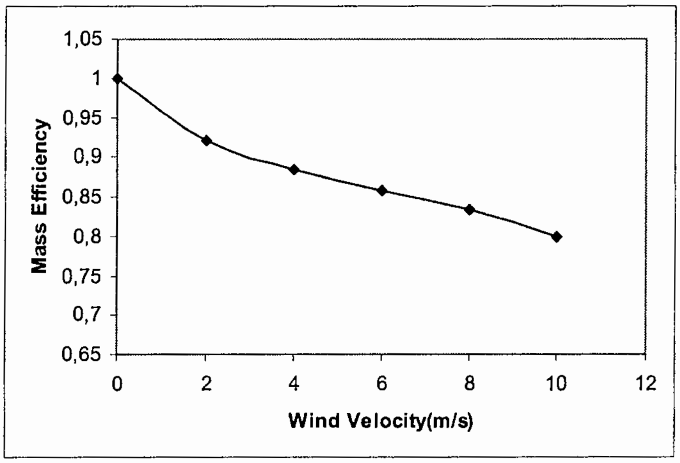


Figure 6: Mass Efficiency Values in Different Wind Velocities.

Another type of wind break walls was applied by Al-Waked & Behnia [6]. They used eight rectangular shaped walls on the outside of the Homon type cooling tower. Their work shows the good improvement in the performance of the tower under cross wind condition.
In this paper five different types of wind break walls were introduced and employed for cooling tower performance improvement these wind break wall types are shown in Figs 7 to 11. Figures 7 and 8 show using two and four curve shaped wind break walls. As mentioned in last part, the air flux decreases in the side part of the tower and thus the walls located at the $\theta = 120°$ where θ measured from wind direction. So the walls were considered in θ=120°.
In Fig. 9 the walls are normal to the inlet boundary of the tower. All aforementioned walls are dependent to wind direction. But we apply other two wall types that are not dependent on wind direction. Fig. 10 shows two walls which installed at θ=90.
As shown in Fig 11 eight wind break walls are located radially at the tower inlet. This walls arrangement is not depend on wind direction and it is an advantage for this arrangement.

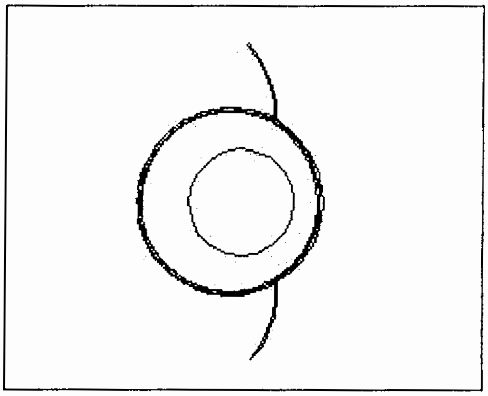


Figure 7: Two Curve Shaped Wind Break Walls.

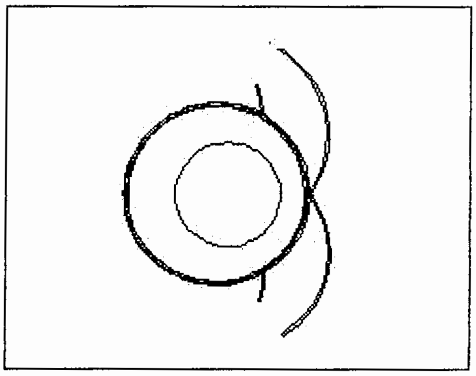


Figure 8: Four Curve Shaped Wind Break Walls.

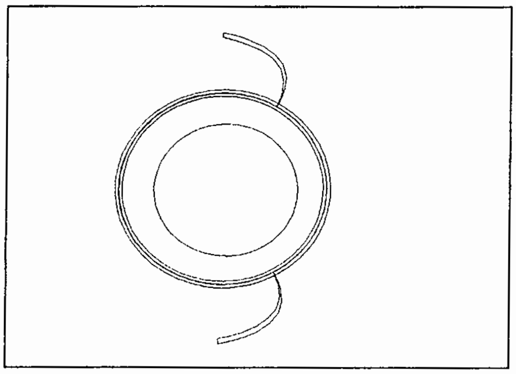


Figure 9: Two Curve Shaped Wind Break Walls (normal to tower inlet).

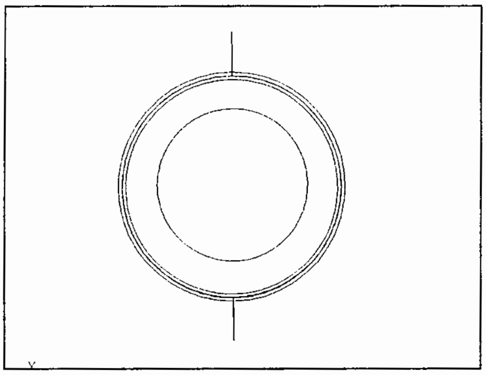


Figure 10: Two Radial Wind Break Walls.

4 Results and Discussions

To investigate the performance of cooling towers in aforementioned models a real industrial scale cooling tower was used with following specifications is used.
Bottom diameter: 110 m
Tower height: 130 m
Throat diameter: 62 m
Radiator height: 20 m
Ejected heat: 404 MW

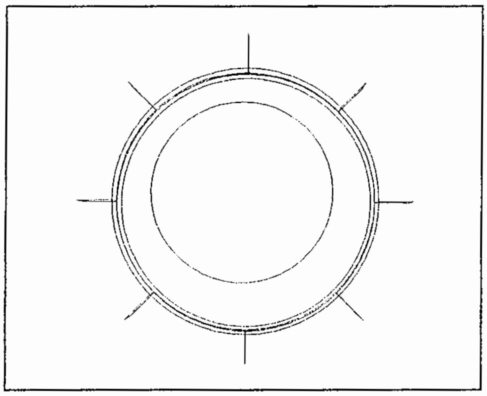


Figure 11: Eight Radial Wind Break Walls

Domain is consists of a rectangular parallelepiped with 700×400×400 m dimensions, around the tower. The wind velocity was used to define the inlet boundary condition and the air static pressure at the flow outlet boundary for forced convection case (windy condition).
The no slip principle was used for solid regions such as wind break walls, tower shell and ground. The pressure drop through radiators is assumed to be proportional to the dynamic head of air and for modelling radiators; a porous zone with thermal source term is selected at the inlet boundary of the tower. Three dimensional fluid flow has a symmetric plane, parallel to the wind direction.

The velocity vectors in horizontal plane at the level of 10 m were shown in Fig. 12. According to this Fig., it is clear that the aforementioned couple of vortexes become weaker, so the air flux entering the tower and moving through radiators increases at the side part of the tower.

Fig. 13 shows the velocity vectors in symmetry plane when eight radial wind break walls were used. It can be seen that the wind cover effect is weaker too.

Therefore the air flux entering the tower and moving through radiators increases and then the value of η increases. Different values of η were shown in Fig. 14 and Fig. 15 for twotype s of wind break walls.

According to these Figs the tower performance improves with using wind break walls.

For all wind velocities the mass efficiency was increased and so the tower performance was improved using any wind break walls. But using eight radial wind break walls is recommended because these walls are not sensitive to wind orientation.

5 Conclusion

According to the predicted numerical results, changes in the external shape of natural draft dry cooling towers will improve their performance under cross wind condition by removing the vortexes at the tower bottom and increasing the air flux moving through radiators.

Five types of wind break walls were considered and showed that using all of them has a positive impact on tower performance subjected to cross wind. But some of these walls are wind direction dependent and useful for power plants located in regions that have one wind orientation. For regions with different wind orientation using eight radial walls is recommended because this arrangement is not depend on wind direction.

Using wind break walls improved the wind-cover effect too, but already this phenomenon exists at tower outlet. Also our researches showed that changes in the shape of tower outlet or installing special devices at the top of the tower will improve the wind-cover effect and so the cooling efficiency.

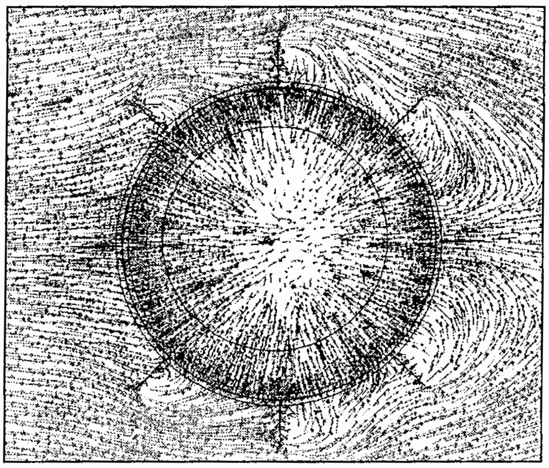


Figure 12: Velocity Vectors in Horizontal Plane Using Eight Radial Wind Break Walls..

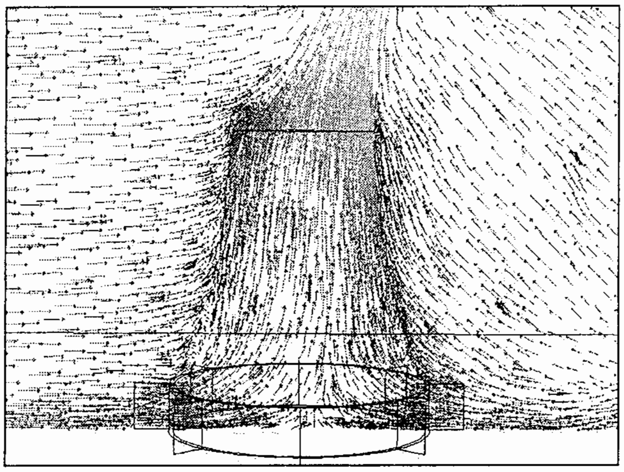


Figure 13: Velocity Vectors in Symmetry Plane Using Eight Radial Wind Break Walls.

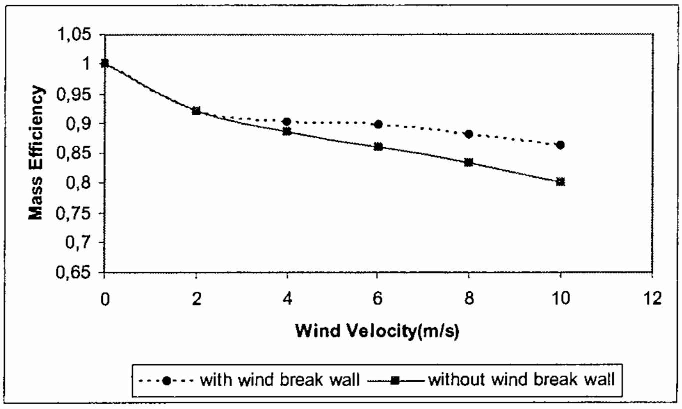


Figure 14: Mass Efficiency Values Using Eight Radial Walls.

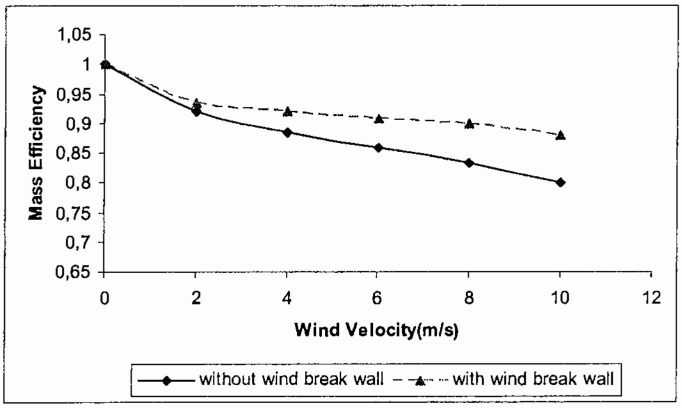


Figure 15: Mass Efficiency Values Using Four Curved Walls.

References

[1] D. Bergetrom, D. Derkson and K.Rezkallah, *Numerical Study of Wind Flow Over a Cooling Tower*, Journal of Wind Engineering and Industrial Aerodynamics, Vol. 46-47, pp. 657-664, (1993).

[2] D. Demoren and W. Rodi, *Three Dimensional Numerical Calculations of Flow and Plume Spreading Past Cooling Towers*, Journal of Heat Transfer, Vol. 109, pp. 113-119, (1987).

[3] FLUENT, User's Guide, FLUENT Incorporated, Lebanon, NH, (1999).

[4] A.F. Du Preez and D. Kroger, *the Effect of The Heat Exchanger Arrangement and Wind Break Walls on The Performance of Natural Draft Dry-Cooling Towers Subjected to Cross-Winds,* Journal of Wind Engineering and Industrial aerodynamics, Vol. 58, pp. 293-303, (1995).

[5] M. Su, G. Tang and S. Fu, *Numerical simulation of fluid Flow and Thermal Performance of a Dry-Cooling Tower Under Cross Wind Condition*, Journal of Wind Engineering and Industrial Aerodynamics, Vol. 79, No. 3, pp. 289-306, (1999).

[6] R. Al-Waked and M. Behnia, *The Performance of Natural Draft Dry Cooling Towers under Cross wind: CFD Study*, International Journal of energy Research, Vol. 28, pp. 147-161, (2004).

[7] D.G. Kroger, *Air-Cooled Heat Exchanger and Cooling Towers, Thermal Flow Performance Evaluation and Design*, Begell House, Inc, New York, (1998).

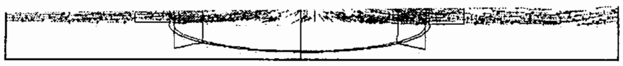


Figure 13: Velocity Vectors in Symmetry Plane Using Eight Radial Wind Break Walls.

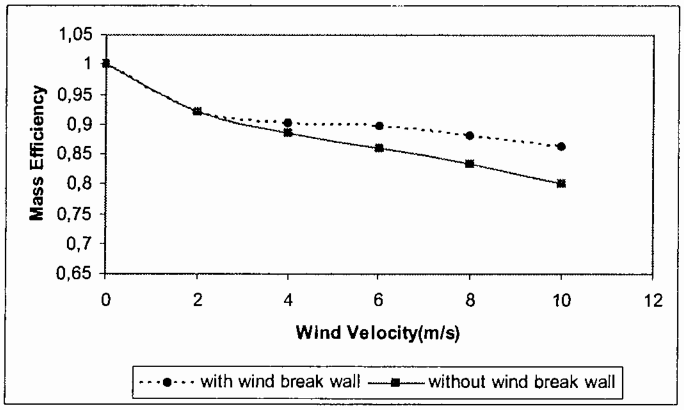


Figure 14: Mass Efficiency Values Using Eight Radial Walls.

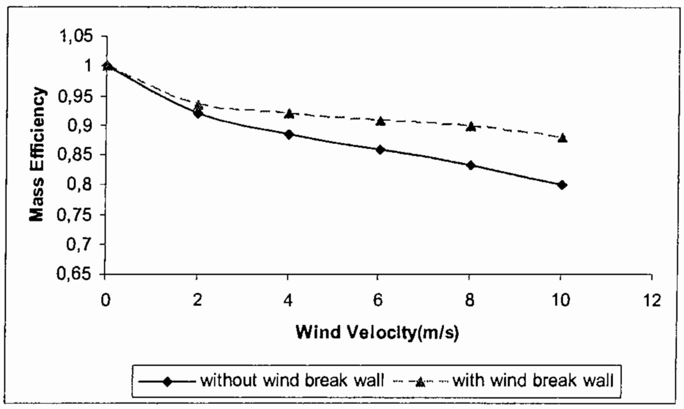


Figure 15: Mass Efficiency Values Using Four Curved Walls.

Journal of Wind Engineering and Industrial aerodynamics, Vol. 58, pp. 293-303, (1995).

[5] M. Su, G. Tang and S. Fu, *Numerical simulation of fluid Flow and Thermal Performance of a Dry-Cooling Tower Under Cross Wind Condition*, Journal of Wind Engineering and Industrial Aerodynamics, Vol. 79, No. 3, pp. 289-306, (1999).

[6] R. Al-Waked and M. Behnia, *The Performance of Natural Draft Dry Cooling Towers under Cross wind: CFD Study*, International Journal of energy Research, Vol. 28, pp. 147-161, (2004).

[7] D.G. Kroger, *Air-Cooled Heat Exchanger and Cooling Towers, Thermal Flow Performance Evaluation and Design*, Begell House, Inc, New York, (1998).

Title:

A Study of Dry Cooling Towers Performance under Cross Wind and High Ambient Temperature Effects Using Numerical Analysis

Supervisor:
Dr. Mohammad Hassan Kayhani

Consultant Supervisor:
Dr. Mohammad Shahsavan

Student:
Ali Abbasnejad

Abstract

Natural draft dry cooling towers are a type of cooling towers which are used under certain conditions such as insufficient water supplies. But it is found that the cooling efficiency is seriously dependent on the environmental conditions, such as ambient temperature, speed of cross winds, etc.

In present dissertation, firstly the power plants problems under cross wind and high ambient temperature is introduced. Then using forward finite difference for time derivatives and upwind scheme for spatial derivatives, the governing equations in natural convection case (no cross wind) are solved numerically. The governing equations in 3D case (presence of cross wind), are simulated numerically using FLUENT software and the performance losses reasons due to wind speed and ambient temperature are discussed.

Finally, changes in external shape of cooling tower (using wind break walls) are introduced to improve the tower performance and effects of different wind break walls are studied. The results show that using wind break walls at the inlet and outlet of the tower will improve the tower performance under cross wind condition.

Shahrood University of Technology

M.Sc. Dissertation
Mechanical Engineering- Energy Conversion

Title:
A Study of Dry Cooling Towers Performance under Cross Wind and High Ambient Temperature Effects Using Numerical Analysis

Supervisor:
Dr. Mohammad Hassan Kayhani

Consultant Supervisor:
Dr. Mohammad Shahsavan

Student:
Ali Abbasnejad