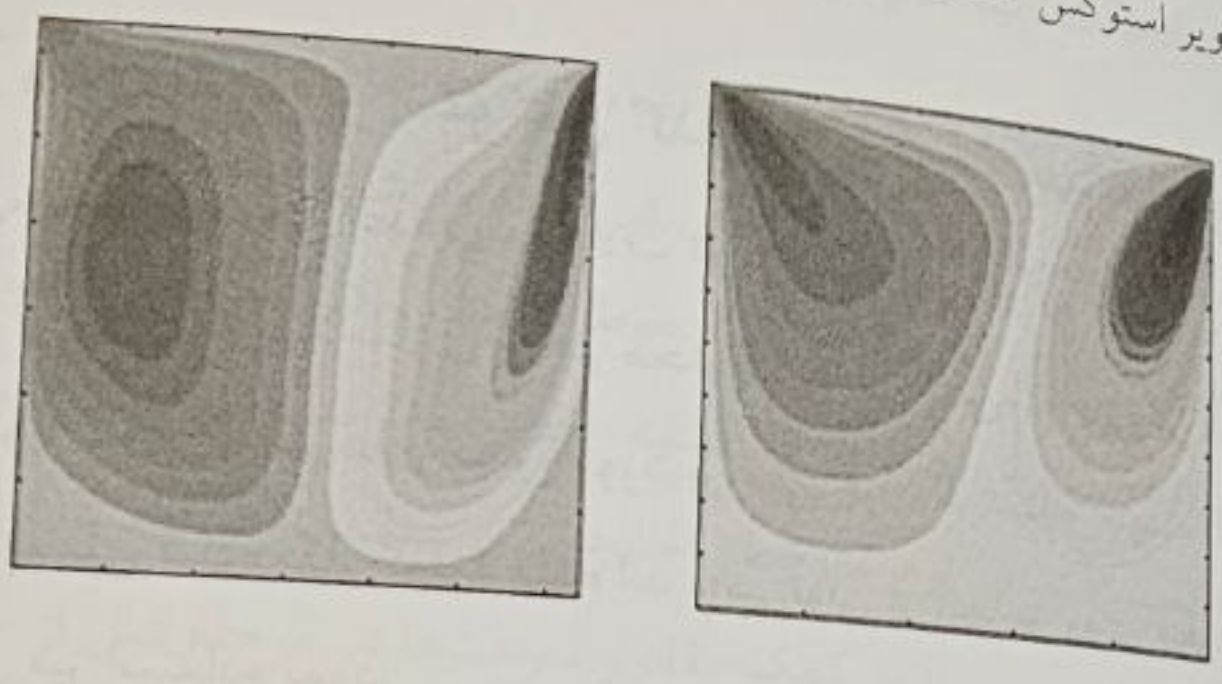


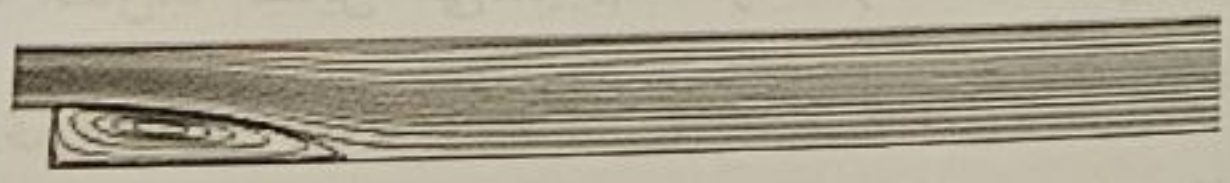
شکل (۱۲-۱۶) توزیع سرعت افقی (بالا) و عمودی (پایین) در رینولدز ۱۰۰ (راست) و ۱۰۰۰ (چپ).

ملاحظه می شود که نتایج بسیار نزدیک به نتایج بدست آمده در روشهای دیگر است.

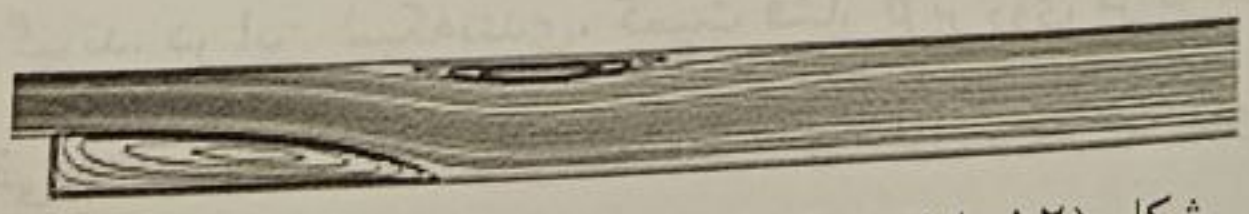
۱۲-۳-۲ مسئلهٔ کاربردی ۴: جریان حول پلهٔعقب گرا

مسئلهٔ جریان سیال لزج حول پلهٔ عقب گرا در بخش روش تفاضل محدود تشریح شد. برای حلّ این مسئله، همانگونه که در حین حلّ مسئلهٔ حفره اشاره شد، می توان با ایجاد تغییراتی در نرم افزار ارائه شده، از آن برای حلّ مسئلهٔ پلهٔ عقب گرا استفاده نمود. در اینجا، به ارائهٔ نتایج حاصل از نرم افزار فوق الذّکر بسنده می شود.

در شکل‌های (۱۲-۱۷) و (۱۲-۱۸)، خطوط جریان حاصله برای رینولدزهای ۱۰۰ و ۱۰۰۰ ارائه شده است که تطابق خوبی با نتایج روشهای دیگر دارد.



شکل (۱۲-۱۷) خطوط جریان در عدد رینولدز ۱۰۰.



شکل (۱۲-۱۸) خطوط جریان در عدد رینولدز ۱۰۰0.

۱۲-۴ حلّ معادلات ناویر استوکس گذرا به روش حجم محدود

در این بخش، به منظور حل عددی معادلات ناویر-استوکس برای جریان گذرای سیال تراکم ناپذیر، از روش گسسته‌سازی حجم محدود استفاده می‌شود. روش‌های گسسته-سازی، همانطور که در بخش‌های پیش مورد بررسی قرار گرفتند، جهت گسسته‌سازی معادلات دیفرانسیلی ساده بکار گرفته شده اند و می‌توان از آنها بر روی معادلات پیوسته ناویر-استوکس نیز استفاده نمود.

در شرایط دوبعدی، دامنهٔ محاسباتی به یک ناحیه مستطیلی شکل محدود خواهد گردید:

$$\Omega := [0, a] \times [0, b] \subset \mathbb{R}^2$$

که بر روی این دامنهٔ مستطیلی شبکه‌بندی مورد نظر جهت گسسته‌سازی معادلات شکل خواهد گرفت. این شبکه به i_{max} سلول در جهت x و $j_{\max}$ سلول در جهت y تقسیم‌بندی خواهد شد. در نتیجه فواصل سلولی بصورت زیر تعریف می‌شوند:

$$\delta x := \frac{1}{i_{\max}} \quad \text{و} \quad \delta y := \frac{b}{j_{max}}$$

همانطور که در فصل ۷ معرفی گردید، در رویکرد حجم محدود از معادلات حاکم بر روی المان‌های حجم کنترل، انتگرال‌گیری نموده و با استفاده از قضیه دیورژانس، در حالت سه‌بعدی انتگرال‌های حجمی را به انتگرال‌های سطحی (سطوح جانبی المان) و در حالت دوبعدی انتگرال‌های سطحی را به انتگرال‌های خطی (خطوط شبکه دوبعدی و تشکیل دهنده المان‌ها تبدیل می‌نماییم. عموماً در گسسته‌سازی معادلات ناویر-استوکس بر روی شبکه‌بندی‌های یکنواخت و منظم از رویکرد شبکه‌های جابجاشده[۷] استفاده می‌گردد. در این رویکرد متغیرهای مجهول مختلف بر روی نقاط مشترکی از شبکه قرار نمی‌گیرند. در این شبکه‌بندی، کمیت فشار p بر روی مراکز سلولی قرار داده شده، سرعت افقی u بر روی نقاط میانی خطوط قائم سلول و سرعت عمودی v بر روی

---

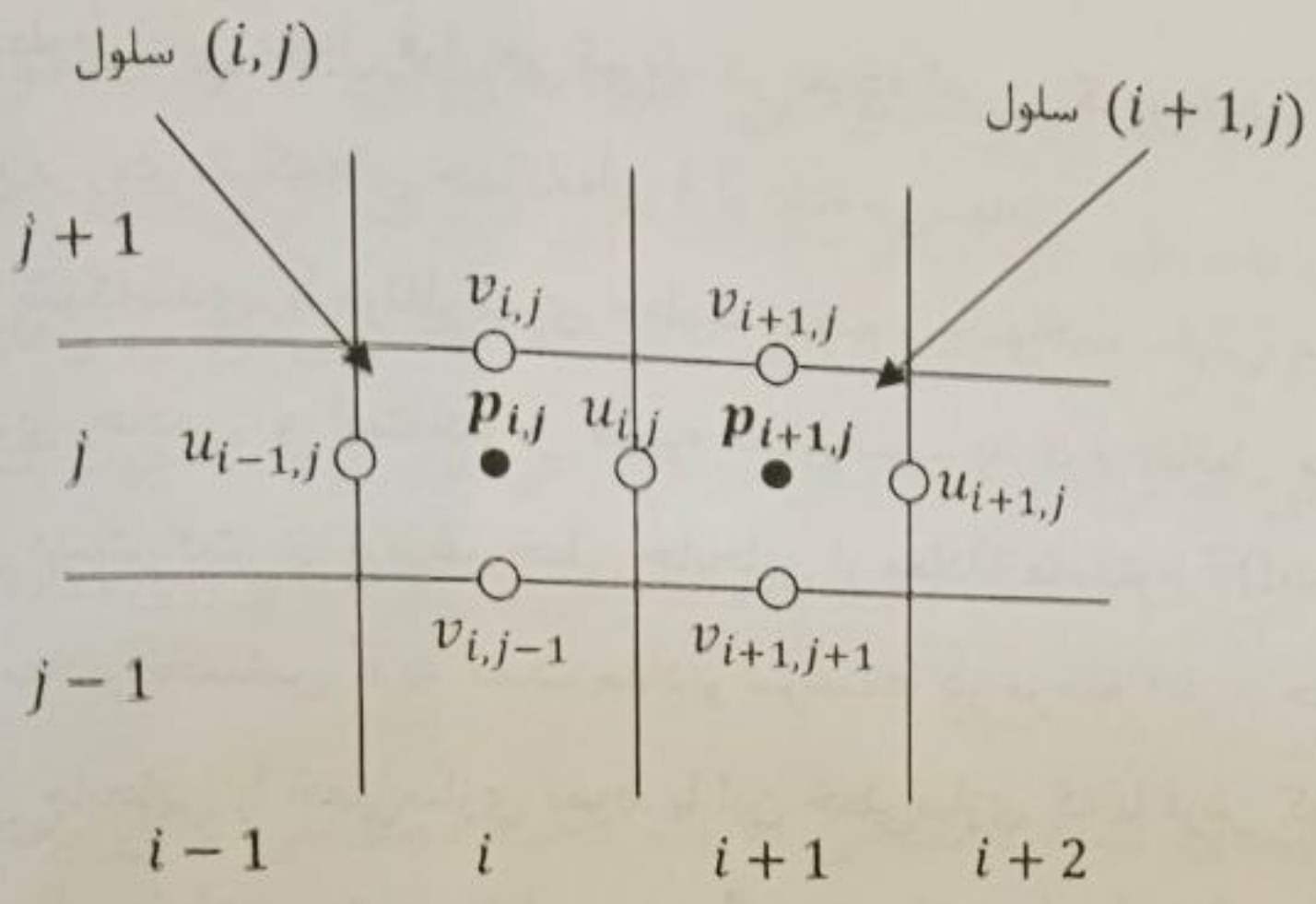
[7] Staggered Grids

نقاط میانی خطوط افقی سلول قرار می‌گیرند. در نتیجه این رویکرد، مقادیر گسسته u، v و p در واقع بر روی شبکه‌های جداگانه‌ای قرار داده می‌شوند.

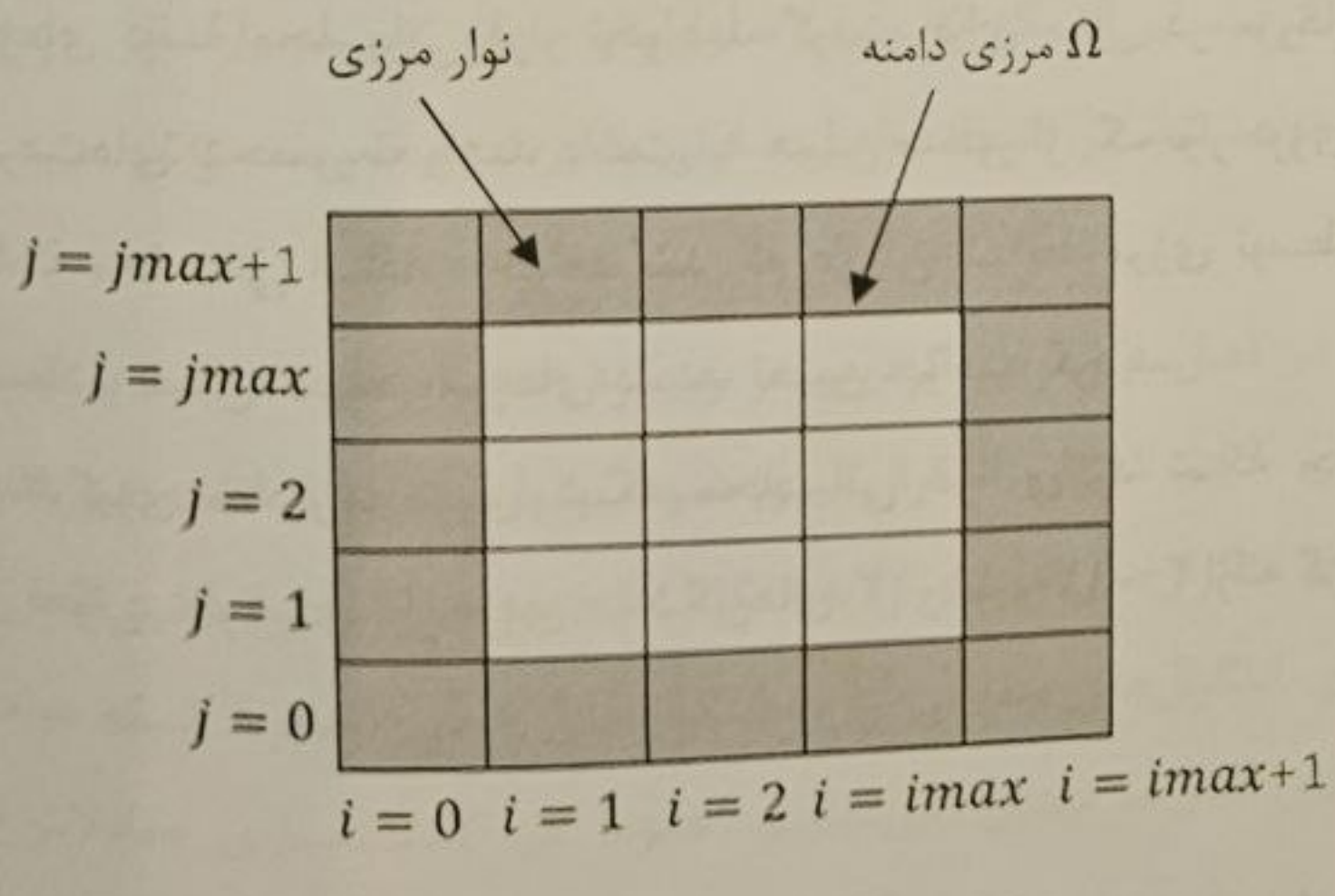
در این نوع از شبکه‌بندی، با درنظرگیری خطوط منظم و یکنواخت سلولی، می‌توان پس از انتگرال‌گیری حجمی و استفاده از قضیه دیورژانس، به فرم تفاضل محدودی از گسسته‌سازی دست‌یافت. در ترم غیرخطی جابجایی از معادلهٔ مومنتوم $(\vec{u} \cdot grad)\vec{u}$، با درنظرگیری مقادیر نخستین $\vec{u}$ به کمک مقادیر سرعت‌ها در مرحلهٔ قبل از حل، می‌توان ترم غیرخطی جابجایی را خطی‌سازی نمود. با این خطی‌سازی که با فرض کوچک بودن بازه زمانی عملی خواهد بود، می‌توان روند گسسته‌سازی را ساده‌سازی نموده و از میانیابی‌های تفاضلی مرتبهٔ اول یا دوّم استفاده نمود.

با درنظرگیری رویکرد شبکهٔ جابجاشده، تمامی نقاط مرزی شبکه به علت جابجایی در داخل مرزهای دامنهٔ محاسباتی قرار نخواهند گرفت. برای مثال در مرزهای عمودی مقادیر سرعت‌های v حضور نخواهند داشت. به همین منظور از یک نوار مرزی اضافی در اطراف دامنهٔ محاسباتی استفاده خواهد شد که در آن شرایط مرزی توسط میانگین-گیری از مقادیر تعیین شده در مرزهای دامنه، تعیین خواهند گردید.

نحوهٔ درنظرگیری مقادیر بر روی شبکه محاسباتی در رویکرد شبکهٔ جابجاشده و همچنین نحوهٔ درنظرگیری نوار مرزی در شکل‌های ۱۲-۱۹ و ۱۲-۲۰ ارائه گردیده‌اند.



شکل (۱۲-۱۹) شبکه‌بندی جابجاشده



شکل (۱۲-۲۰): دامنهٔ محاسباتی با درنظرگیری سلول‌های مرزی

نکتهٔ قابل توجه در استفاده از رویکرد شبکه جابجاشده این است که با درنظرگیری این چیدمان برای مجهولات شبکه، می‌توان از نوسانات احتمالی میدان فشار یا بعبارت دیگر اثر شطرنجی فشار در دامنه محاسباتی جلوگیری نمود.

بعنوان یک روش جایگزین در مقابل روش چیدمان جابجاشده، روش چیدمان هم‌مکان[۸] پیشنهاد می‌گردد که همانطور که از اسم آن برمی‌آید، تمامی مقادیر u، v و p در مراکز سلول‌ها درنظر گرفته شده و در این شرایط، یک گسسته‌سازی حجم محدود کامل با بکارگیری روش‌های میانیابی ویژه‌ای به منظور محاسبهٔ شار عبوری از سطوح جانبی حجم کنترل، بکارگرفته خواهد شد.

با توجه به آنچه مطرح گردید، به منظور جلوگیری از بروز نوسانات غیرفیزیکی فشار در داخل دامنه محاسباتی، در این مسئله از رویکرد چیدمان جابجاشده استفاده گردیده و نحوهٔ گسسته‌سازی بر روی این نوع از شبکه با استفاده از رویکرد حجم محدود مبتنی بر تفاضلات محدود، بصورتی که در ادامه ارائه می‌گردد، معرفی خواهد شد.

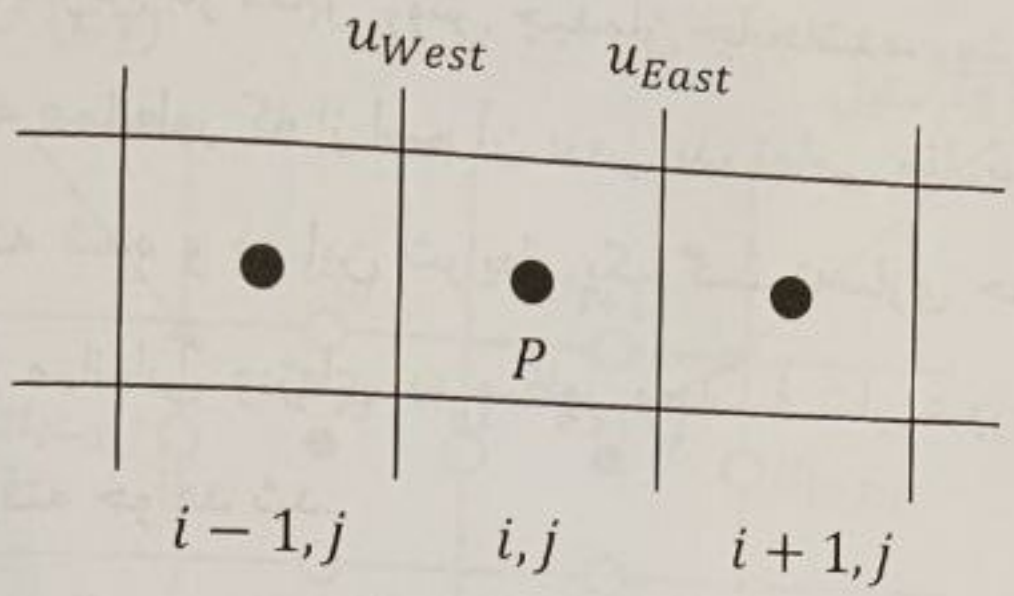
معادلهٔ پیوستگی با جایگذاری مشتقّات مکانی $\frac{\partial u}{\partial x}$ و $\frac{\partial v}{\partial y}$ به کمک تفاضلات مرتبهٔ اوّل و بر روی مراکز سلول‌های محاسباتی گسسته‌سازی خواهد گردید. لازم به ذکر است که همانطور که در ادامه نشان داده می‌شود، با انتگرال‌گیری روی سطح از عبارت مشتق، استفاده از قضیه دیورژانس، بکارگیری رویکرد چیدمان جابجاشده، میانیابی مرتبهٔ اوّل بالادستی و نهایتاً تقسیم نمودن تمامی ترم‌ها بر حجم سلول‌ها (یا بر سطح سلول‌ها در حالت دوّم)، گسسته‌سازی حجم محدود به فرم تفاضل محدود، در خواهد آمد.

$$\int_{C.V.} \frac{\partial u}{\partial x}\, dV \overset{2D}{\rightarrow} \int_{C.F.} \frac{\partial u}{\partial x}\, dA =$$

$$\oint_{cell\ edges} \frac{u_{face}}{dx}\, dx\, dy = \oint_{cell\ edges} u_{edge}\, dy = (u_{East} - u_{West})\, dy$$

بکارگیری میانیابی بالادستی جهت تقریب نمودن مقادیر روی مرزهای شرقی و غربی، با فرض جریان سیال از سمت غرب به شرق نتیجه می‌دهد:

$$u_{West} = u_{i-1,j} \quad \text{و} \quad u_{East} = u_{i,j}$$



شکل (۱۲-۲۱): نحوهٔ قرارگیری سرعتهای افقی و فشار در شبکهٔ محاسباتی

با توجه به این موضوع، می‌توان نشان داد که تقریب حجم محدود عبارت مشتق مرتبهٔ اوّل در چارچوب چیدمان جابجاشده، با بکارگیری میانیابی مرتبهٔ اوّل بالادستی، به فرم تقریب تفاضل محدود با تقریب تفاضلی مرتبهٔ اوّل پسرو درخواهد آمد.

$$\int_{C.F.} \frac{\partial u}{\partial x}\, dA = \left(u_{i,j} - u_{i-1,j}\right)dy \xrightarrow{\text{تقسیم بر حجم سلول}} = \frac{u_{i,j}-u_{i-1,j}}{dx}$$

مشابه همین نتیجه، برای مشتق مرتبهٔ اوّل در جهت y (سلول‌های جابجاشده در راستای قائم) حاصل خواهد گردید:

$$\frac{\partial v}{\partial y}|_{i,j} := \frac{v_{i,j}-v_{i,j-1}}{\delta y}$$

مشتقات مرتبه دوم نیز بصورت کاملاً مشابه محاسبه گردیده و گسسته‌سازی ترم انتشار[9] را نتیجه خواهند داد. نهایتاً جهت گسسته‌سازی ترم‌های جابجایی، انتشار و گرادیان فشار در معادلات مومنتوم در جهات x و y، بصورت زیر خواهیم داشت:

- معادلهٔ مومنتوم در جهت x

$$\left[\frac{\partial(u^2)}{\partial x}\right]_{i,j} :=$$
$$\frac{1}{\delta x}\left(\left(\frac{u_{i,j}+u_{i+1,j}}{2}\right)^2 - \left(\frac{u_{i-1,j}+u_{i,j}}{2}\right)^2\right) + \gamma\frac{1}{\delta x}\left(\frac{|u_{i,j}+u_{i+1,j}|}{2}\frac{(u_{i,j}-u_{i+1,j})}{2} - \frac{|u_{i-1,j}+u_{i,j}|}{2}\frac{(u_{i-1,j}-u_{i,j})}{2}\right)$$

[9] Diffusion Term

$$\left[\frac{\partial(uv)}{\partial y}\right]_{i,j} := \frac{1}{\delta y}\left(\frac{(v_{i,j}+v_{i+1,j})}{2}\frac{(u_{i,j}+u_{i,j+1})}{2} - \frac{(v_{i,j-1}+v_{i+1,j-1})}{2}\frac{(u_{i,j-1}+u_{i,j})}{2}\right) + \gamma\frac{1}{\delta y}\left(\frac{|v_{i,j}+v_{i+1,j}|}{2}\frac{(u_{i,j}-u_{i,j+1})}{2} - \frac{|v_{i,j-1}+v_{i+1,j-1}|}{2}\frac{(u_{i,j-1}-u_{i,j})}{2}\right)$$

$$\left[\frac{\partial^2 u}{\partial x^2}\right]_{i,j} := \frac{u_{i+1,j}-2u_{i,j}+u_{i-1,j}}{(\delta x)^2}$$

$$\left[\frac{\partial^2 u}{\partial y^2}\right]_{i,j} := \frac{u_{i,j+1}-2u_{i,j}+u_{i,j-1}}{(\delta y)^2}$$

$$\left[\frac{\partial p}{\partial x}\right]_{i,j} := \frac{p_{i+1,j}-p_{i,j}}{\delta x}$$

- معادلهٔ مومنتوم در جهت y

$$\left[\frac{\partial(uv)}{\partial x}\right]_{i,j} := \frac{1}{\delta x}\left(\frac{(u_{i,j}+u_{i,j+1})}{2}\frac{(v_{i,j}+v_{i+1,j})}{2} - \frac{(u_{i-1,j}+u_{i-1,j+1})}{2}\frac{(v_{i-1,j}+v_{i,j})}{2}\right) + \gamma\frac{1}{\delta x}\left(\frac{|u_{i,j}+u_{i,j+1}|}{2}\frac{(v_{i,j}-v_{i+1,j})}{2} - \frac{|u_{i-1,j}+u_{i-1,j+1}|}{2}\frac{(v_{i-1,j}-v_{i,j})}{2}\right)$$

$$\left[\frac{\partial(v^2)}{\partial x}\right]_{i,j} := \frac{1}{\delta y}\left(\left(\frac{v_{i,j}+v_{i,j+1}}{2}\right)^2 - \left(\frac{v_{i,j-1}+v_{i,j}}{2}\right)^2\right) + \gamma\frac{1}{\delta y}\left(\frac{|v_{i,j}+v_{i,j+1}|}{2}\frac{(v_{i,j}-v_{i,j+1})}{2} - \frac{|v_{i,j-1}+v_{i,j}|}{2}\frac{(v_{i,j-1}-v_{i,j})}{2}\right)$$

$$\left[\frac{\partial^2 v}{\partial x^2}\right]_{i,j} := \frac{v_{i+1,j}-2v_{i,j}+v_{i-1,j}}{(\delta x)^2}$$

$$\left[\frac{\partial^2 v}{\partial y^2}\right]_{i,j} := \frac{v_{i,j+1}-2v_{i,j}+v_{i,j-1}}{(\delta y)^2}$$

$$\left[\frac{\partial p}{\partial y}\right]_{i,j} := \frac{p_{i,j+1}-p_{i,j}}{\delta y}$$

پارامتر γ در روابط بالا مقداری بین ۰ و ۱ را اختیار نموده بطوری که برای $\gamma = 0$ به گسسته‌سازی تفاضل مرکزی و برای $\gamma = 1$ یک گسسته‌سازی کاملاً مرتبهٔ اوّل با استفاده از مقادیر سلول‌های بالادستی خواهیم داشت. با توجه به مطالعات انجام شده در زمینهٔ گسسته‌سازی حجم محدود، مشخص گردیده که مقدار پارامتر γ بایستی رابطهٔ زیر را ارضاء نماید:

$$\gamma \geq \max\left(\left|\frac{u_{i,j}\delta t}{\delta x}\right|, \left|\frac{v_{i,j}\delta t}{\delta y}\right|\right)$$

به منظور گسسته‌سازی ترم زمانی معادلهٔ ناویر-استوکس، با تقسیم زمان کلی حل $[0, t_{end}]$ به بازه‌های زمانی δt، مقادیر u، v و p در زمان‌های $n\delta t$ محاسبه می‌گردند. برای گسسته‌سازی ترم زمانی به روش حجم محدود، مشابه آنچه در بخش‌های قبلی مطرح گردید، از انتگرال‌گیری عبارت دیفرانسیلی بر روی حجم کنترل استفاده می‌نماییم. با توجه به اینکه مقادیر سرعت‌ها در این ترم، تنها وابسته به زمان می‌باشند، در نتیجه از انتگرال خارج شده و با استفاده از روش اویلر و با بکارگیری مقادیر تفاضلی مرتبهٔ اوّل، به عبارت جبری گسسته‌سازی شده تبدیل می‌گردند.

$$\int_{C.V.} \frac{\partial u}{\partial t}\, dV = \frac{du}{dt}\, V$$

نهایتاً، با تقسیم‌نمودن تمامی ترم‌ها بر حجم سلول محاسباتی، خواهیم داشت:

$$\left[\frac{du}{dt}\right]^{(n+1)} := \frac{u^{(n+1)} - u^{(n)}}{\delta t} \quad , \quad \left[\frac{dv}{dt}\right]^{(n+1)} := \frac{v^{(n+1)} - v^{(n)}}{\delta t}$$

که کاملاً مشابه رویکرد تفاضل محدود پیشرو زمانی می‌باشد.

نکتهٔ قابل توجه در این بخش این است که، چنانچه تمامی ترم‌های باقیمانده در عبارت‌های دیفرانسیلی و به خصوص مشتقات مکانی موجود در معادلهٔ ناویر-استوکس، مقادیر زمان فعلی (t_n) را اختیار نمایند، ما اصطلاحاً در حال استفاده از رویکرد صریح[10] در حلّ معادلات می‌باشیم که در آن مقادیر حل در زمان t_{n+1} مستقیماً به کمک مقادیر زمان t_n محاسبه می‌گردند. در مقابل رویکرد ضمنی[11] وجود داشته که در آن تمامی مقادیر مشتقات مکانی در زمان بعدی (t_{n+1}) محاسبه می‌گردند. استفاده از رویکرد ضمنی با حفظ پایداری حل، سبب ایجاد امکان استفاده از بازه‌های زمانی بزرگتری برای حل مسئله خواهد گردید. امّا در صورت بکارگیری هر یک از این روش‌ها، در هر بازهٔ زمانی نیازمند حلّ یک سیستم خطی و یا حتّی غیرخطی معادلات جبری خواهیم بود.

۱-۴-۱۲ نحوهٔ اعمال شرایط مرزی

[10] Explicit Approach
[11] Implicit Approach

پس از بررسی نحوهٔ گسسته‌سازی معادلهٔ ناویر-استوکس برای سلول‌های درون دامنهٔ محاسباتی، بایستی به نحوهٔ اعمال شرایط مرزی در نواحی اطراف دامنهٔ محاسباتی پرداخت. در حالت کلی، نیازمند مقادیر زیر بر روی مرزهای دامنهٔ محاسباتی خواهیم بود:

$$u_{0,j} \quad , \quad u_{i_{\max},j} \qquad j = 1, \dots, j_{\max}$$
$$v_{i,0} \quad , \quad v_{i,j_{\max}} \qquad i = 1, \dots, i_{\max}$$

همچنین جهت اعمال شرایط مرزی، نیازمند محاسبهٔ مقادیر زیر در خارج از دامنهٔ محاسباتی خواهیم بود:

$$u_{i,0} \quad , \quad u_{i,j_{\max}+1} \qquad i = 1, \dots, i_{\max}$$
$$v_{0,j} \quad , \quad v_{i_{\max}+1,j} \qquad j = 1, \dots, j_{\max}$$

این مقادیر سرعت‌ها از گسسته‌سازی شرایط مرزی در مسئله پیوستهٔ مورد بررسی بدست خواهند آمد. در ادامه به بررسی چند شرط مرزی متفاوت و پرکاربرد در حل مسائل دینامیک سیالات محاسباتی پرداخته و نحوه اعمال آنها در معادلات گسسته-سازی شده تشریح می شود.

الف) شرط مرزی دیواره بدون لغزش[۱۲]

سرعت‌های پیوسته بایستی به منظور ارضاء شرط عدم لغزش در مرزها صفر شوند. در نتیجه برای مقادیر سرعت‌هایی که کاملاً بر روی مرزها قرار می‌گیرند، خواهیم داشت:

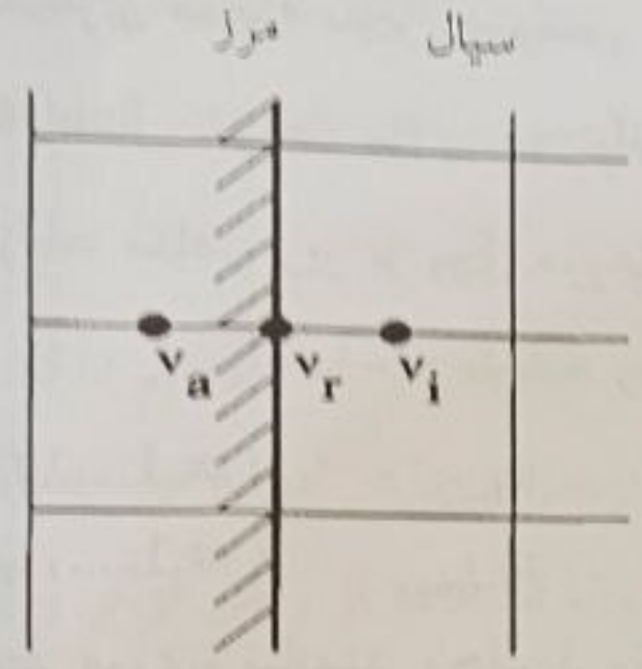
$$u_{0,j} = 0 \quad , \quad u_{i_{\max},j} = 0 \qquad j = 1, \dots, j_{\max}$$
$$v_{i,0} = 0 \quad , \quad v_{i,j_{\max}} = 0 \qquad i = 1, \dots, i_{\max}$$

با توجه به اینکه در مرزهای قائم، مقادیر سرعت‌های v و در مرزهای افقی، مقادیر سرعت‌های u حضور نخواهند داشت، مقدار مرزی صفر برای این متغیرها با میانگین-گیری از مقادیر دو طرف مرز، اعمال خواهد گردید:

$$v_r := \frac{v_a + v_i}{2} = 0 \quad \Rightarrow \quad v_a = -v_i$$

---

[12] No-Slip Boundary Condition



شکل (۱۲-۲۲): محلّ گرها در سیال و مرز

در نتیجه، بر روی چهار مرز براساس شرایط مرزی، خواهیم داشت:

$$u_{i,0} = -u_{i,1} \quad , \quad u_{i,j_{\max}+1} = -u_{i,j_{\max}} \qquad i = 1, \ldots, i_{\max}$$
$$v_{0,j} = -v_{1,j} \quad , \quad v_{i_{\max}+1,j} = -v_{i_{\max},j} \qquad j = 1, \ldots, j_{\max}$$

البته، بایستی به این نکته توجه داشت که در مرزهای دیوارهٔ در حال حرکت، مقادیر سرعت غیر صفر و معادل سرعت حرکت دیواره در نظر گرفته خواهد شد. این شرایط برای مثال در مسئلهٔ جریان داخل حفره[۱۳] که در ادامه به حل و بررسی آن پرداخته می‌می‌شود، بکار گرفته شده است.

ب) شرط مرزی دیواره با لغزش آزاد[۱۴]

در مرز با دیوارهٔ لغزنده، مؤلفهٔ عمود بر سطح مرزی سرعت و همچنین مؤلفهٔ مماس بر سطح مرزی گرادیان سرعت بایستی برابر صفر درنظر گرفته شوند. در این شبکه مربعی حاضر که در آن از رویکرد چیدمان جابجاشده استفاده گردیده است، مقادیر سرعت‌های عمود به مرز مستقیماً بر روی مرز قرار گرفته و در نتیجه مشابه آنچه برای شرط مرزی بدون لغزش در نظر گرفته بودیم، خواهیم داشت:

$$u_{0,j} = 0 \quad , \quad u_{i_{\max},j} = 0 \qquad j = 1, \ldots, j_{\max}$$
$$v_{i,0} = 0 \quad , \quad v_{i,j_{\max}} = 0 \qquad i = 1, \ldots, i_{\max}$$

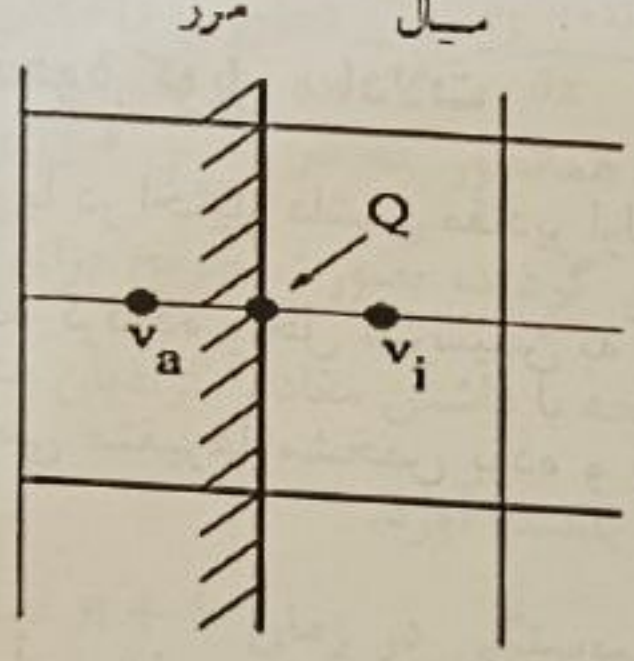
[13] Driven Cavity Problem
[14] Free-Slip Boundary Condition

همچنین گرادیان عمود بر سطح سرعت مماس بر دیواره در نقطهٔ مرزی Q، به کمک رابطهٔ $\frac{v_i - v_a}{\delta x}$ گسسته‌سازی گردیده و با اعمال در رابطهٔ $\frac{\partial v}{\partial n} = 0$ ، عبارت زیر بدست می آید:

$$v_a = v_i$$

به همین ترتیب، شرایط مرزی دیگر نیز بصورت زیر بدست می آید:

$$u_{i,0} = u_{i,1} \quad , \quad u_{i,j_{\max}+1} = u_{i,j_{\max}} \qquad i = 1, \dots, i_{\max}$$
$$v_{0,j} = v_{1,j} \quad , \quad v_{i_{\max}+1,j} = v_{i_{\max},j} \qquad j = 1, \dots, j_{\max}$$



شکل(۱۲-۲۳) شرط مرزی لغزش آزاد در مرز دامنهٔ محاسباتی

ج) شرط مرزی جریان خروجی

در مرز خروجی جریان، گرادیان‌های نرمال هر دو مؤلفهٔ سرعت بر روی مرز برابر صفر قرار داده می‌شوند. این شرایط بدین معنی است که سرعت‌های کلّی در جهت عمود بر مرز خروجی تغییر نخواهند کرد. در چارچوب معادلات گسسته‌سازی شده، این شرایط با مساوی قرار دادن مقادیر سرعت در مرز با مقادیر سرعت در سلول‌های همسایه در درون دامنه محاسباتی، اعمال خواهد گردید.

$$u_{0,j} = u_{1,j} \quad , \quad u_{i_{\max},j} = u_{i_{\max-1},j}$$
$$v_{0,j} = v_{1,j} \quad , \quad v_{i_{\max}+1,j} = v_{i_{\max},j} \qquad j = 1, \dots, j_{\max}$$
$$u_{i,0} = u_{i,1} \quad , \quad u_{i,j_{\max}+1} = u_{i,j_{\max}}$$
$$v_{i,0} = v_{i,1} \quad , \quad v_{i,j_{\max}} = v_{i,j_{\max}-1} \qquad i = 1, \dots, i_{\max}$$

د) شرط مرزی ورودی جریان

در مرزی که جریان سیال از آن به داخل دامنه محاسباتی راه می‌یابد، سرعت‌های بصورت صریح اعمال می‌گردند. در نتیجه مقادیر سرعت عمود بر مرز (برای مثال مقدار سرعت u بر روی مرز قائم در سمت چپ)، با ثابت نمودن مستقیم مقادیر سرعت بر روی خط مرزی تعیین می‌گردند. برای مقادیر سرعت مماس بر مرز (برای مثال مقدار سرعت v بر روی مرز قائم در سمت چپ)، از میانگین‌گیری سرعت‌ها در طرفین خط مرزی استفاده خواهد شد.

۱۲-۴-۲ الگوریتم حل و نحوهٔ کوپل معادلات

با شروع حل از زمان $t = 0$ و با در اختیار داشتن مقادیر اولیهٔ سرعت‌های u و v، در هر بازهٔ زمانی، به میزان δt افزوده گردیده و حل تا رسیدن به زمان نهایی t_{end} ادامه می‌یابد. در زمان n، مقادیر تمامی متغیرها مشخص بوده و بوسیلهٔ آنها مقادیر در زمان t_{n+1} محاسبه می‌گردند.

به منظور بررسی نحوهٔ حلّ توأم میدان سرعت و فشار، ابتدا معادلات مومنتوم در جهت x و y بین زمان n و $n+1$ گسسته‌سازی می‌شود.

$$\begin{cases} u^{(n+1)} = u^{(n)} + \delta t \left[\frac{1}{Re}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) - \frac{\partial(u^2)}{\partial x} - \frac{\partial(uv)}{\partial y} + g_x - \frac{\partial p}{\partial x}\right] \\ v^{(n+1)} = v^{(n)} + \delta t \left[\frac{1}{Re}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) - \frac{\partial(uv)}{\partial x} - \frac{\partial(v^2)}{\partial y} + g_y - \frac{\partial p}{\partial y}\right] \end{cases}$$

در این شرایط، ابتدا با حذف عبارت گرادیان فشار از هر دو معادلهٔ مومنتوم، مقادیر F و G بصورت زیر معرفی می‌شوند.

$$\begin{cases} F := u^{(n)} + \delta t \left[\frac{1}{Re}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) - \frac{\partial(u^2)}{\partial x} - \frac{\partial(uv)}{\partial y} + g_x\right] \\ G := v^{(n)} + \delta t \left[\frac{1}{Re}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) - \frac{\partial(uv)}{\partial x} - \frac{\partial(v^2)}{\partial y} + g_y\right] \end{cases}$$

و در نتیجه خواهیم داشت:

$$u^{(n+1)} = F - \delta t \frac{\partial p}{\partial x}$$
$$v^{(n+1)} = G - \delta t \frac{\partial p}{\partial x}$$

به منظور تکمیل فرآیند گسسته‌سازی معادلهٔ مومنتوم در زمان، بایستی به هر یک از ترم‌های سمت راست معادلهٔ بالا، زمانی را اطلاق نمود. در نتیجه با توجه به رابطهٔ معرفی شده برای F و G، فرض می‌شود که این دو عبارت در زمان n محاسبه گردیده و مقادیر گرادیان فشار نیز هر دو در زمان $n+1$. در نتیجه می‌توان به عبارت گسسته‌سازی شدهٔ زیر برای معادلهٔ مومنتوم دست یافت:

$$u^{(n+1)} = F^{(n)} - \delta t \frac{\partial p^{(n+1)}}{\partial x}$$

$$v^{(n+1)} = G^{(n)} - \delta t \frac{\partial p^{(n+1)}}{\partial x}$$

این رویکرد بکارگرفته شده، همانطور که در بخش قبل و در گسسته‌سازی مکانی و زمانی ترم‌های معادله معرفی گردید، بصورت صریح برای سرعت‌ها و بصورت ضمنی برای فشار خواهد بود. در نتیجه با داشتن مقادیر گرادیان فشار در زمان $n+1$، می‌توان مقادیر سرعت را در این زمان بدست آورد.

برای این منظور، از رابطهٔ پیوستگی در زمان $n+1$ استفاده می‌شود. با جایگذاری عبارت اخیر در معادلهٔ پیوستگی خواهیم داشت:

معادله پیوستگی

$$0 = \frac{\partial u^{(n+1)}}{\partial x} + \frac{\partial v^{(n+1)}}{\partial y} = \frac{\partial F^{(n)}}{\partial x} - \delta t \frac{\partial^2 p^{(n+1)}}{\partial x^2} + \frac{\partial G^{(n)}}{\partial y} - \delta t \frac{\partial^2 p^{(n+1)}}{\partial y^2}$$

عبارت بالا پس از مرتب‌سازی، معادله پواسون فشار را برای اسکالر $p^{(n+1)}$ در زمان t_{n+1} بصورت زیر بدست خواهد داد:

$$\frac{\partial^2 p^{(n+1)}}{\partial x^2} + \frac{\partial^2 p^{(n+1)}}{\partial y^2} = \frac{1}{\delta t}\left(\frac{\partial F^{(n)}}{\partial x} + \frac{\partial G^{(n)}}{\partial y}\right)$$

استفاده از این دو معادله در کنار یکدیگر سبب تضمین میدان سرعت بدون دیورژانس[15] خواهد گردید. الگوریتم کلی حل و پیشبرد زمانی میدان حلّ توأمِ سرعت و فشار بصورت زیر جمع‌بندی می‌گردد:

- **گام اول:** محاسبهٔ $F^{(n)}$ و $G^{(n)}$ با استفاده از مقادیر $u^{(n)}$ و $v^{(n)}$
- **گام دوم:** حلّ معادلهٔ پواسون برای محاسبهٔ فشار $p^{(n+1)}$

[15] Divergence-Free Velocity Field

- **گام سوم:** محاسبهٔ میدان سرعت جدید $\left(u^{(n+1)}, v^{(n+1)}\right)^T$ با استفاده از مقادیر جدید فشار $p^{(n+1)}$ محاسبه شده در گام دوّم. حلّ معادلهٔ پواسون در گام دوّم، نیازمند محاسبهٔ مقادیر مرزی گرادیان فشار خواهد بود که بصورت زیر معرفی می‌گردد:

$$grad\, p^{(n+1)} \cdot \vec{n} = \frac{\partial p^{(n+1)}}{\partial x} n_1 + \frac{\partial p^{(n+1)}}{\partial y} n_2 = -\frac{1}{\delta t}\left(\left(u^{(n+1)} - F^{(n)}\right)n_1 + \left(v^{(n+1)} - G^{(n)}\right)n_2\right)$$

به روش معرفی شدهٔ حاضر برای حل توأم میدان سرعت و فشار، روش پروژکشن کورین[16] گفته می‌شود.

● برنامه کامپیوتری ۱۲-۵

به منظور حل عددی معادلهٔ ناویر-استوکس، با توجه به موارد مطرح شده، یک برنامهٔ محاسباتی به کمک زبان برنامه‌نویسی فرترن ۹۰ نوشته شده که در زیر ارائه می‌گردد. لازم به ذکر است که به منظور حلّ دو مسئلهٔ متفاوت جریان داخل حفره و جریان بر روی پلهٔ عقبگرد[17] در این فصل و همچنین حلّ مسئلهٔ جریان سیّال حول سیلندر دایروی در فصل ۱۳، برنامهٔ کامپیوتری بصورت عمومی نوشته شده است؛ بدین معنی که مشخصات دامنهٔ محاسباتی، شرایط حل و تنظیمات مسئله برای هر موضوع بطور جداگانه در یک فایل پارامتری ذخیره گردیده و برای حل مسئله مورد نظر تنها لازم است که فایل پارامترهای مسئله به بخش تنظیمات برنامه وارد شود.

در این بخش برنامه کامپیوتری عمومی حلگر ناویر-استوکس ارائه گردیده و در ادامه این بخش، تنظمیات، مشخصات دامنهٔ محاسباتی و شرایط مرزی برای دو مسئلهٔ متفاوت جریان داخل حفره و جریان بر روی پلّهٔ عقبگرد، ارائه خواهند شد.

بایستی به این نکته توجه داشت که کد ناویر-استوکس حاضر بصورت کاملاً عمومی نگارش یافته و برای حلّ مسائل مختلف تنها بخش اولیه کد بایستی تغییر نماید. در

---

[16] Chorin Projection Method
[17] Flow Over Backward Step

اینجا کد عمومی ارائه می‌گردد و در بخش‌های پیش رو و در حلّ مسائل مختلف، بخش اختصاصی هر مسئله ارائه می‌گردد.

```
PROGRAM PGRG2DNAVISTO
IMPLICIT NONE

  ! DEFINITIONS HEADER USED FOR FLAGS
  INTEGER , PARAMETER :: C_B = 0
  INTEGER , PARAMETER :: B_N = 1
  INTEGER , PARAMETER :: B_S = 2
  INTEGER , PARAMETER :: B_W = 4
  INTEGER , PARAMETER :: B_E = 8
  INTEGER , PARAMETER :: B_NW = B_N + B_W
  INTEGER , PARAMETER :: B_SW = B_S + B_W
  INTEGER , PARAMETER :: B_NE = B_N + B_E
  INTEGER , PARAMETER :: B_SE = B_S + B_E

  INTEGER , PARAMETER :: C_F = 16
  INTEGER , PARAMETER :: C_X = C_F - 1
  INTEGER , PARAMETER :: C_A = C_F + B_N + B_S + B_E + B_W
  !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

  INTEGER , PARAMETER :: C_O = 256
  INTEGER , PARAMETER :: C_W = 512
  INTEGER , PARAMETER :: C_S = 1024
  INTEGER , PARAMETER :: C_N = 2048
  INTEGER , PARAMETER :: C_E = 4096

  INTEGER , PARAMETER :: C_WO = C_W + C_O
  INTEGER , PARAMETER :: C_NS = C_N + C_S
  INTEGER , PARAMETER :: C_SW = C_S + C_W
  INTEGER , PARAMETER :: C_NW = C_N + C_W
  INTEGER , PARAMETER :: C_NO = C_N + C_O
  INTEGER , PARAMETER :: C_SO = C_S + C_O

  INTEGER , PARAMETER :: C_SWO = C_S + C_W + C_O
  INTEGER , PARAMETER :: C_NSW = C_N + C_S + C_W
  INTEGER , PARAMETER :: C_NWO = C_N + C_W + C_O
  INTEGER , PARAMETER :: C_NSO = C_N + C_S + C_O

  INTEGER , PARAMETER :: C_NSWO = C_N + C_S + C_W + C_O
  !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

  ! FOR FLAG INITIALIZATION
  INTEGER IHI,ILO,LOW,UP
  REAL :: MX,MY,RAD1,X,Y

  ! FOR COMPUTING F & G
  REAL :: DU2DX,DUVDX,DUVDY,DV2DY,LAPLU,LAPLV

  ! FOR POISSON EQUATION
```

```
  REAL :: ADD,BETA_2,BETA_MOD,P0,RDX2,RDY2
  INTEGER EPS_E,EPS_N,EPS_S,EPS_W,ITER

  ! FOR MAIN
  REAL :: DELT,DELX,DELY,EPS,GAMMA,GX,GY,OMEGA,RE,RES,T,T_END,UI,VI
  INTEGER :: I,J,IBOUND,IFULL,IMAX,ITERMAX,ITERSOR,IWRITE,JMAX
  INTEGER :: OUTCOUNT,PRINTLABEL,WE,WN,WS,WW,CYCLES
  REAL :: XLENGTH,YLENGTH
  REAL , DIMENSION(:,:), ALLOCATABLE ::
U,V,P,RHS,UOUT,VOUT,POUT,F,G,PSI,ZETA
  INTEGER , DIMENSION(:,:), ALLOCATABLE :: FLAG
  CHARACTER ( LEN = 30 ) :: PROBLEM
  CHARACTER ( LEN = 2 ) :: OUTNUM

!****************************!
!****************************!
!***** PROBLEM SETUP *****!
!****************************!
!****************************!
UI = 0.
VI = 0.
!****************************!
!
!ALLOCATE ARRAYS.
!
  ALLOCATE ( F(0:IMAX+1,0:JMAX+1) )
  ALLOCATE ( FLAG(0:IMAX+1,0:JMAX+1) )
  ALLOCATE ( G(0:IMAX+1,0:JMAX+1) )
  ALLOCATE ( P(0:IMAX+1,0:JMAX+1) )
  ALLOCATE ( PSI(0:IMAX,0:JMAX) )
  ALLOCATE ( RHS(0:IMAX+1,0:JMAX+1) )
  ALLOCATE ( U(0:IMAX+1,0:JMAX+1) )
  ALLOCATE ( V(0:IMAX+1,0:JMAX+1) )
  ALLOCATE ( UOUT(0:IMAX,0:JMAX) )
  ALLOCATE ( VOUT(0:IMAX,0:JMAX) )
  ALLOCATE ( POUT(0:IMAX,0:JMAX) )
  ALLOCATE ( ZETA(0:IMAX,0:JMAX) )
!****************************!
!
!  SETTING INITIAL VALUES
!
  U(0:IMAX+1,0:JMAX+1) = UI
  V(0:IMAX+1,0:JMAX+1) = VI
  P(0:IMAX+1,0:JMAX+1) = 0.0

  IF ( PROBLEM == "BACKWARDSTEP" ) THEN
    U(0:IMAX+1,0:JMAX/2) = 0.0
  END IF
!****************************!
!  INITIALIZE FLAGS.
!
!  INITIALIZE THE BOUNDARY CELLS.

  DO I = 0, IMAX+1
    FLAG(I,0) = C_B
```

```
      FLAG(I,JMAX+1) = C_B
    END DO

    DO J = 1, JMAX
      FLAG(0,J) = C_B
      FLAG(IMAX+1,J) = C_B
    END DO
!
!   INITIALIZE THE FLUID CELLS.
!
    DO I = 1, IMAX
      DO J = 1, JMAX
        FLAG(I,J) = C_F
      END DO
    END DO
!
!   SPECIALIZE TO THE PROBLEM
!
    IF ( PROBLEM == "BACKWARDSTEP" ) THEN

      DO I = 1, JMAX
        DO J = 1, JMAX/2
          FLAG(I,J) = C_B
        END DO
      END DO

      WRITE ( *, '(A)' ) '  PROBLEM = ', PROBLEM

    ELSE IF ( PROBLEM == "CYLINDER" ) THEN

      MX = 20.0 / 41.0 * JMAX * DELY
      MY = MX
      RAD1 = 5.0 / 41.0 * JMAX * DELY

      DO I = 1, IMAX
        DO J = 1, JMAX
          X = (I-0.5) * DELX
          Y = (J-0.5) * DELY
          IF ((X-MX)*(X-MX)+(Y-MY)*(Y-MY) <= RAD1 * RAD1 ) THEN
            FLAG(I,J) = C_B
          END IF
        END DO
      END DO

      WRITE ( *, '(A)' ) '  PROBLEM = ', PROBLEM

    ELSE IF ( PROBLEM == "CAVITY" ) THEN

      WRITE ( *, '(A)' ) '  PROBLEM = ', PROBLEM

    END IF
!
!   FLAGS FOR BOUNDARY CELLS
!
    IBOUND = 0

    DO I = 1, IMAX
      DO J = 1, JMAX
```

```
      IF ( IAND ( FLAG(I,J), C_F ) /= C_F ) THEN

        IBOUND = IBOUND + 1

      END IF

      FLAG(I,J) = FLAG(I,J) + ( IAND ( FLAG(I-1,J), C_F ) * B_W  &
                            +   IAND ( FLAG(I+1,J), C_F ) * B_E  &
                            +   IAND ( FLAG(I,J-1), C_F ) * B_S  &
                            +   IAND ( FLAG(I,J+1), C_F ) * B_N ) /
C_F
     END DO
    END DO
!*************************!
!
!   INITIALIZE THE BOUNDARY CONDITIONS.
!
!   DO J = 0, JMAX+1
!
!   WESTERN AND EASTERN BOUNDARY.
!
!   FREE SLIP, U = 0, DVDN = 0.
!
!     IF ( WW == 1 ) THEN

        U(0,J) = 0.0
        V(0,J) = V(1,J)
!
!   NO SLIP, U = 0, V = 0 AT THE BOUNDARY BY AVERAGING.
!
      ELSE IF ( WW == 2 ) THEN

        U(0,J) = 0.0
        V(0,J) = (-1.0) * V(1,J)
!
!   OUTFLOW
!
      ELSE IF ( WW == 3 ) THEN

        U(0,J) = U(1,J)
        V(0,J) = V(1,J)
!
!   PERIODIC, LEFT AND RIGHT CELLS OVERLAP.
!
      ELSE IF ( WW == 4 ) THEN

        U(0,J) = U(IMAX-1,J)
        V(0,J) = V(IMAX-1,J)
        V(1,J) = V(IMAX,J)
        P(1,J) = P(IMAX,J)

      END IF
!
!   FREE SLIP
!
```

```
      IF ( WE == 1 ) THEN

        U(IMAX,J) = 0.0
        V(IMAX+1,J) = V(IMAX,J)
!
!   NO SLIP
!
      ELSE IF ( WE == 2 ) THEN

        U(IMAX,J) = 0.0
        V(IMAX+1,J) = -V(IMAX,J)
!
!   OUTFLOW
!
      ELSE IF ( WE == 3 ) THEN

        U(IMAX,J) = U(IMAX-1,J)
        V(IMAX+1,J) = V(IMAX,J)
!
!   PERIODIC
!
      ELSE IF ( WE == 4 ) THEN

        U(IMAX,J) = U(1,J)
        V(IMAX+1,J) = V(2,J)
      END IF

    END DO
!
!   NORTHERN AND SOUTHERN BOUNDARY
!
    DO I = 0, IMAX+1

      IF ( WN == 1 ) THEN
        V(I,JMAX) = 0.0
        U(I,JMAX+1) = U(I,JMAX)
      ELSE IF ( WN == 2 ) THEN
        V(I,JMAX) = 0.0
        U(I,JMAX+1) = -U(I,JMAX)
      ELSE IF ( WN == 3 ) THEN
        V(I,JMAX) = V(I,JMAX-1)
        U(I,JMAX+1) = U(I,JMAX)
      ELSE IF ( WN == 4 ) THEN
        V(I,JMAX) = V(I,1)
        U(I,JMAX+1) = U(I,2)
      END IF


      IF ( WS == 1 ) THEN
        V(I,0) = 0.0
        U(I,0) = U(I,1)
      ELSE IF ( WS == 2 ) THEN
        V(I,0) = 0.0
        U(I,0) = -U(I,1)
      ELSE IF ( WS == 3 ) THEN
        V(I,0) = V(I,1)
        U(I,0) = U(I,1)
      ELSE IF ( WS == 4 ) THEN
```

```
      V(I,0) = V(I,JMAX-1)
      U(I,0) = U(I,JMAX-1)
      U(I,1) = U(I,JMAX)
      P(I,1) = P(I,JMAX)
    END IF

  END DO
!
!  SET THE BOUNDARY VALUES AT INNER OBSTACLE CELLS (ONLY NO-SLIP).
!
  DO I = 1, IMAX
    DO J = 1, JMAX
!
!  MASK C_X = 000F FILTERS THE OBSTACLE CELLS ADJACENT TO FLUID CELLS.
!
      IF ( IAND ( FLAG(I,J), C_X ) /= 0 ) THEN

        SELECT CASE ( FLAG(I,J) )

          CASE (B_N)
            V(I,J)   = 0.0
            U(I,J)   = -U(I,J+1)
            U(I-1,J) = -U(I-1,J+1)
          CASE (B_E)
            U(I,J)   = 0.0
            V(I,J)   = -V(I+1,J)
            V(I,J-1) = -V(I+1,J-1)
          CASE (B_S)
            V(I,J-1) = 0.0
            U(I,J)   = -U(I,J-1)
            U(I-1,J) = -U(I-1,J-1)
          CASE (B_W)
            U(I-1,J) = 0.0
            V(I,J)   = -V(I-1,J)
            V(I,J-1) = -V(I-1,J-1)
          CASE (B_NE)
            V(I,J)   = 0.0
            U(I,J)   = 0.0
            V(I,J-1) = -V(I+1,J-1)
            U(I-1,J) = -U(I-1,J+1)
          CASE (B_SE)
            V(I,J-1) = 0.0
            U(I,J)   = 0.0
            V(I,J)   = -V(I+1,J)
            U(I-1,J) = -U(I-1,J-1)
          CASE (B_SW)
            V(I,J-1) = 0.0
            U(I-1,J) = 0.0
            V(I,J)   = -V(I-1,J)
            U(I,J)   = -U(I,J-1)
          CASE (B_NW)
            V(I,J)   = 0.0
            U(I-1,J) = 0.0
            V(I,J-1) = -V(I-1,J-1)
            U(I,J)   = -U(I,J+1)
          CASE DEFAULT
```

```
      END SELECT

    END IF

  END DO
END DO
!****************************!
!
!  BACKWARDSTEP
!  U = 1.0 AT THE LEFT BOUNDARY
!
IF ( PROBLEM == "BACKWARDSTEP" ) THEN

  DO J = JMAX/2+1, JMAX
    U(0,J) = 1.0
  END DO

!
!  CYLINDER
!  U = 1.0 AT LEFT BOUNDARY
!
ELSE IF ( PROBLEM == "CYLINDER" ) THEN

  V(0,0) = 2.0 * VI - V(1,0)
  DO J = 1, JMAX
    U(0,J) = UI
    V(0,J) = 2.0 * VI - V(1,J)
  END DO

!
!  CAVITY:
!
!  U = 1.0 AT THE UPPER BOUNDARY
!
ELSE IF ( PROBLEM == "CAVITY" ) THEN

  DO I = 0, IMAX
    U(I,JMAX+1) = 2.0 - U(I,JMAX)
  END DO

END IF

T = 0.0
CYCLES = 0
!****************************!
!
!  TIME LOOP.
!
DO WHILE ( T < T_END ) = 5

!****************************!
!
!  DETERMINE FLUID CELLS
!
IFULL = IMAX * JMAX - IBOUND
!****************************!
!
!  COMPUTE TENTATIVE VELOCITY FIELD (F,G).
!
      !
```

```
!  COMPUTE FLUX FIELD F
!
!  ONLY IF BOTH ADJACENT CELLS ARE FLUID CELLS.
!
  DO I = 1, IMAX-1
       DO J = 1, JMAX

          IF ((( IAND ( FLAG(I,J),  C_F) /= 0) .AND. FLAG(I,J)
< C_E ) .AND.&
                  (( IAND ( FLAG(I+1,J),C_F) /= 0) .AND.
FLAG(I+1,J) < C_E ) ) THEN

                  DU2DX = ((U(I,J)+U(I+1,J))*(U(I,J)+U(I+1,J)) &
                                  + GAMMA*ABS (
U(I,J)+U(I+1,J))*(U(I,J)-U(I+1,J)) &
                                  - (U(I-
1,J)+U(I,J))*(U(I-1,J)+U(I,J)) &
                                  - GAMMA*ABS ( U(I-
1,J)+U(I,J))*(U(I-1,J)-U(I,J))) &
                                  / (4.0*DELX)

                  DUVDY = ((V(I,J)+V(I+1,J))*(U(I,J)+U(I,J+1)) &
                                  + GAMMA*ABS (
V(I,J)+V(I+1,J))*(U(I,J)-U(I,J+1)) &
                                  - (V(I,J-1)+V(I+1,J-
1))*(U(I,J-1)+U(I,J)) &
                                  - GAMMA*ABS ( V(I,J-
1)+V(I+1,J-1))*(U(I,J-1)-U(I,J))) &
                                  / ( 4.0 * DELY )

                  LAPLU = ( U(I+1,J) - 2.0 * U(I,J) + U(I-1,J) ) /
DELX / DELX &
                                  + ( U(I,J+1) - 2.0 *
U(I,J) + U(I,J-1) ) / DELY / DELY

                  F(I,J) = U(I,J) + DELT * ( LAPLU / RE - DU2DX -
DUVDY + GX )

                ELSE

                     F(I,J) = U(I,J)

                END IF

             END DO
        END DO
!
!  COMPUTE FLUX FIELD G
!
!  ONLY IF BOTH ADJACENT CELLS ARE FLUID CELLS
!
        DO I = 1, IMAX
             DO J = 1, JMAX-1
                          IF ((( IAND ( FLAG(I,J),
C_F)/=0).AND.(FLAG(I,J)<C_E)).AND.&
```

```
                                        (( IAND (
FLAG(I,J+1),C_F)/=0).AND.(FLAG(I,J+1)<C_E))) THEN
                                    DUVDX =
((U(I,J)+U(I,J+1))*(V(I,J)+V(I+1,J)) &
                                          + GAMMA*ABS (
U(I,J)+U(I,J+1))*(V(I,J)-V(I+1,J)) &
                                          - (U(I-1,J)+U(I-
1,J+1))*(V(I-1,J)+V(I,J)) &
                                          - GAMMA*ABS ( U(I-
1,J)+U(I-1,J+1))*(V(I-1,J)-V(I,J))) &
                                       / ( 4.0 * DELX )

                                    DV2DY =
((V(I,J)+V(I,J+1))*(V(I,J)+V(I,J+1)) &
                                          + GAMMA*ABS (
V(I,J)+V(I,J+1))*(V(I,J)-V(I,J+1)) &
                                          - (V(I,J-
1)+V(I,J))*(V(I,J-1)+V(I,J)) &
                                          - GAMMA*ABS ( V(I,J-
1)+V(I,J))*(V(I,J-1)-V(I,J))) &
                                       / ( 4.0 * DELY )

                                 LAPLV = ( V(I+1,J) - 2.0 * V(I,J) +
V(I-1,J) ) / DELX / DELX &
                                       + ( V(I,J+1) - 2.0 *
V(I,J) + V(I,J-1) ) / DELY / DELY

                                 G(I,J) = V(I,J) + DELT * ( LAPLV / RE
- DUVDX - DV2DY + GY )

                              ELSE

                                 G(I,J) = V(I,J)

                              END IF

                           END DO
                     END DO
               !
               !  F AND G AT EXTERNAL BOUNDARY
               !
                 DO J = 1, JMAX
                       F(0,J)    = U(0,J)
                       F(IMAX,J) = U(IMAX,J)
                 END DO

                 DO I = 1, IMAX
                       G(I,0)    = V(I,0)
                       G(I,JMAX) = V(I,JMAX)
                 END DO
!****************************!
!
!  COMPUTE RIGHT-HAND SIDE FOR PRESSURE EQUATION.
!
                 DO I = 1, IMAX
                       DO J = 1, JMAX
               !
```

```
!  ONLY FOR FLUID AND NON-SURFACE CELLS.
!
            IF ( IAND ( FLAG(I,J), C_F ) /= 0 .AND. FLAG(I,J) <
C_O ) THEN
                RHS(I,J) = ( ( F(I,J) - F(I-1,J) ) / DELX &
                           +   ( G(I,J) - G(I,J-1) ) / DELY
) / DELT
            END IF

          END DO
        END DO
!*************************!
!
!  SOLVE THE PRESSURE EQUATION BY SUCCESSIVE OVER RELAXATION (SOR).
!
 IF ( 0 < IFULL ) THEN

          RDX2 = 1.0 / DELX / DELX
          RDY2 = 1.0 / DELY / DELY
          BETA_2 = - OMEGA / ( 2.0 * ( RDX2 + RDY2 ) )

          P0 = 0.0

          DO I = 1, IMAX
               DO J = 1, JMAX

                 IF ( IAND ( FLAG(I,J), C_F ) /= 0 ) THEN
                      P0 = P0 + P(I,J) * P(I,J)
                 END IF

               END DO
          END DO

          P0 = SQRT ( P0 / IFULL )

          IF ( P0 < 0.0001 ) THEN
               P0 = 1.0
          END IF
!
!  SOR ITERATION
!
          DO ITER = 1, ITERMAX
!
!  COPY VALUES AT EXTERNAL BOUNDARY...
!
               DO I = 1, IMAX
                    P(I,0)      = P(I,1)
                    P(I,JMAX+1) = P(I,JMAX)
               END DO

               DO J = 1, JMAX
                    P(0,J)      = P(1,J)
                    P(IMAX+1,J) = P(IMAX,J)
               END DO
!
!  AND AT INTERIOR BOUNDARY CELLS.
!
```

```
                DO I = 1, IMAX
                    DO J = 1, JMAX

                IF ( B_N <= FLAG(I,J) .AND. FLAG(I,J) <= B_SE
) THEN
                                IF ( FLAG(I,J) == B_N ) THEN
                                    P(I,J) = P(I,J+1)
                                ELSE IF ( FLAG(I,J) ==
B_E ) THEN
                                    P(I,J) = P(I+1,J)
                                ELSE IF ( FLAG(I,J) ==
B_S ) THEN
                                    P(I,J) = P(I,J-1)
                                ELSE IF ( FLAG(I,J) ==
B_W ) THEN
                                    P(I,J) = P(I-1,J)
                                ELSE IF ( FLAG(I,J) ==
B_NE ) THEN
                                    P(I,J) = 0.5 * (
P(I,J+1) + P(I+1,J) )
                                ELSE IF ( FLAG(I,J) ==
B_SE ) THEN
                                    P(I,J) = 0.5 * ( P(I,J-
1) + P(I+1,J) )
                                ELSE IF ( FLAG(I,J) ==
B_SW ) THEN
                                    P(I,J) = 0.5 * ( P(I,J-
1) + P(I-1,J) )
                                ELSE IF ( FLAG(I,J) ==
B_NW ) THEN
                                    P(I,J) = 0.5 * (
P(I,J+1) + P(I-1,J) )
                                END IF
                            END IF

                        END DO
                    END DO
                !
                !  RELAXATION METHOD FOR FLUID CELLS.
                !
                    DO I = 1,IMAX
                        DO J = 1,JMAX

                IF ( IAND ( FLAG(I,J),C_F) /= 0 .AND.
FLAG(I,J) < C_O ) THEN
                            P(I,J) = ( 1.0 - OMEGA ) * P(I,J)
&
                                - BETA_2 * (
(P(I+1,J) + P(I-1,J)) * RDX2 &
                                +
(P(I,J+1) + P(I,J-1)) * RDY2 &
                                - RHS(I,J) )
                            END IF

                        END DO
                    END DO
                !
```

```
!  COMPUTATION OF THE RESIDUAL.
!
                        RES = 0.0

                        DO I = 1,IMAX
                              DO J = 1,JMAX
!
!  ONLY FLUID CELLS
!
                    IF ( ( IAND ( FLAG(I,J),C_F) /= 0 ) .AND.
FLAG(I,J) < C_O ) THEN
                                    ADD = ( P(I+1,J) - 2.0 * P(I,J) +
P(I-1,J)) * RDX2 &
                                        + ( P(I,J+1) - 2.0 * P(I,J) +
P(I,J-1)) * RDY2 - RHS(I,J)
                                    RES = RES + ADD * ADD
                        END IF

                              END DO
                        END DO

                        RES = SQRT ( RES / IFULL ) / P0
                        IF ( RES < EPS ) THEN
                              GO TO 100
                        END IF
                        ITERSOR=ITERSOR+1
                  END DO

100      WRITE (6, '(A,G10.4,A,G10.4,A,I4,A,G12.6,A,I4,I4,I4)' ) &
         ' T=', T+DELT, ' DELT=', DELT, ' ITS=', ITERSOR, ' RES=', RES

      END IF
         ITERSOR=0
!**************************!
!
!  COMPUTE THE NEW VELOCITY FIELD.
!
         DO I = 1, IMAX - 1
               DO J = 1, JMAX

                  IF ((( IAND ( FLAG(I,J),   C_F ) > 0 ) .AND. (FLAG(I,
J) < C_E ) ) .AND. &
                              (( IAND ( FLAG(I+1,J), C_F ) > 0 ) .AND.
(FLAG(I+1,J) < C_E ) )) THEN
                        U(I,J) = F(I,J) - ( P(I+1,J) - P(I,J) ) * DELT /
DELX

                  END IF

               END DO
         END DO

         DO I = 1, IMAX
               DO J = 1, JMAX - 1
```

```
                IF ((( IAND ( FLAG(I,J),  C_F ) > 0 ) .AND. (FLAG(I,
J) < C_E ) ) .AND.&
                   (( IAND ( FLAG(I,J+1),C_F ) > 0 ) .AND.
(FLAG(I,J+1) < C_E ) )) THEN
                      V(I,J) = G(I,J) - ( P(I,J+1) - P(I,J) ) * DELT /
DELY
                   END IF

                END DO
             END DO
!*************************!
!
!  SET THE BOUNDARY CONDITIONS FOR THE NEXT TIME STEP
!
             DO J = 0, JMAX+1
!
!  WESTERN AND EASTERN BOUNDARY.
!
!  FREE SLIP, U = 0, DVDN = 0.
!
          IF ( WW == 1 ) THEN

            U(0,J) = 0.0
            V(0,J) = V(1,J)
!
!  NO SLIP, U = 0, V = 0 AT THE BOUNDARY BY AVERAGING.
!
          ELSE IF ( WW == 2 ) THEN

            U(0,J) = 0.0
            V(0,J) = (-1.0) * V(1,J)
!
!  OUTFLOW
!
          ELSE IF ( WW == 3 ) THEN

            U(0,J) = U(1,J)
            V(0,J) = V(1,J)
!
!  PERIODIC, LEFT AND RIGHT CELLS OVERLAP.
!
          ELSE IF ( WW == 4 ) THEN

            U(0,J) = U(IMAX-1,J)
            V(0,J) = V(IMAX-1,J)
            V(1,J) = V(IMAX,J)
            P(1,J) = P(IMAX,J)

          END IF
!
!  FREE SLIP
!
          IF ( WE == 1 ) THEN

            U(IMAX,J) = 0.0
            V(IMAX+1,J) = V(IMAX,J)
```

```
!
!  NO SLIP
!
        ELSE IF ( WE == 2 ) THEN

          U(IMAX,J) = 0.0
          V(IMAX+1,J) = -V(IMAX,J)
!
!  OUTFLOW
!
        ELSE IF ( WE == 3 ) THEN

          U(IMAX,J) = U(IMAX-1,J)
          V(IMAX+1,J) = V(IMAX,J)
!
!  PERIODIC
!
        ELSE IF ( WE == 4 ) THEN

          U(IMAX,J) = U(1,J)
          V(IMAX+1,J) = V(2,J)
        END IF

      END DO
!
!   NORTHERN AND SOUTHERN BOUNDARY
!
      DO I = 0, IMAX+1

        IF ( WN == 1 ) THEN
          V(I,JMAX) = 0.0
          U(I,JMAX+1) = U(I,JMAX)
        ELSE IF ( WN == 2 ) THEN
          V(I,JMAX) = 0.0
          U(I,JMAX+1) = -U(I,JMAX)
        ELSE IF ( WN == 3 ) THEN
          V(I,JMAX) = V(I,JMAX-1)
          U(I,JMAX+1) = U(I,JMAX)
        ELSE IF ( WN == 4 ) THEN
          V(I,JMAX) = V(I,1)
          U(I,JMAX+1) = U(I,2)
        END IF


        IF ( WS == 1 ) THEN
          V(I,0) = 0.0
          U(I,0) = U(I,1)
        ELSE IF ( WS == 2 ) THEN
          V(I,0) = 0.0
          U(I,0) = -U(I,1)
        ELSE IF ( WS == 3 ) THEN
          V(I,0) = V(I,1)
          U(I,0) = U(I,1)
        ELSE IF ( WS == 4 ) THEN
          V(I,0) = V(I,JMAX-1)
          U(I,0) = U(I,JMAX-1)
          U(I,1) = U(I,JMAX)
```

```
                P(I,1) = P(I,JMAX)
              END IF

          END DO
        !
        !  SET THE BOUNDARY VALUES AT INNER OBSTACLE CELLS (ONLY NO-
SLIP).
        !
          DO I = 1, IMAX
               DO J = 1, JMAX
        !
        !  MASK C_X = 000F FILTERS THE OBSTACLE CELLS ADJACENT TO FLUID
CELLS.
        !
                  IF ( IAND ( FLAG(I,J), C_X ) /= 0 ) THEN

                    SELECT CASE ( FLAG(I,J) )

                      CASE (B_N)
                    V(I,J)   = 0.0
                             U(I,J)   = -U(I,J+1)
                             U(I-1,J) = -U(I-1,J+1)
                      CASE (B_E)
                    U(I,J)   = 0.0
                             V(I,J)   = -V(I+1,J)
                             V(I,J-1) = -V(I+1,J-1)
                      CASE (B_S)
                    V(I,J-1) = 0.0
                             U(I,J)   = -U(I,J-1)
                             U(I-1,J) = -U(I-1,J-1)
                      CASE (B_W)
                    U(I-1,J) = 0.0
                             V(I,J)   = -V(I-1,J)
                             V(I,J-1) = -V(I-1,J-1)
                      CASE (B_NE)
                    V(I,J)   = 0.0
                             U(I,J)   = 0.0
                             V(I,J-1) = -V(I+1,J-1)
                             U(I-1,J) = -U(I-1,J+1)
                       CASE (B_SE)
                    V(I,J-1) = 0.0
                             U(I,J)   = 0.0
                             V(I,J)   = -V(I+1,J)
                             U(I-1,J) = -U(I-1,J-1)
                       CASE (B_SW)
                    V(I,J-1) = 0.0
                             U(I-1,J) = 0.0
                             V(I,J)   = -V(I-1,J)
                             U(I,J)   = -U(I,J-1)
                       CASE (B_NW)
                    V(I,J)   = 0.0
                             U(I-1,J) = 0.0
                             V(I,J-1) = -V(I-1,J-1)
                             U(I,J)   = -U(I,J+1)
                CASE DEFAULT

                    END SELECT
```

```
            END IF

          END DO
        END DO
!*****************************!
!  SETTING SPECIAL BOUNDARY CONDITIONS
!  FOR THE NEXT TIME STEP.
      !
      !  BACKWARDSTEP
      !  U = 1.0 AT THE LEFT BOUNDARY
      !
  IF ( PROBLEM == "BACKWARDSTEP" ) THEN

      DO J = JMAX/2+1, JMAX
        U(0,J) = 1.0
      END DO
!
!  CYLINDER
!  U = 1.0 AT LEFT BOUNDARY
!
  ELSE IF ( PROBLEM == "CYLINDER" ) THEN

      V(0,0) = 2.0 * VI - V(1,0)
      DO J = 1, JMAX
        U(0,J) = UI
        V(0,J) = 2.0 * VI - V(1,J)
      END DO
!
      !  CAVITY:
      !
      !  U = 1.0 AT THE UPPER BOUNDARY
      !
  ELSE IF ( PROBLEM == "CAVITY" ) THEN

      DO I = 0, IMAX
        U(I,JMAX+1) = 2.0 - U(I,JMAX)
      END DO
      !
      !  UNRECOGNIZED PROBLEM.
      !
    END IF
!*****************************!
      !WRITE(*,*) T
      !WRITE(*,*) DELT
      !WRITE(*,*) CYCLES
!*****************************!
!
!  ADVANCE THE TIME
!

    T = T + DELT
  CYCLES = CYCLES + 1

    IF (MOD(CYCLES,OUTCOUNT)==0) THEN
          !CALL DIGIT_TO_CH (CYCLES, OUTNUM )
          !OUTNUM=CHAR (CYCLES)
          PRINTLABEL = CYCLES/OUTCOUNT
```

```
                WRITE (OUTNUM,'(I2)') PRINTLABEL
                !WRITE(*,*) PRINTLABEL
                !READ(OUTNUM,*) CYCLES
                !
                !  COMPUTATION OF THE VORTICITY (ZETA) AT THE UPPER
RIGHT CORNER
                !  OF CELL (I,J) (ONLY IF THE CORNER IS SURROUNDED BY
FLUID CELLS)
                !
                        DO I = 1, IMAX-1
                             DO J = 1, JMAX-1

                                IF (((IAND(FLAG(I,  J),C_F) /= 0 )
.AND. FLAG(I,J) < C_E ) .AND.&
                                    ((IAND(FLAG(I+1,J),C_F) /= 0 )
.AND. FLAG(I+1,J) < C_E ) .AND.&
                                    ((IAND(FLAG(I,J+1),C_F) /= 0 )
.AND. FLAG(I,J+1) < C_E ) .AND.&
                                    ((IAND(FLAG(I+1,J+1),C_F) /= 0
) .AND. FLAG(I+1,J+1) < C_E ))&
                                THEN

                                    ZETA(I,J) = ( U(I,J+1) - U(I,J) )
/ DELY &
                                    -( V(I+1,J) -
V(I,J) ) / DELX
                                ELSE

                                    ZETA(I,J) = 0.0

                                END IF

                             END DO
                        END DO
                        ZETA(0,0:JMAX)=0.0
                        ZETA(0:IMAX,0)=0.0
                        ZETA(IMAX,0:JMAX)=0.0
                        ZETA(0:IMAX,JMAX)=0.0
                !****************************!
                !
                !  COMPUTATION OF THE STREAM FUNCTION AT THE UPPER
RIGHT CORNER
                !  OF CELL (I,J), BUT ONLY IF BOTH LOWER CELLS ARE
FLUID CELLS.
                !
                        DO I = 0, IMAX
                             PSI(I,0) = 0.0
                             DO J = 1, JMAX

                                IF (((IAND(FLAG(I,J),  C_F) /= 0 )
.AND. (FLAG(I,  J)<C_E)) .OR. &
                                    ((IAND(FLAG(I+1,J),C_F) /= 0 )
.AND. (FLAG(I+1,J)<C_E))) THEN
                                    PSI(I,J) = PSI(I,J-1) + U(I,J) *
DELY
                                ELSE
                                    PSI(I,J) = PSI(I,J-1)
```

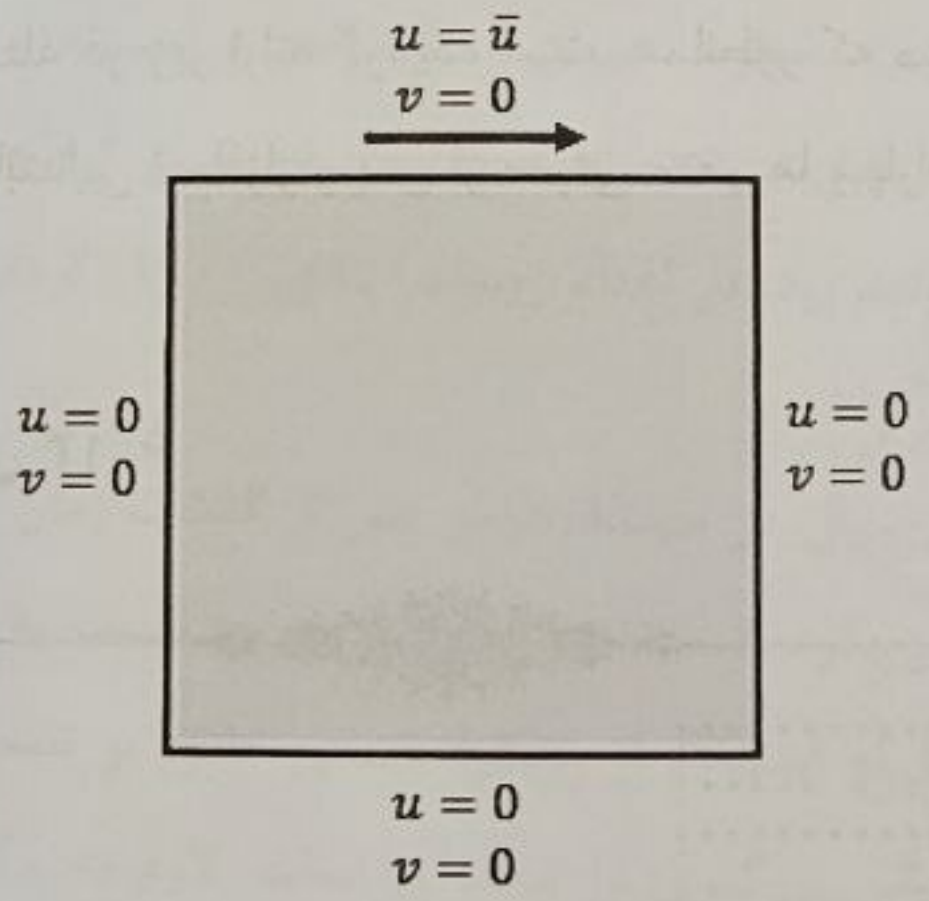
```
                    END IF

                  END DO
                END DO
!*****************************!
!
!  COMPUTING OUTPUT VELOCITY AND PRESSURE
!  DATA AND WRITING THEM OUT IN SEPERATE
!  FILES FOR POST-PROCESSING
!
      DO J = 1, JMAX
        DO I = 1, IMAX
          IF ( IAND ( FLAG(I,J), C_F ) /= 0
.AND. FLAG(I,J) < C_E ) THEN
            UOUT(I,J) = ( U(I,J) + U(I-1,J) )
/ 2.0
          ELSE
            UOUT(I,J) = 0.0
          END IF
        END DO
      END DO
      UOUT(0,0:JMAX)=0.0
      UOUT(0:IMAX,0)=0.0

      DO J = 1, JMAX
        DO I = 1, IMAX
          IF ( IAND ( FLAG(I,J), C_F ) /= 0
.AND. FLAG(I,J) < C_E ) THEN
            VOUT(I,J) = ( V(I,J) + V(I,J-1) )
/ 2.0
          ELSE
            VOUT(I,J) = 0.0
          END IF
        END DO
      END DO
      VOUT(0,0:JMAX)=0.0
      VOUT(0:IMAX,0)=0.0

      DO J = 1, JMAX
        DO I = 1, IMAX
          IF ( IAND ( FLAG(I,J), C_F ) /= 0
.AND. FLAG(I,J) < C_E ) THEN
            POUT(I,J) = P(I,J)
          ELSE
            POUT(I,J) = 0.0
          END IF
        END DO
      END DO
      POUT(0,0:JMAX)=0.0
      POUT(0:IMAX,0)=0.0
!*****************************!
!
!  OUTPUT U-VELOCITY FOR VISUALIZATION.
!
      OPEN ( UNIT=1, FILE =
'U_DATA_'//TRIM(OUTNUM)//'.TXT')
```

```
                        DO J = 0, JMAX
                             DO I = 0, IMAX
                               WRITE ( 1, '(1X,I5,1X,I5,1X,F12.6)' )
I, J, UOUT(I,J)
                             END DO
                        END DO

                        CLOSE (1)

                    !****************************!
                    !
                    !  OUTPUT V-VELOCITY FOR VISUALIZATION.
                    !
                        OPEN ( 2, FILE =
'V_DATA_'//TRIM(OUTNUM)//'.TXT')
                        DO J = 0, JMAX
                             DO I = 0, IMAX
                               WRITE ( 2, '(1X,I5,1X,I5,1X,F12.6)' )
I, J, VOUT(I,J)
                             END DO
                        END DO

                        CLOSE (2)

                    !****************************!
                    !
                    !  OUTPUT PRESSURE FIELD FOR VISUALIZATION.
                    !
                        OPEN ( 3, FILE =
'PRESSURE_'//TRIM(OUTNUM)//'.TXT')
                        DO J = 0, JMAX
                             DO I = 0, IMAX
                               WRITE ( 3, '(1X,I5,1X,I5,1X,F12.6)' )
I, J, POUT(I,J)
                             END DO
                        END DO

                        CLOSE (3)

                    !****************************!
                    !
                    !  OUTPUT THE STREAM FUNCTION FOR VISUALIZATION.
                    !
                        OPEN ( 4, FILE =
'PSI_DATA_'//TRIM(OUTNUM)//'.TXT')
                        DO J = 0, JMAX
                             DO I = 0, IMAX
                               WRITE ( 4, '(1X,I5,1X,I5,1X,F12.6)' )
I, J, PSI(I,J)
                             END DO
                        END DO

                        CLOSE (4)

                    !****************************!
```

```
!
!  OUTPUT THE VORTICITY FOR VISUALIZATION.
!
                        OPEN ( 5, FILE =
'ZETA_DATA_'//TRIM(OUTNUM)//'.TXT')

                      DO J = 0, JMAX
                           DO I = 0, IMAX
                             WRITE ( 5, '(1X,I5,1X,I5,1X,F12.6)' )
I, J, ZETA(I,J)
                           END DO
                      END DO

                        CLOSE (5)
                !*****************************!
                WRITE(*,*) ' '
                WRITE(*,*) 'PROBLEM: ',PROBLEM
                WRITE(*,*) 'RESULTS HAVE BEEN PRINTED OUT'
                WRITE(*,*) 'FOR TIME ',T,' (S)'
                WRITE(*,*) ' '

      ENDIF


  END DO
!*****************************!
!&&&&&&&&&&&&&&&&&&&&&&&&&&!
!*****************************!
!$$$$$$$$$$$$$$$$$$$$$$$$!
!*****************************!

!
!  FREE MEMORY.
!
  DEALLOCATE ( F )
  DEALLOCATE ( FLAG )
  DEALLOCATE ( G )
  DEALLOCATE ( P )
  DEALLOCATE ( PSI )
  DEALLOCATE ( RHS )
  DEALLOCATE ( U )
  DEALLOCATE ( V )
  DEALLOCATE ( ZETA )
  DEALLOCATE ( UOUT )
  DEALLOCATE ( VOUT )
  DEALLOCATE ( POUT )

  STOP

END
```

۱۲-۴-۳ مسئلهٔ کاربردی ۵: جریان داخل حفره

نخستین مسئله‌ای که با استفاده از کد عددی معادلات ناویر-استوکس به روش حجم محدود مورد بررسی قرار می گیرد، جریان داخل حفره است. همان طور که قبلاً بیان شد، جریان درون حفره یکی از مسائل شناخته شده و معتبر در زمینهٔ دینامیک سیّالات محاسباتی بوده و در بسیاری از کارهای تحقیقاتی از آن برای معتبرسازی و صحّت-سنجی نتایج بدست آمده از حلگرهای عددی، استفاده می‌شود. مشخّصات فیزیکی این مسئله همانطور که در شکل نیز نمایش داده شده است، نشان می‌دهد که جریان در داخل یک حفرهٔ مربعی شکل که با یک سیال بطور کامل پر شده است، مورد بررسی قرار خواهد گرفت. دیوارهٔ فوقانی این حفره با یک سرعت ثابت و مشخص بطور دائم در حال حرکت بوده و در نتیجه سیال را به حرکت وادار می نماید.



شکل(۱۲-۲۴): مؤلّفه های سرعت در سطوح مختلف

شرایط دیوارهٔ بدون لغزش بر روی هر چهار مرز اعمال گردیده، با این تفاوت که در مرز بالایی، سرعت u در جهت x صفر نبوده و برابر با سرعت حرکت دیواره، $\bar{u}$ خواهد بود تا بتوان حرکت آنرا شبیه‌سازی نمود. در نتیجه در برنامهٔ کامپیوتری شرایط مرزی در مرز بالایی به صورت زیر درنظر گرفته شده‌اند.

$$u_{i,j_{\max+1}} = 2.0\,\bar{u} - u_{(i,j_{\max})} \qquad i = 1, \ldots, i_{\max}$$

همچنین ابعاد مربع 1×1 در نظر گرفته شده و در هر جهت به ۱۲۸ قسمت مساوی تقسیم‌بندی گردیده تا به یک شبکه‌بندی منظم و یکنواخت ۱۲۸ × ۱۲۸ دست یابیم. در شکل‌هایی که در ادامه ارائه گردیده‌اند، میدان سرعت و خطوط جریان برای مسئله با عدد رینولدز ۱۰۰۰ ($Re = 1000$) هم در حالت غیردائمی و متغیر با زمان و هم در حالت پایا نشان داده می‌شود. در لحظهٔ اوّل ($t = 0$) سرعت درپوش $\bar{u}$ بطور آنی اعمال گردیده و سبب ایجاد تحرک در داخل سیال می گردد. تشکیل گردابه اصلی و بزرگ راستگرد و متعاقب آن دومین گردابه که چپ‌گرد است، در گوشهٔ سمت راست و پایین، به خوبی قابل مشاهده می‌باشند. اما گردابهٔ چپ‌گرد بعدی در گوشه سمت چپ پایین به زمان بیشتری جهت تشکیل نیاز خواهد داشت.

کد تنظیمات این مسئله در زیر ارائه گردیده است. همانطور که مشخص می‌باشد، این کد بایستی در بخش ابتدایی نرم‌افزار و پس از معرفی متغیرها و پارامترهای نرم‌افزار قرار داده شود.

برنامه کامپیوتری ۱۲-۶

```
!****************************!
!***** PROBLEM SETUP *****!
!****************************!
!  INITIALIZE.
!
  PROBLEM='CAVITY'

  XLENGTH=1.0
  YLENGTH=1.0
  IMAX=50
  JMAX=50

  DELX = XLENGTH / IMAX
  DELY = YLENGTH / JMAX

  T_END=5.0
  DELT=0.005
  OUTCOUNT=200

  ITERMAX=150
```

```
EPS=0.001
OMEGA=1.7
GAMMA=0.9

RE=100
GX=0.0
GY=0.0
UI=0.0
VI=0.0

WW=2
WE=2
WN=2
WS=2

ITERSOR = 0
IFULL = 0
IBOUND = 0
!*************************!
```

نخست، میدان سرعت، خطوط جریان و کانتورهای ورتیسیته در اعداد رینولدز متفاوت نشان داده می‌شود. قابل مشاهده است که ابعاد نخستین گردابه چپ‌گرد، کاملاً به عدد رینولدز وابسته خواهد بود. در اعداد رینولدز بالای ۱۰۰۰، گردابهٔ چپگرد دوّم در گوشهٔ پایینی تشکیل می‌گردد.

از آنجاییکه جریان سیال در مسئلهٔ حاضر پس از گذشت زمان به شرایط دائمی خواهد رسید، استفاده از رویکرد گسسته‌سازی زمانی صریح نسبت به رویکرد ضمنی از ضعف برخوردار بوده زیرا که استفاده از رویکرد ضمنی علاوه بر تضمین پایداری حل، امکان استفاده از بازهٔ زمانی بزرگتری را ایجاد می‌نماید. لازم به ذکر است که در مطالعات [Ghia et al., 1982] که بعنوان مرجع صحّت سنجی مسئلهٔ حفره در جوامع علمی پذیرفته شده است، معادلات ناویر-استوکس در شکل دائمی خود و با استفاده از فرمولاسیون گردابه - تابع جریان حل گردیده و در نتیجه تغییرات جریان در طول زمان در مطالعات آنها قابل مشاهده نمی‌باشد.

نکتهٔ دیگری که بایستی به آن توجه نمود، این است که بعلّت اختلاف بسیار کم میان مقادیر خطوط جریان در محل گردابه‌های چپگرد، کانتورهای خطّ جریان در این نواحی

بطور مجزّا و با مقادیر دقیق‌تری رسم گردیده و نمایش داده شده‌اند. همچنین برای مشاهدهٔ نتایج بدست آمده و پردازش آنها، یک کد نمایشگر در محیط نرم‌افزار **MATLAB** برنامه‌نویسی شده که در زیر ارائه می‌گردد.

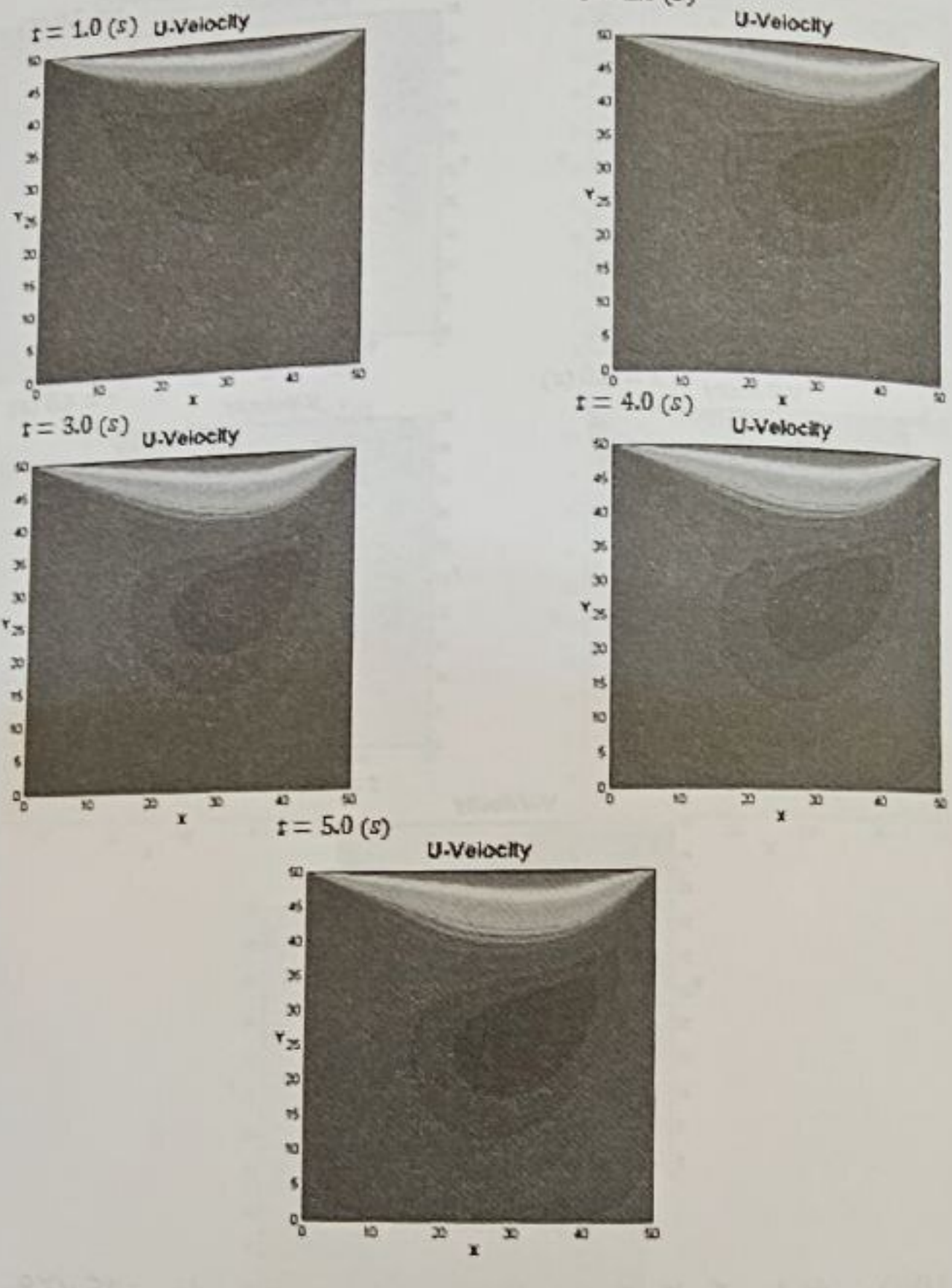
```
FUNCTION PGRG2D_POSTPROC(TIMELEVEL)
  FPRINTF ( 1, '\N' );
  FPRINTF ( 1, 'PGRG2DNAVISTO_CONTOUR:\N' );
  FPRINTF ( 1, ' THIS PROGRAM PLOTS CONTOURS FROM THE OUTPUT FILE\N'
);
  FPRINTF ( 1, ' GENERATED BY PGRG2DNAVISTO CODE.\N' );

  % CAVITY = DRIVEN CAVITY
  % STEP = FLOW OVER BACKWARD STEP
  % CYLINDER = FLOW AROUND CIRCULAR CYLINDER
  %NUMBER OF CONTOUR LEVELS
  NO_LEVELS = 20;
  PROBLEM = 'CAVITY';
  VECTORPLOT = 'N' ; % (Y)ES= PLOT VECTORS    (N)O= NO VECTOR-PLOT
  LX = 30;
  LY = 1.5;
  UFILENAME=['U_DATA_' TIMELEVEL '.TXT'];
  VFILENAME=['V_DATA_' TIMELEVEL '.TXT'];
  PRESSFILENAME=['PRESSURE_' TIMELEVEL '.TXT'];
  PSIFILENAME=['PSI_DATA_' TIMELEVEL '.TXT'];
  ZETAFILENAME=['ZETA_DATA_' TIMELEVEL '.TXT'];

  U = LOAD (UFILENAME);
  [ M, N ] = SIZE ( U );
  NX = MAX ( U(1:M,1) ) - MIN ( U(1:M,1) ) + 1;
  NY = MAX ( U(1:M,2) ) - MIN ( U(1:M,2) ) + 1;
  INDEX = 0;
  FOR J = 1 : NY
    FOR I = 1: NX
      INDEX = INDEX + 1;
      X(I,J) = U(INDEX,1);
      Y(I,J) = U(INDEX,2);
      UOUT(I,J) = U(INDEX,3);
    END
  END

  V = LOAD (VFILENAME);
  [ M, N ] = SIZE ( V );
  NX = MAX ( V(1:M,1) ) - MIN ( V(1:M,1) ) + 1;
  NY = MAX ( V(1:M,2) ) - MIN ( V(1:M,2) ) + 1;
  INDEX = 0;
  FOR J = 1 : NY
    FOR I = 1: NX
      INDEX = INDEX + 1;
      X(I,J) = V(INDEX,1);
      Y(I,J) = V(INDEX,2);
```

```
      VOUT(I,J) = V(INDEX,3);
    END
  END

  P = LOAD (PRESSFILENAME);
  [ M, N ] = SIZE ( P );
  NX = MAX ( P(1:M,1) ) - MIN ( P(1:M,1) ) + 1;
  NY = MAX ( P(1:M,2) ) - MIN ( P(1:M,2) ) + 1;
  INDEX = 0;
  FOR J = 1 : NY
    FOR I = 1: NX
      INDEX = INDEX + 1;
      X(I,J) = P(INDEX,1);
      Y(I,J) = P(INDEX,2);
      POUT(I,J) = P(INDEX,3);
    END
  END

  PSI = LOAD (PSIFILENAME);
  [ M, N ] = SIZE ( U );
  NX = MAX ( PSI(1:M,1) ) - MIN ( PSI(1:M,1) ) + 1;
  NY = MAX ( PSI(1:M,2) ) - MIN ( PSI(1:M,2) ) + 1;
  INDEX = 0;
  FOR J = 1 : NY
    FOR I = 1: NX
      INDEX = INDEX + 1;
      X(I,J) = PSI(INDEX,1);
      Y(I,J) = PSI(INDEX,2);
      PSIOUT(I,J) = PSI(INDEX,3);
    END
  END

  ZETA = LOAD (ZETAFILENAME);
  [ M, N ] = SIZE ( U );
  NX = MAX ( PSI(1:M,1) ) - MIN ( PSI(1:M,1) ) + 1;
  NY = MAX ( PSI(1:M,2) ) - MIN ( PSI(1:M,2) ) + 1;
  INDEX = 0;
  FOR J = 1 : NY
    FOR I = 1: NX
      INDEX = INDEX + 1;
      X(I,J) = ZETA(INDEX,1);
      Y(I,J) = ZETA(INDEX,2);
      ZETAOUT(I,J) = ZETA(INDEX,3);
    END
  END
%
%   FOR J = 1 : NY
%     FOR I = 1: NX
%       XXX = (I-1/NX)*LX;
%       YYY = (J-1/NY)*LY;
%       X(I,J) = XXX;
%       Y(I,J) = YYY;
%     END
%   END
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    XC = 10;
    YC = 10;
```

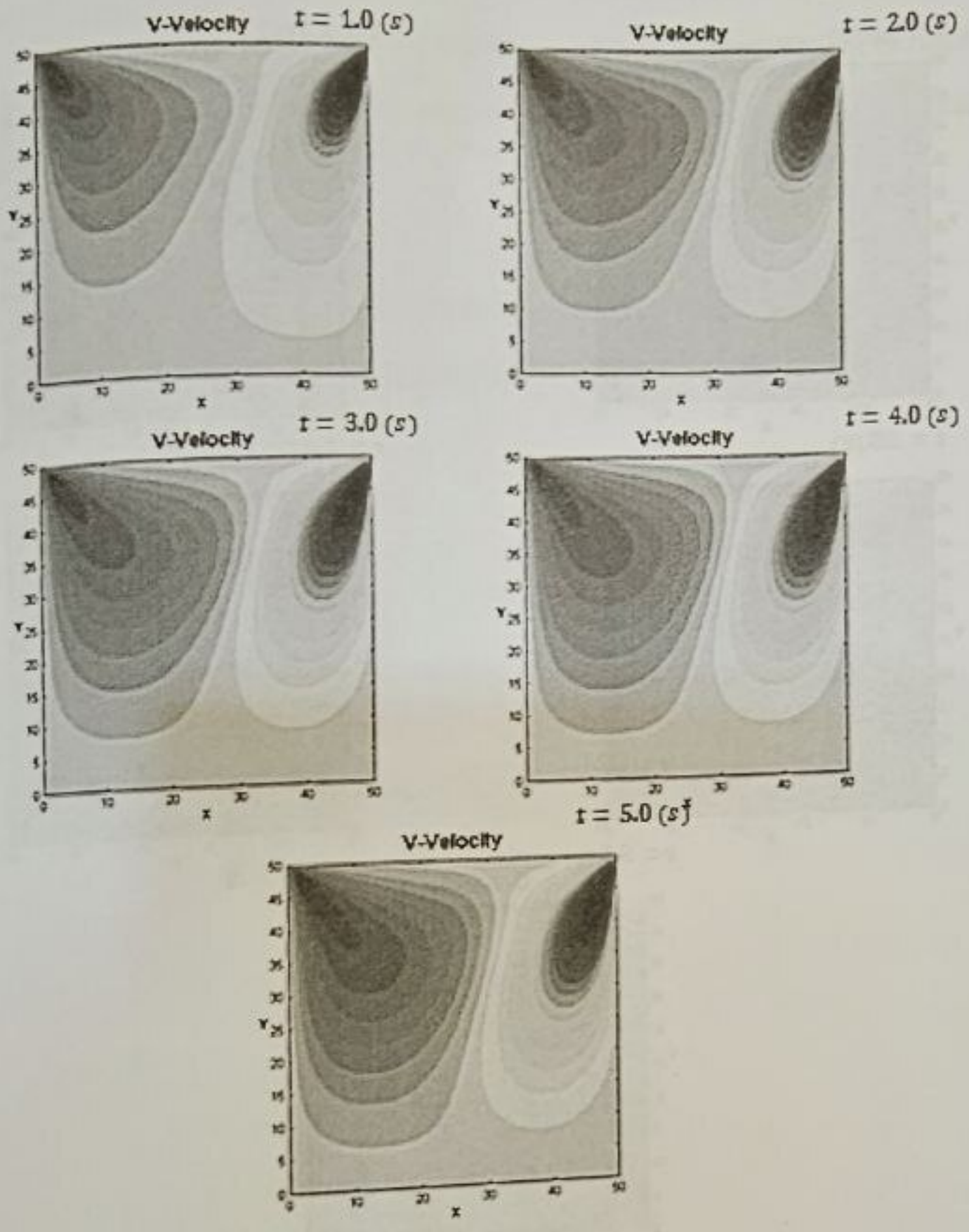
```
    RC = 3;
    TETA = 0:0.01:2*PI;
    CIRCLEX = RC*COS(TETA) + XC;
    CIRCLEY = RC*SIN(TETA) + YC;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    XREC1 =0;
    XREC2 =25;
    YREC1 =0;
    YREC2 =12;
    RECTANGX = [XREC1 XREC2 XREC2 XREC1 XREC1];
    RECTANGY = [YREC1 YREC1 YREC2 YREC2 YREC1];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  FIGURE(1)
  [ C1, H1 ] = CONTOURF ( X, Y, UOUT, NO_LEVELS, 'LINESTYLE','NONE' );
  COLORMAP ( JET );
  XLABEL ( 'X', 'FONTNAME', 'HELVETICA', 'FONTWEIGHT',
'BOLD','FONTSIZE', 10 );
  YLABEL ( 'Y', 'FONTNAME', 'HELVETICA', 'FONTWEIGHT',
'BOLD','FONTSIZE', 10, 'ROTATION', 0 );
  TITLE ( 'U-VELOCITY', 'FONTNAME', 'HELVETICA', 'FONTWEIGHT','BOLD',
'FONTSIZE', 16 );
  HOLD ON
  SWITCH (PROBLEM)
      CASE 'CYLINDER'
          FILL(CIRCLEX,CIRCLEY,'W')
      CASE 'STEP'
          FILL(RECTANGX,RECTANGY,'W')

  END
  DASPECT([1 1 1])


  FIGURE(2)
  [ C2, H2 ] = CONTOURF ( X, Y, VOUT, NO_LEVELS, 'LINESTYLE','NONE' );
  COLORMAP ( JET );
  XLABEL ( 'X', 'FONTNAME', 'HELVETICA', 'FONTWEIGHT',
'BOLD','FONTSIZE', 10 );
  YLABEL ( 'Y', 'FONTNAME', 'HELVETICA', 'FONTWEIGHT',
'BOLD','FONTSIZE', 10, 'ROTATION', 0 );
  TITLE ( 'V-VELOCITY', 'FONTNAME', 'HELVETICA', 'FONTWEIGHT','BOLD',
'FONTSIZE', 16 );
  HOLD ON
  SWITCH (PROBLEM)
      CASE 'CYLINDER'
          FILL(CIRCLEX,CIRCLEY,'W')
      CASE 'STEP'
          FILL(RECTANGX,RECTANGY,'W')
  END
  DASPECT([1 1 1])

  FIGURE(3)
  [ C3, H3 ] = CONTOURF ( X, Y, POUT, NO_LEVELS, 'LINESTYLE','NONE' );
  COLORMAP ( JET );
  XLABEL ( 'X', 'FONTNAME', 'HELVETICA', 'FONTWEIGHT',
'BOLD','FONTSIZE', 10 );
  YLABEL ( 'Y', 'FONTNAME', 'HELVETICA', 'FONTWEIGHT',
'BOLD','FONTSIZE', 10, 'ROTATION', 0 );
```

```
 TITLE ( 'PRESSURE FIELD', 'FONTNAME', 'HELVETICA',
'FONTWEIGHT','BOLD', 'FONTSIZE', 16 );
 HOLD ON
 SWITCH (PROBLEM)
     CASE 'CYLINDER'
         FILL(CIRCLEX,CIRCLEY,'W')
     CASE 'STEP'
         FILL(RECTANGX,RECTANGY,'W')
 END
 DASPECT([1 1 1])

 FIGURE(4)
 [ C4, H4 ] = CONTOURF ( X, Y, PSIOUT, NO_LEVELS );
 COLORMAP ( JET );
 XLABEL ( 'X', 'FONTNAME', 'HELVETICA', 'FONTWEIGHT',
'BOLD','FONTSIZE', 10 );
 YLABEL ( 'Y', 'FONTNAME', 'HELVETICA', 'FONTWEIGHT',
'BOLD','FONTSIZE', 10, 'ROTATION', 0 );
 TITLE ( 'STREAM FUNCTION', 'FONTNAME', 'HELVETICA',
'FONTWEIGHT','BOLD', 'FONTSIZE', 16 );
 HOLD ON
 SWITCH (PROBLEM)
     CASE 'CYLINDER'
         FILL(CIRCLEX,CIRCLEY,'W')
     CASE 'STEP'
         FILL(RECTANGX,RECTANGY,'W')
 END
 DASPECT([1 1 1])

 FIGURE(6)
 [ C4, H4 ] = CONTOURF ( X, Y, ZETAOUT, NO_LEVELS, 'LINESTYLE','NONE'
);
 COLORMAP ( JET );
 XLABEL ( 'X', 'FONTNAME', 'HELVETICA', 'FONTWEIGHT',
'BOLD','FONTSIZE', 10 );
 YLABEL ( 'Y', 'FONTNAME', 'HELVETICA', 'FONTWEIGHT',
'BOLD','FONTSIZE', 10, 'ROTATION', 0 );
 TITLE ( 'VORTICITY', 'FONTNAME', 'HELVETICA', 'FONTWEIGHT','BOLD',
'FONTSIZE', 16 );
 HOLD ON
 SWITCH (PROBLEM)
     CASE 'CYLINDER'
         FILL(CIRCLEX,CIRCLEY,'W')
     CASE 'STEP'
         FILL(RECTANGX,RECTANGY,'W')
 END
 DASPECT([1 1 1])

 SWITCH (VECTORPLOT)
     CASE 'Y'
         FIGURE(5)
         FOR I=1:10
             YI = 0:1:NY;
             XI = (I/10)*(NX)*ONES(SIZE(YI));
             UI=GRIDDATA(X,Y,UOUT,XI,YI);
             VI=GRIDDATA(X,Y,VOUT,XI,YI);
             QUIVER ( XI, YI, UI, VI, 2 );
             COLORMAP JET;
```

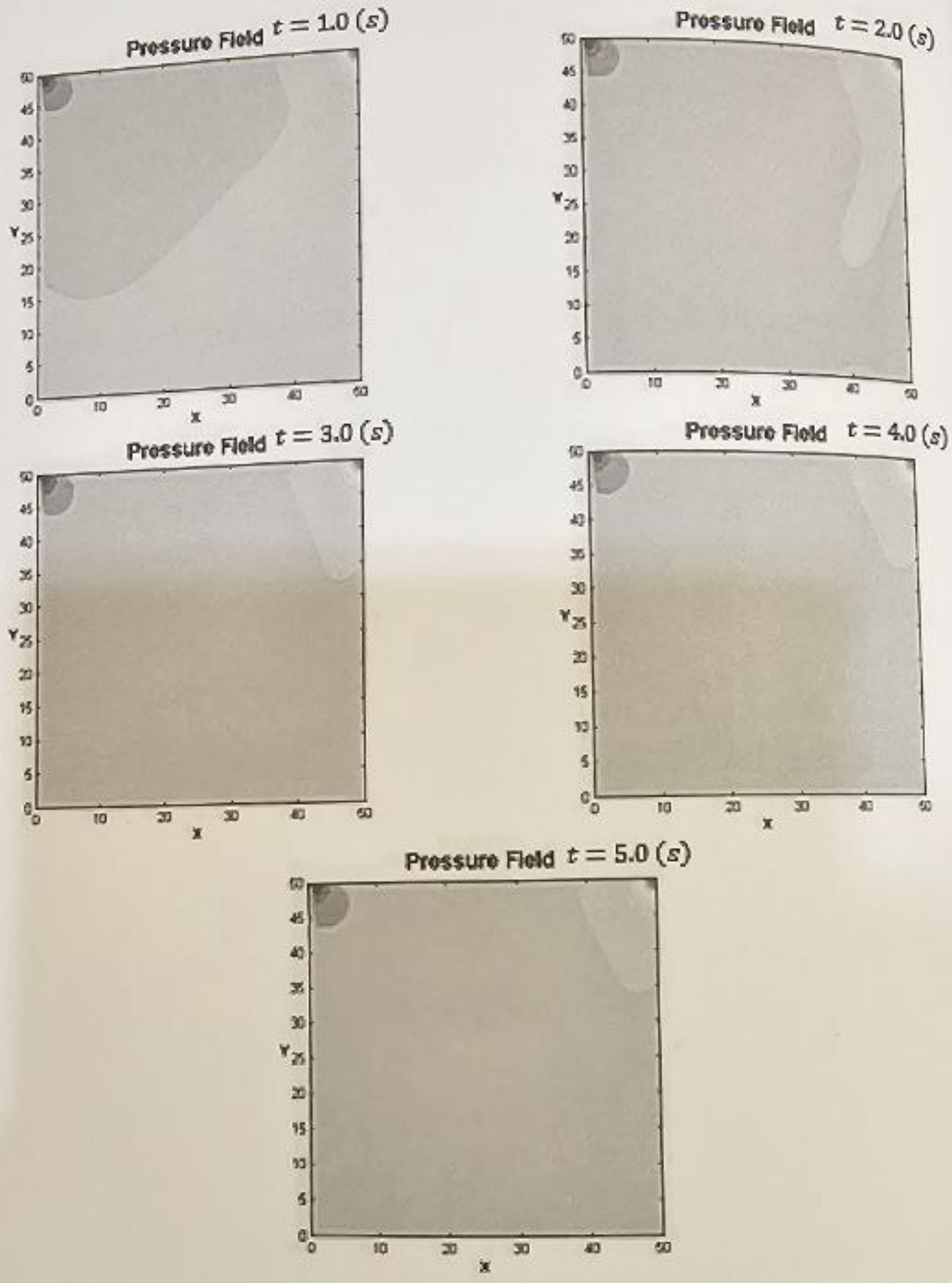
```
            XLABEL ( 'X', 'FONTNAME', 'HELVETICA', 'FONTWEIGHT',
'BOLD','FONTSIZE', 10 );
            YLABEL ( 'Y', 'FONTNAME', 'HELVETICA', 'FONTWEIGHT',
'BOLD','FONTSIZE', 10, 'ROTATION', 0 );
            TITLE ( 'VELOCITY VECTORS', 'FONTNAME', 'HELVETICA',
'FONTWEIGHT','BOLD', 'FONTSIZE', 16 );
            HOLD ON
        END
        SWITCH (PROBLEM)
            CASE 'CYLINDER'
                FILL(CIRCLEX,CIRCLEY,'W')
            CASE 'STEP'
                FILL(RECTANGX,RECTANGY,'W')
            CASE 'CAVITY'
                RETURN
        END
        DASPECT([1 1 1])
        XLIM([0 NX])
        YLIM([0 NY])
END

FPRINTF ( 1, '\N' );
FPRINTF ( 1, 'PGRG2DNAVISTO_CONTOUR:\N' );
FPRINTF ( 1, '  END OF EXECUTION.\N' );
```



شکل (۱۲-۲۵): کانتورهای سرعت u در زمان‌های ۱، ۲، ۳، ۴ و ۵ ثانیه $(Re = 100)$

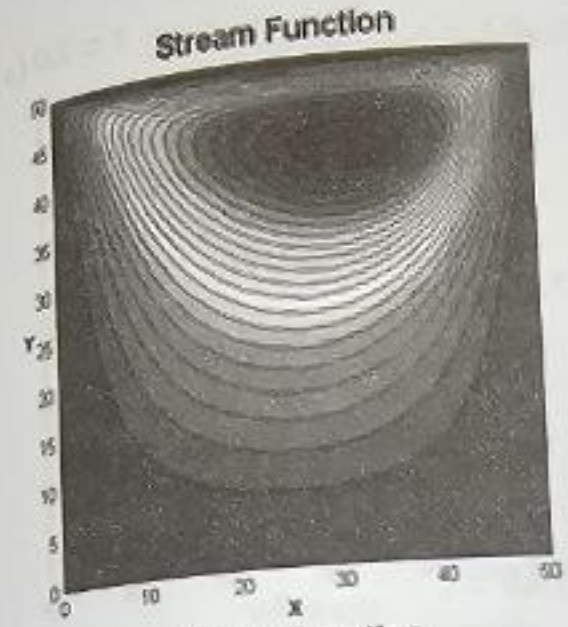


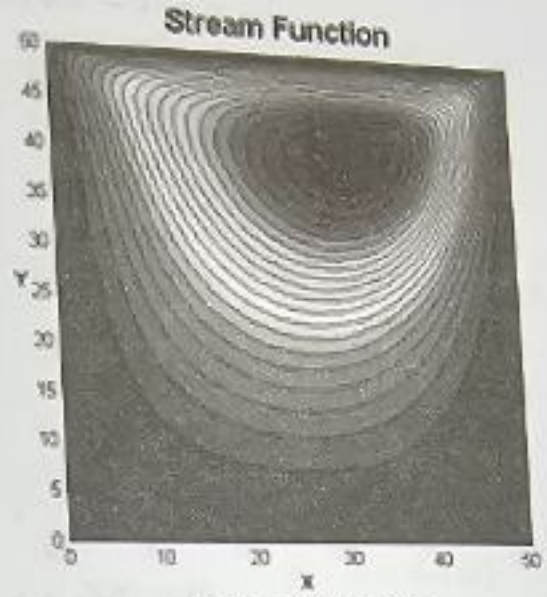
شکل (۱۲-۲۶): کانتورهای سرعت v در زمان‌های ۱، ۲، ۳، ۴ و ۵ ثانیه $(Re = 100)$

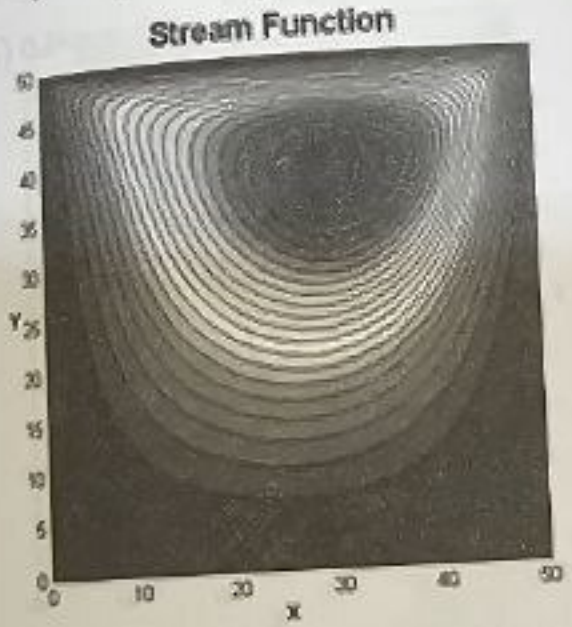


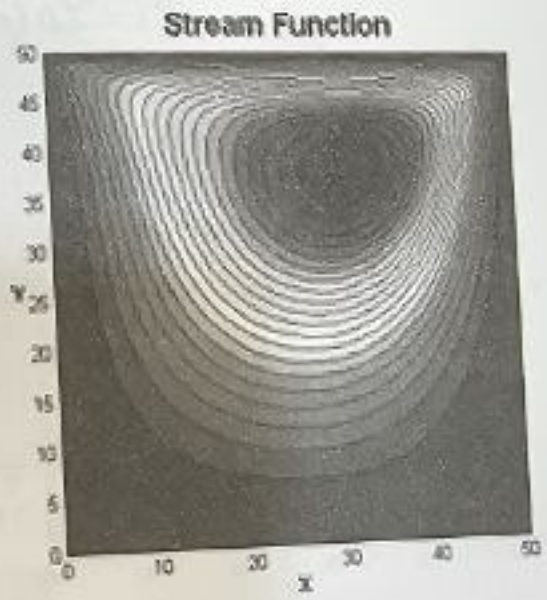
شکل (۱۲-۲۷): کانتورهای فشار p در زمان‌های ۱، ۲، ۳، ۴ و ۵ ثانیه $(Re = 100)$

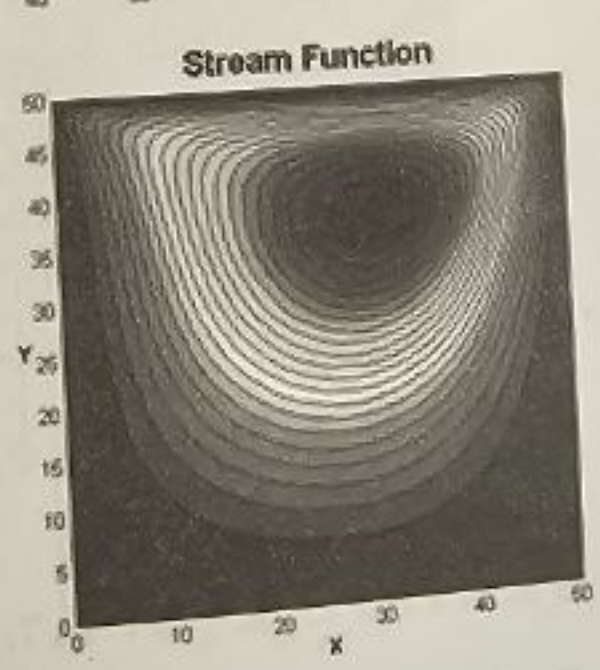
$t = 1.0\ (s)$ $t = 2.0\ (s)$



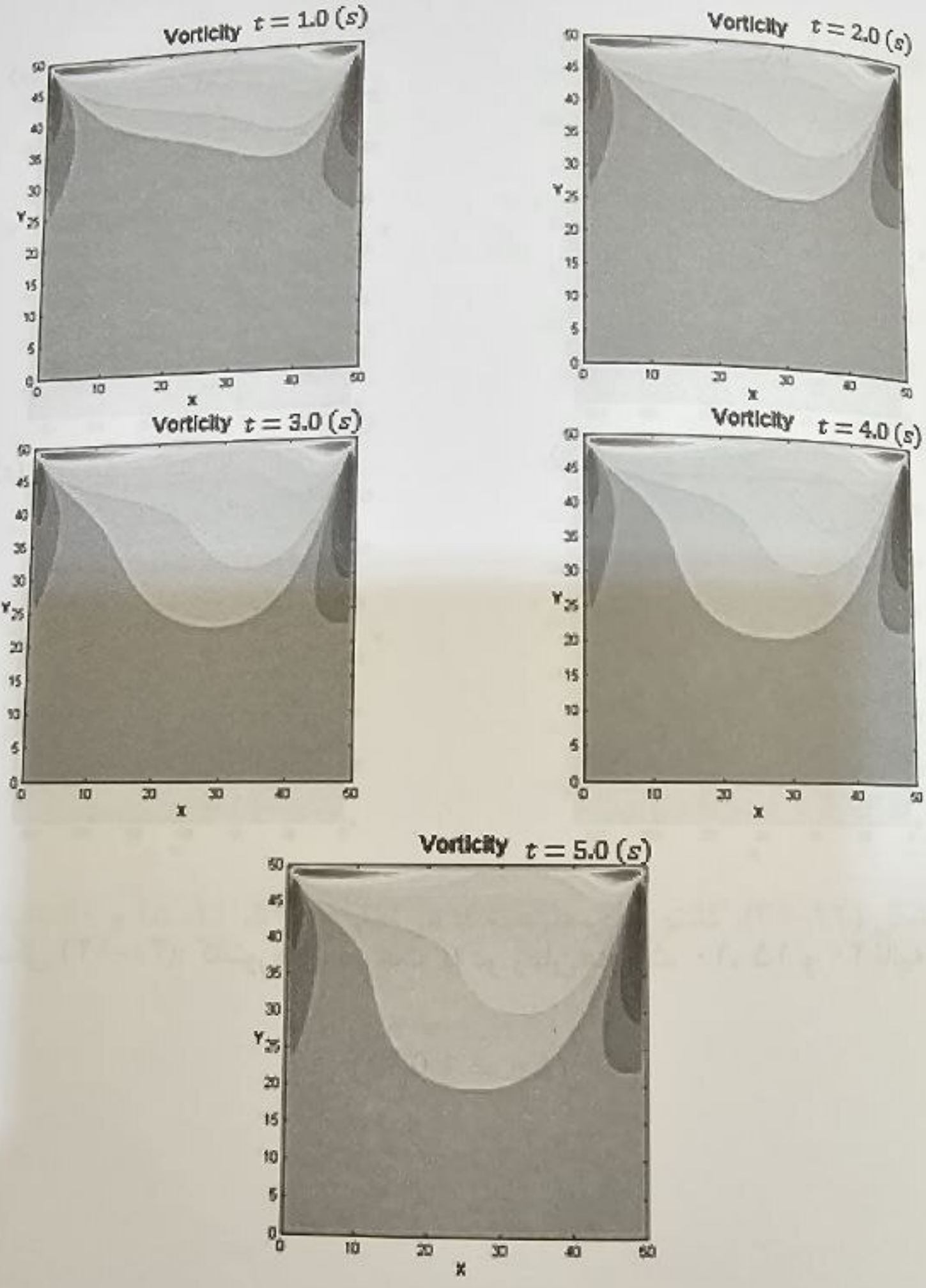






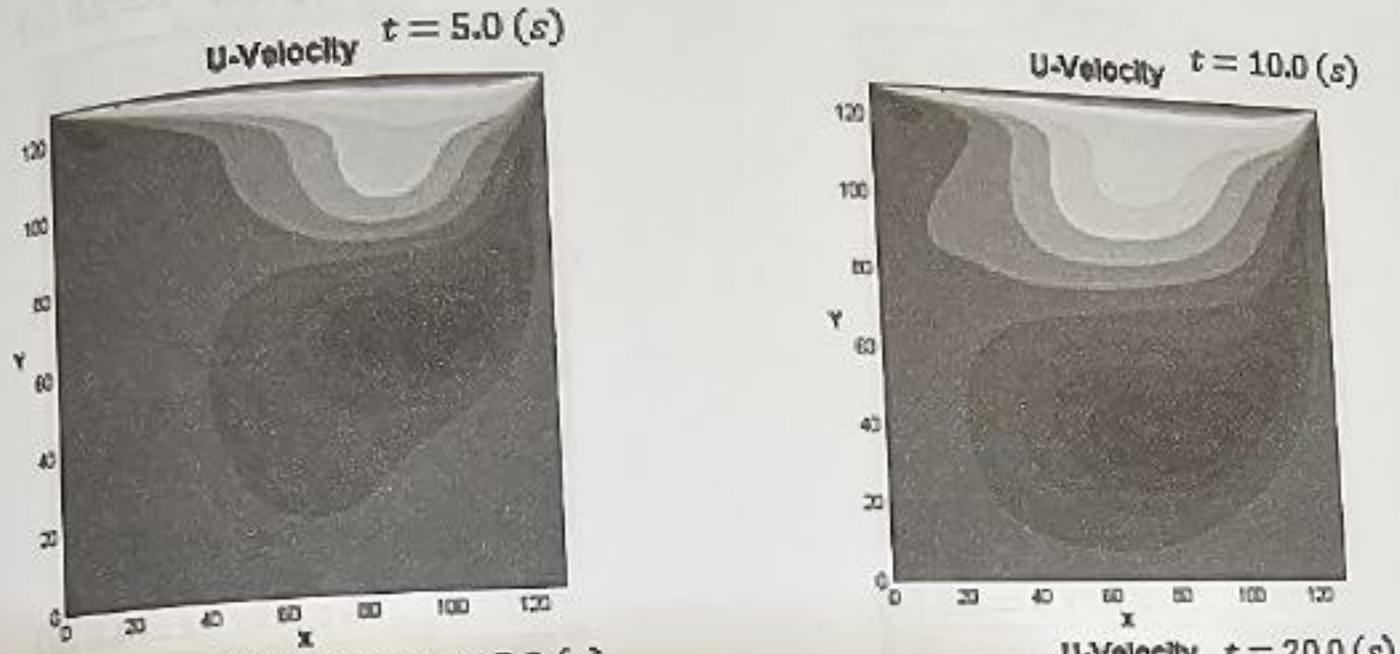


شکل (۱۲-۲۸): خطوط جریان ψ در زمان‌های ۱، ۲، ۳، ۴ و ۵ ثانیه $(Re = 100)$



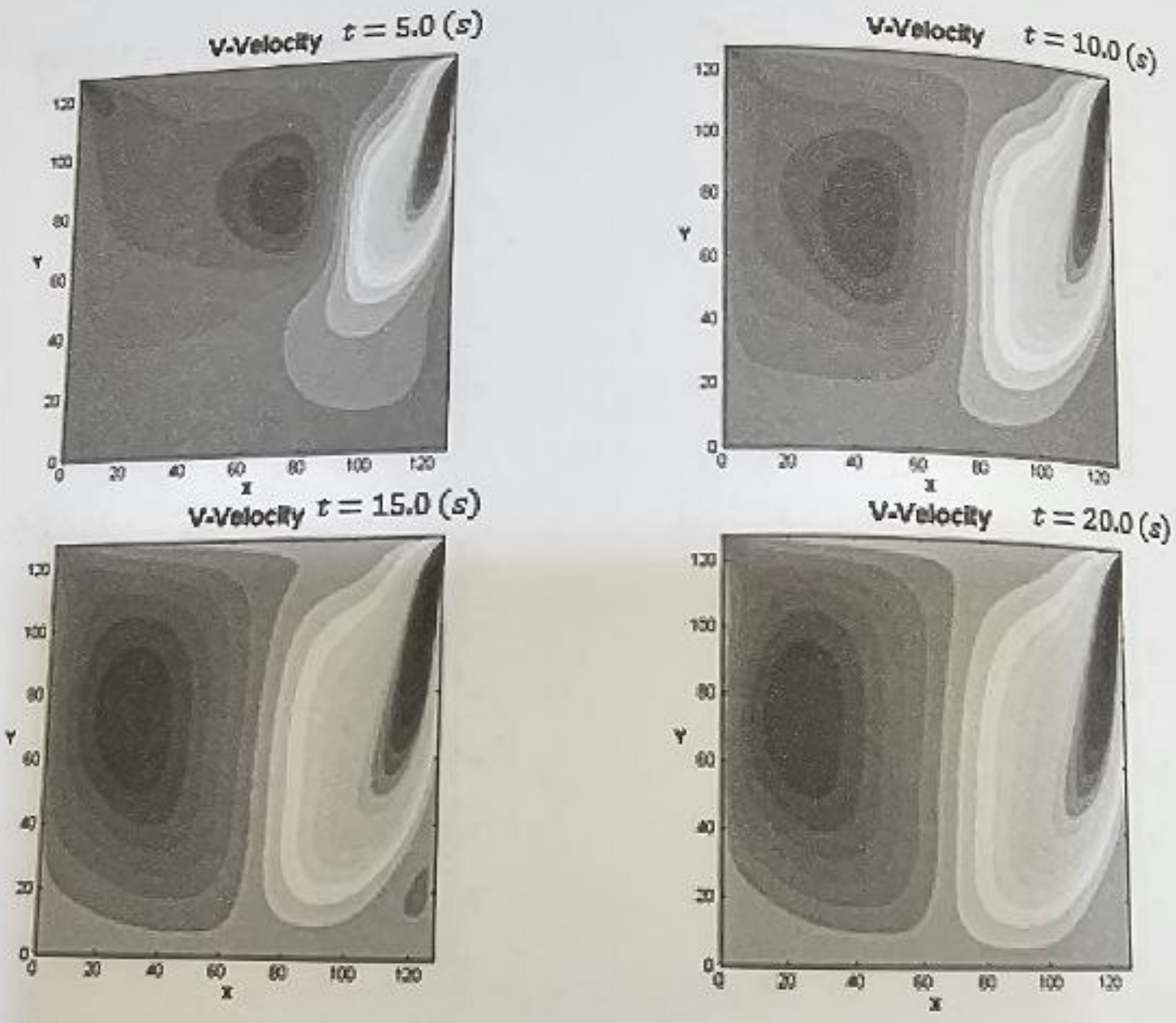
شکل (۱۲-۲۹): کانتورهای ورتیسیته ζ در زمان‌های ۱، ۲، ۳، ۴ و ۵ ثانیه

$$(Re = 100)$$



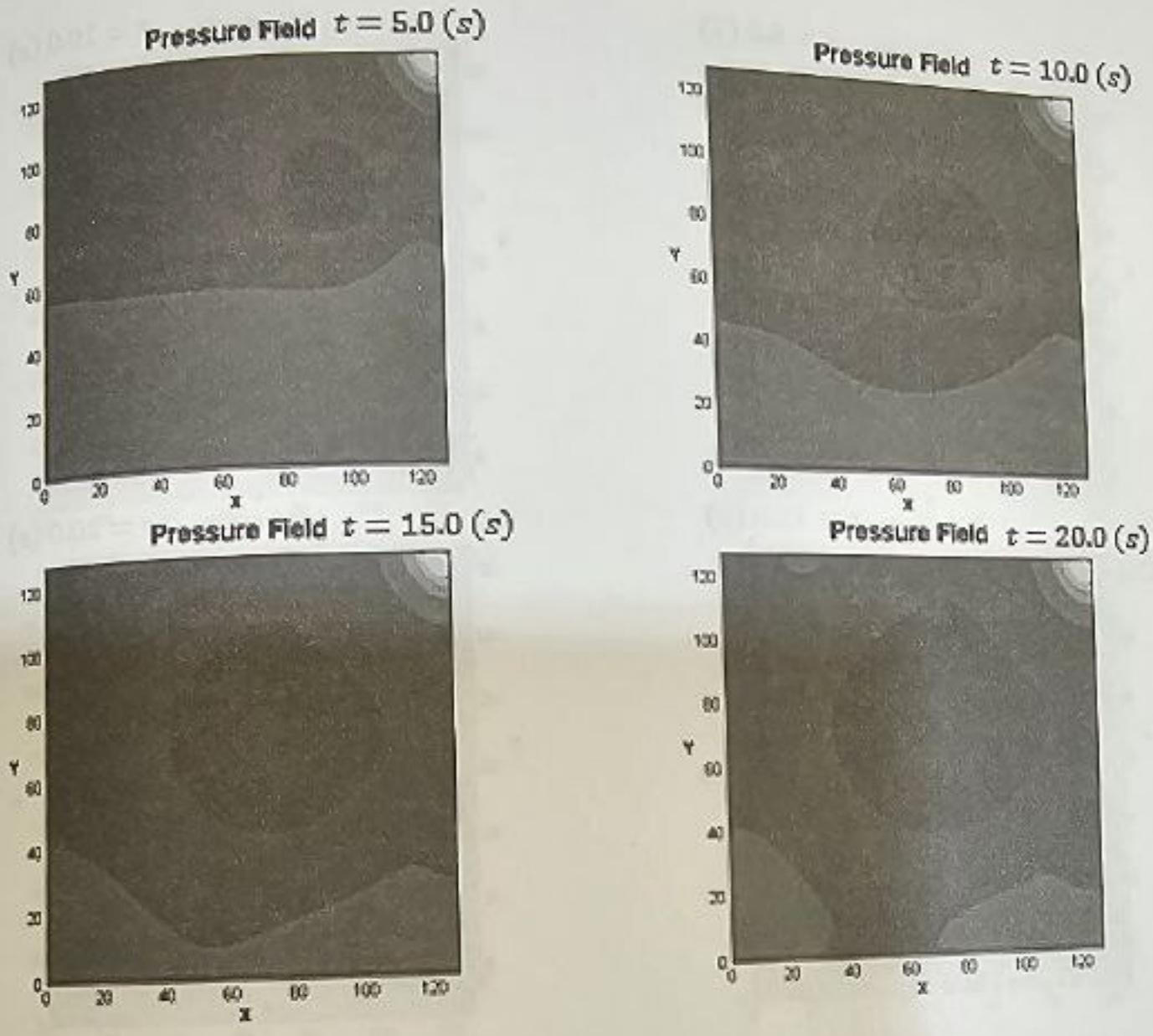
شکل (۱۲-۳۰): کانتورهای سرعت u در زمان‌های ۵، ۱۰، ۱۵ و ۲۰ ثانیه

$(Re = 1{,}000)$



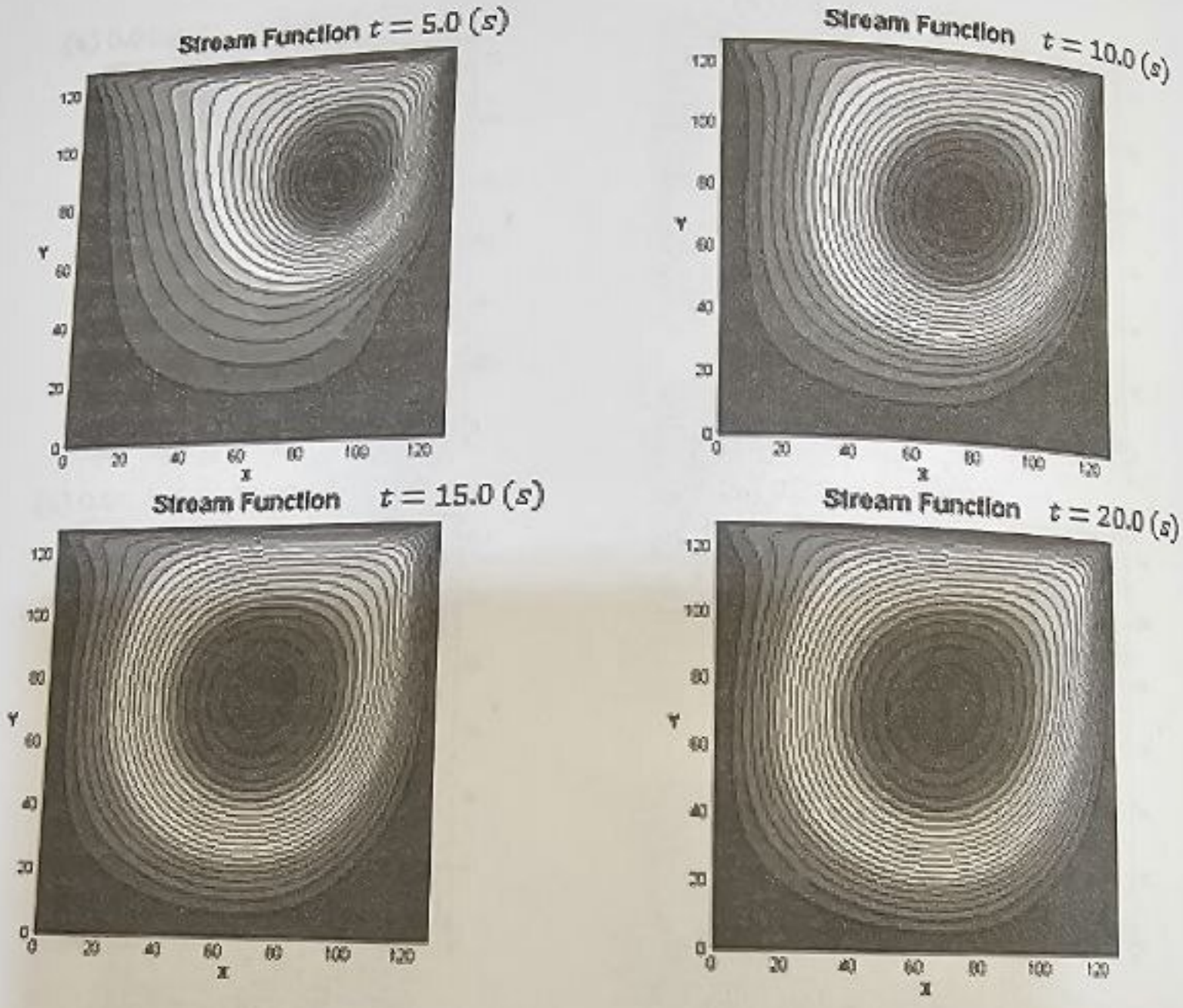
شکل (۱۲-۳۱): کانتورهای سرعت v در زمان‌های ۵، ۱۰، ۱۵ و ۲۰ ثانیه

$$(Re = 1{,}000)$$



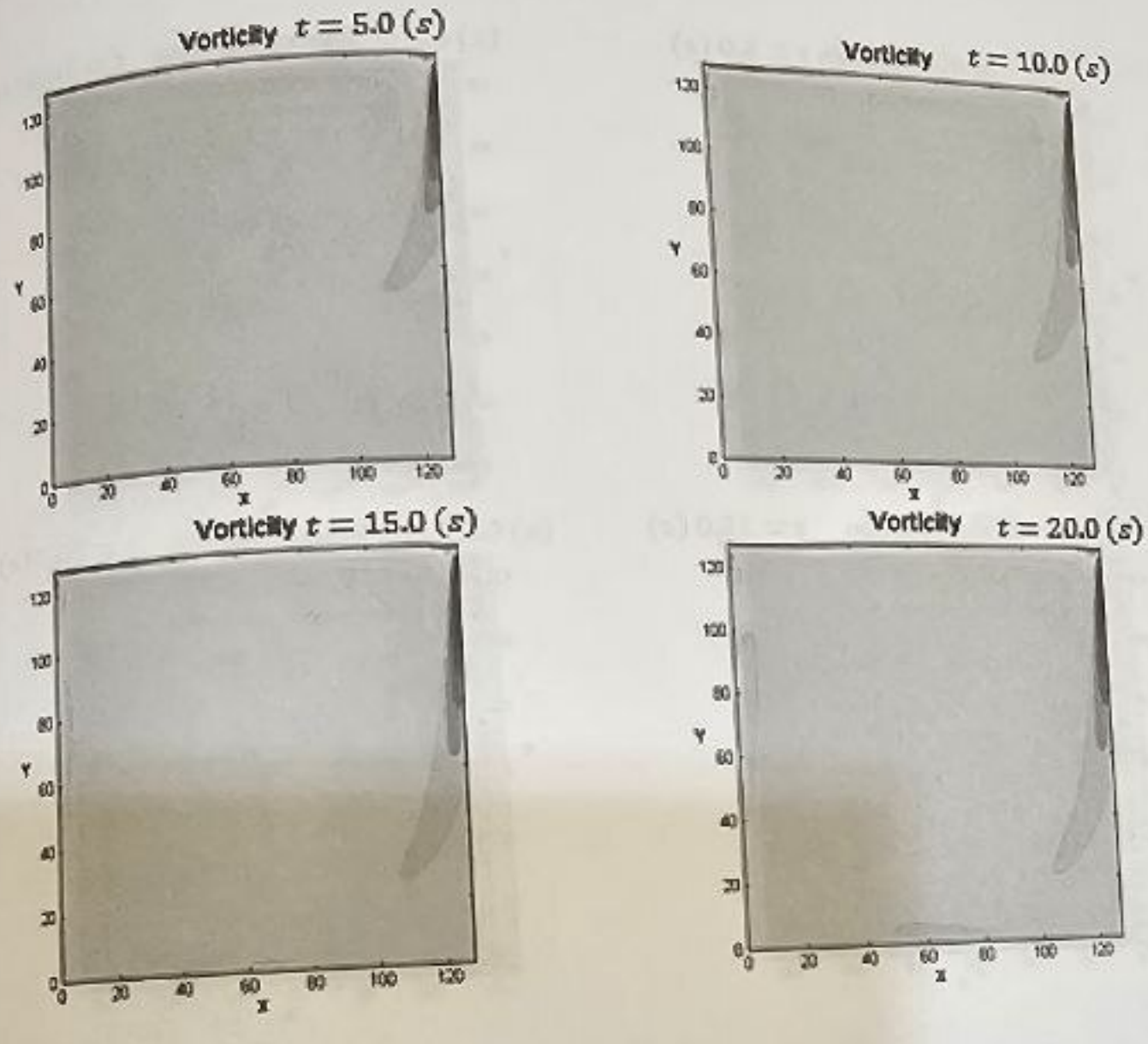
شکل (۱۲-۳۲): کانتورهای فشار p در زمان‌های ۵، ۱۰، ۱۵ و ۲۰ ثانیه

$$(Re = 1{,}000)$$

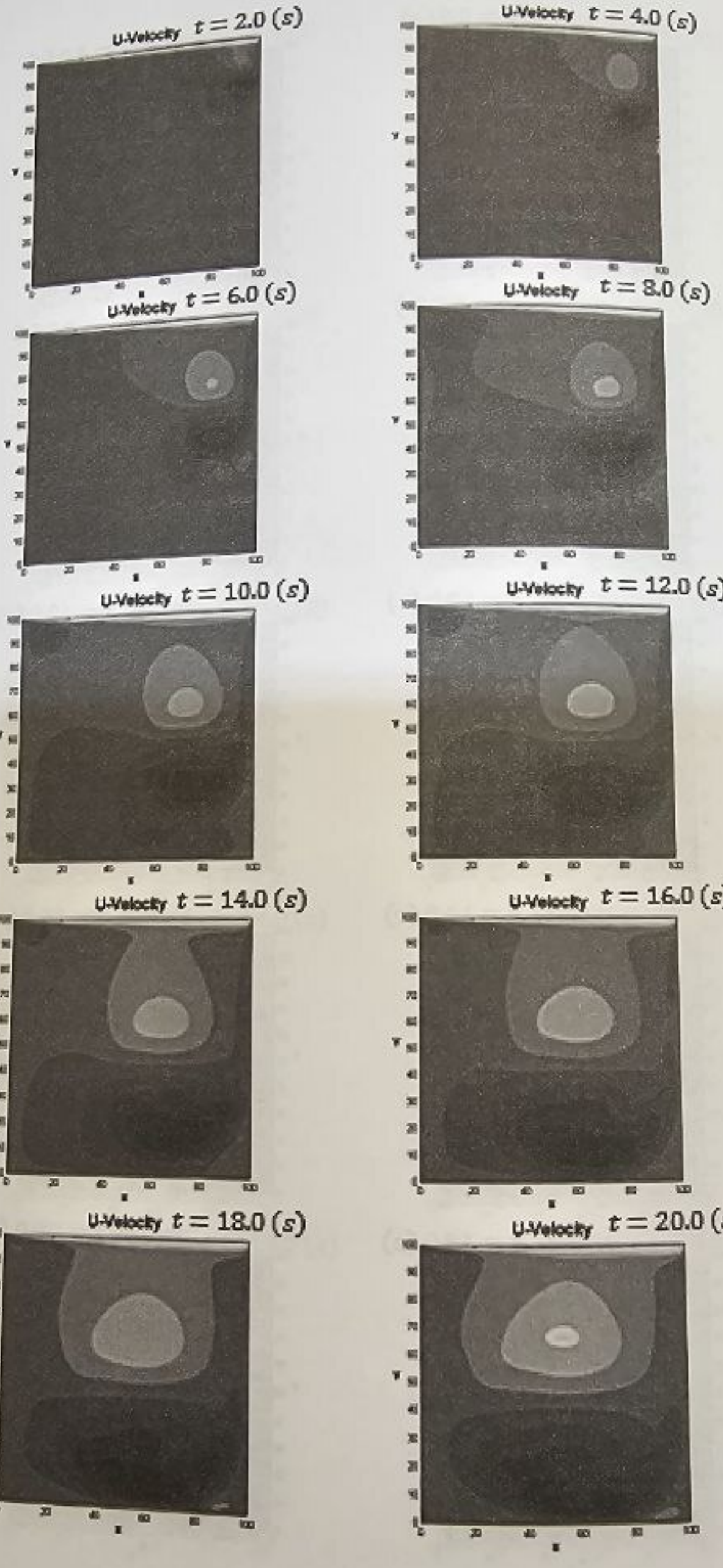


شکل (۱۲-۳۳): خطوط جریان ψ در زمان‌های ۵، ۱۰، ۱۵ و ۲۰ ثانیه

$$(Re = 1{,}000)$$

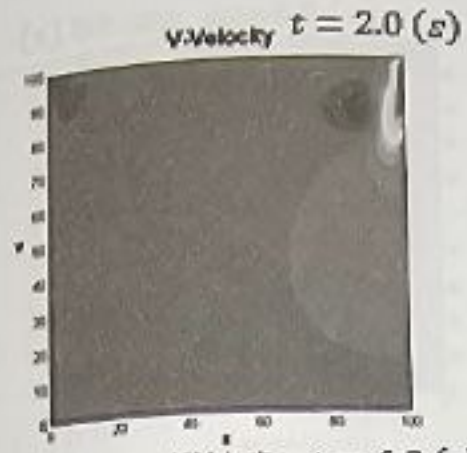


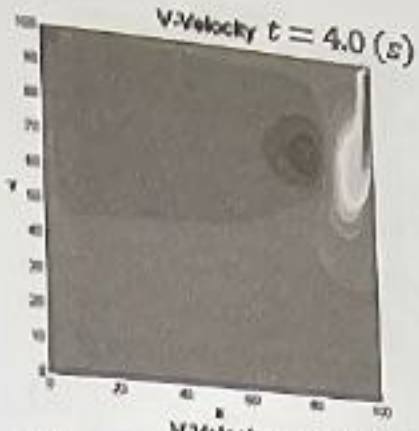
شکل (۱۲-۳۴): کانتورهای ورتیسیتی ζ در زمان‌های ۵، ۱۰، ۱۵ و ۲۰ ثانیه $(Re = 1{,}000)$

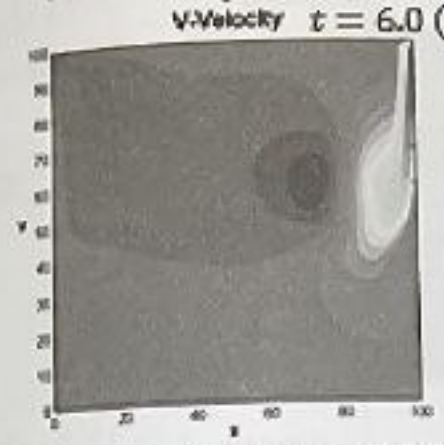


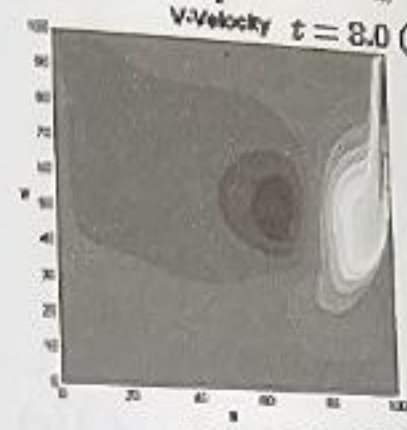
شکل (۱۲-۳۵): کانتورهای سرعت u در زمان‌های ۲ الی ۲۰ ثانیه $(Re = 10{,}000)$

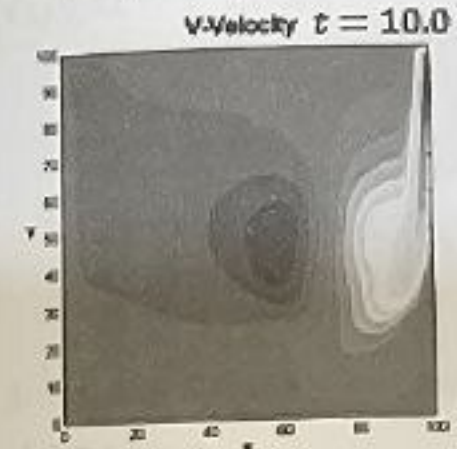
۲۰ ثانیه

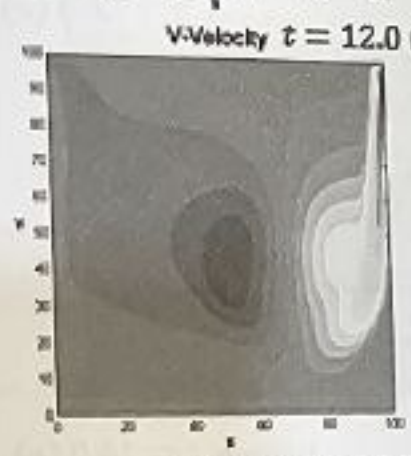


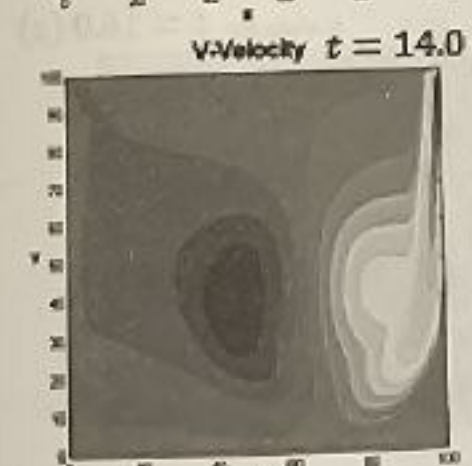


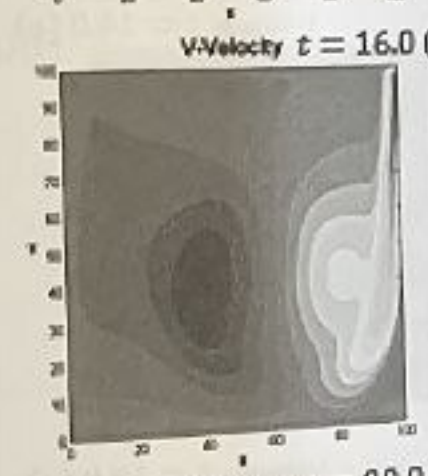


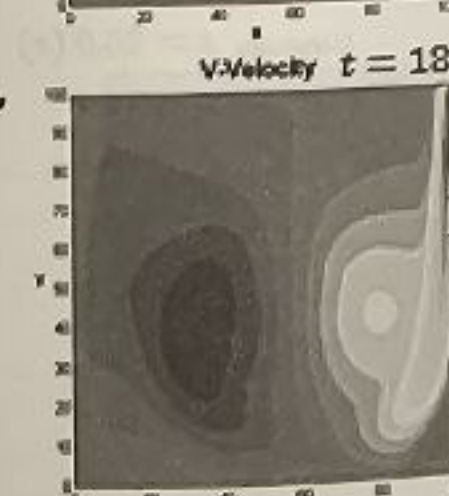


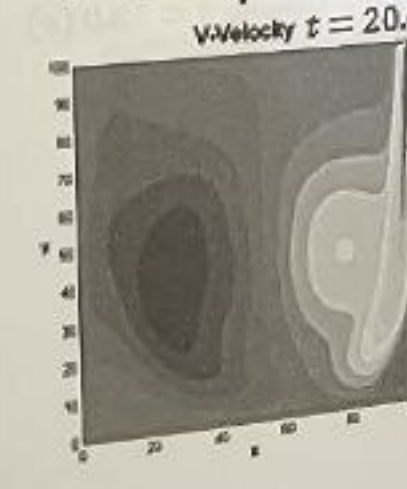




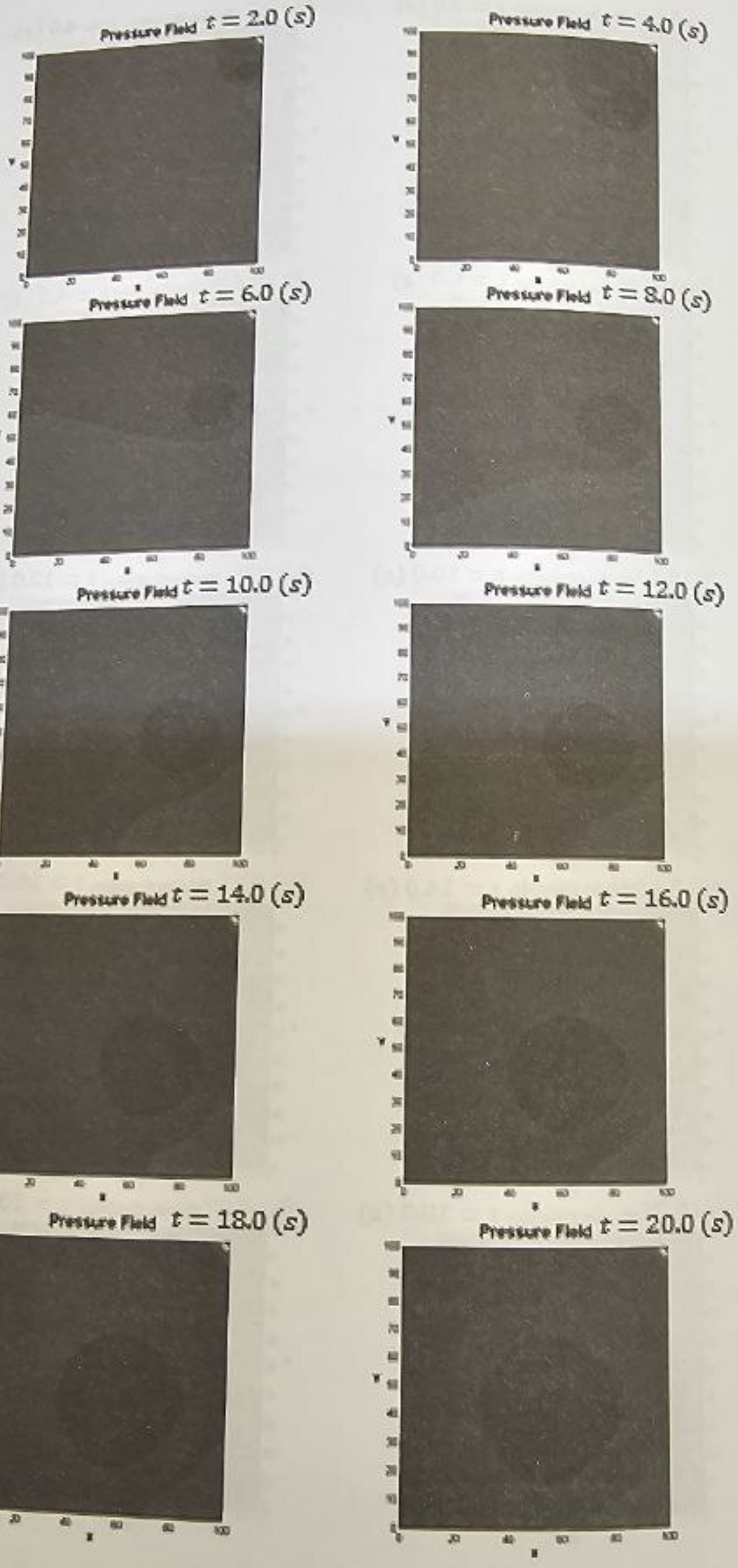




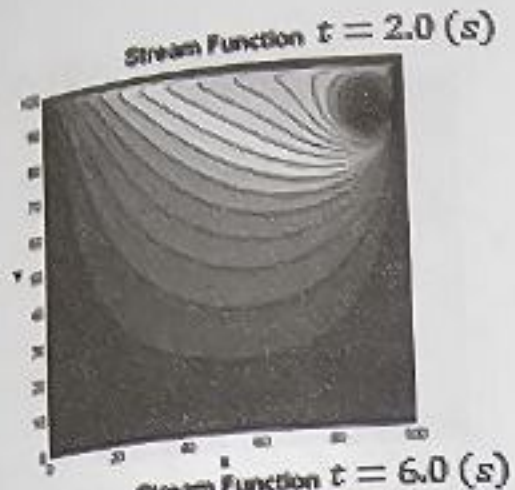


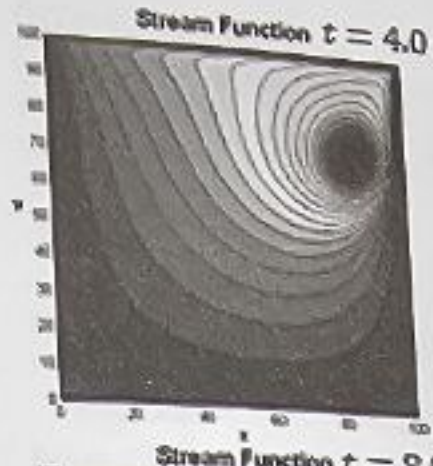


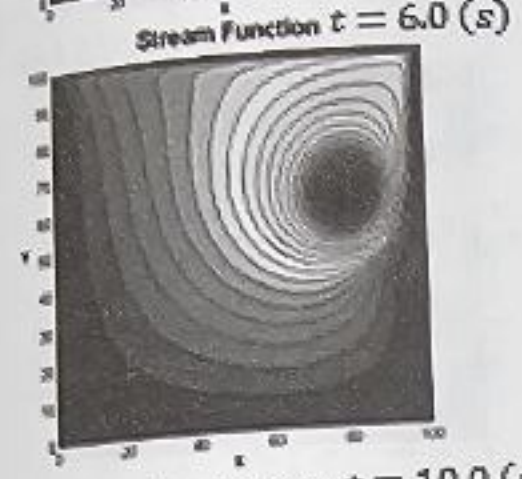
شکل (۱۲-۳۶): کانتورهای سرعت v در زمان‌های ۲ الی ۲۰ ثانیه $(Re = 10{,}000)$

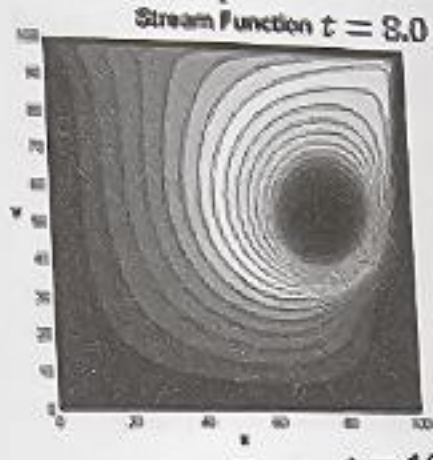


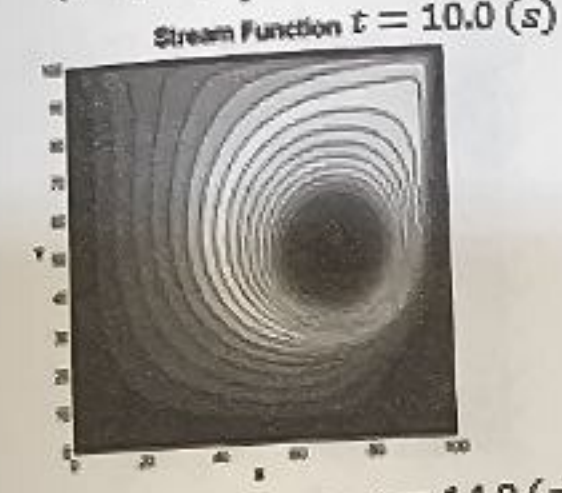
شکل (۱۲-۳۷): کانتورهای فشار p در زمان‌های ۲ الی ۲۰ ثانیه $(Re = 10{,}000)$

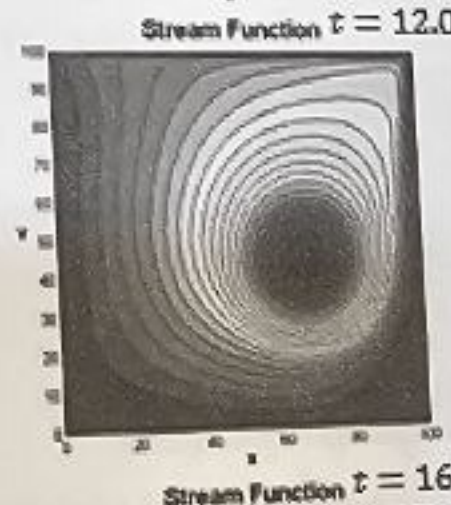


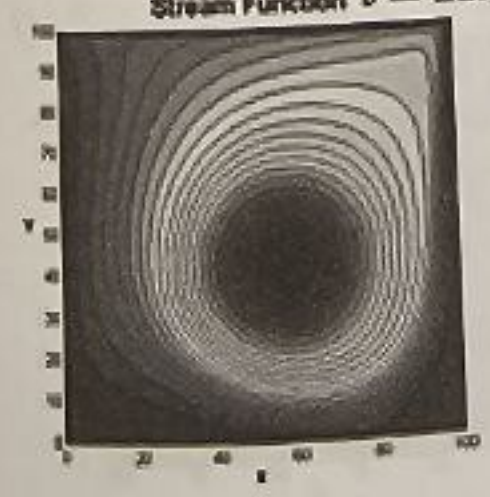


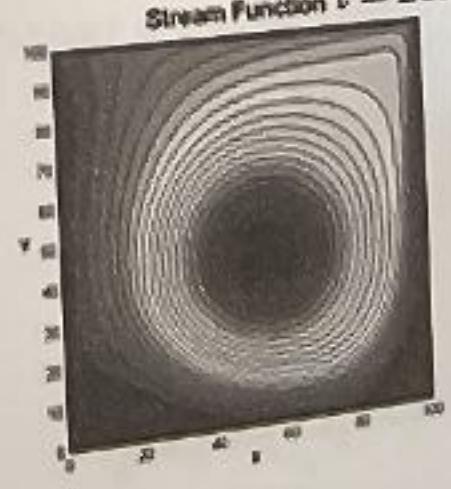




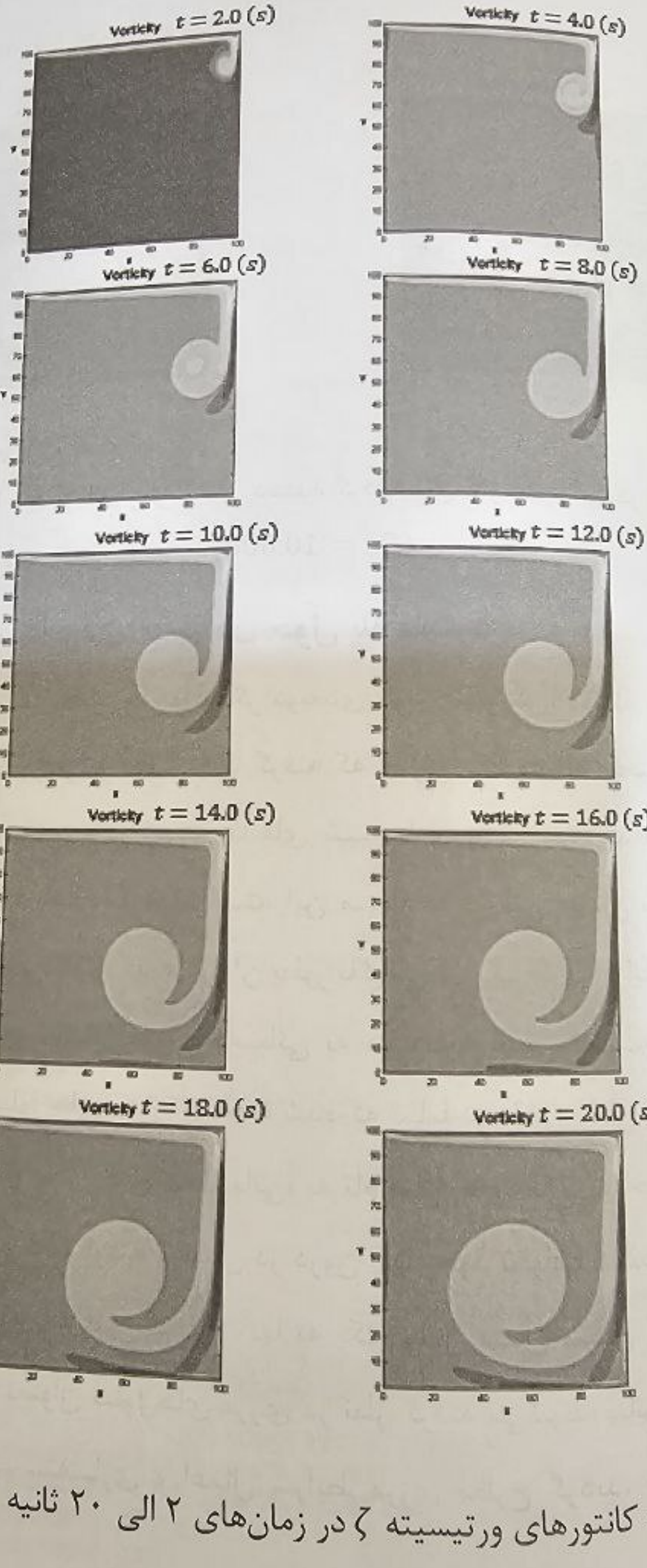




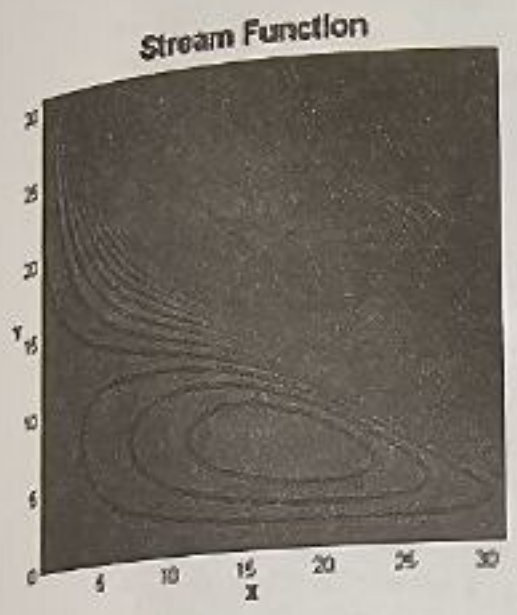


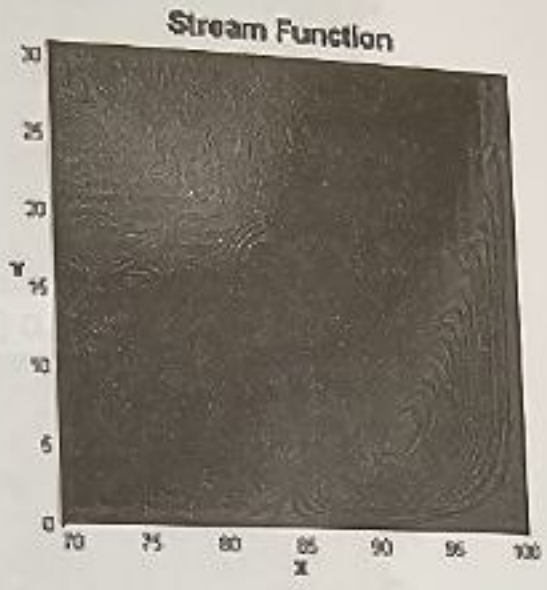


شکل (۱۲-۳۸): خطوط جریان ψ در زمان‌های ۲ الی ۲۰ ثانیه $(Re = 10{,}000)$



شکل (۱۲-۳۹): کانتورهای ورتیسیته ζ در زمان‌های ۲ الی ۲۰ ثانیه $(Re = 10{,}000)$





شکل (۱۲-۴۰): خطوط جریان نشان دهندهٔ گردابه‌های ثانویهٔ چپگرد در نمایی نزدیک $(Re = 10{,}000)$

۱۲-۴-۴ مسئلهٔ کاربردی ۶: جریان حول پلهٔ عقبگرد

بعنوان دوّمین مسئلهٔ قابل حل با حلگر دوبعدی ناویر-استوکس، مسئلهٔ جریان بر روی پله عقبگرد مورد بررسی و تحلیل قرار گرفته که بعنوان یک مسئله صحت‌سنجی بسیار شناخته شده و پرکاربرد در زمینهٔ کدهای شبیه‌سازی رفتار سیالات، مطرح می‌شود. همانطورکه قبلاً هم اشاره گردیده است، این مسئله به بررسی جریان سیال در داخل یک کانال مستقیم پرداخته که عرض آن بطور ناگهانی در یک طرف افزایش می‌یابد. در کد عددی حاضر، سلول‌های محاسباتی به دو دسته اصلی تقسیم‌بندی می‌شوند. دسته اول به نام سلول‌های سیال شناخته شده که تماماً در داخل سیال واقع شده‌اند. در مقابل، دسته دوم از سلول‌های محاسباتی، به نام سلول‌های مانع شناخته می‌شوند که تماماً در داخل مانع قرار گرفته و سیال در درون آنها وجود نخواهد داشت. در این میان، سلول‌های مانعی که حداقل یک لبه آنها به یک سلول سیال متصل گردیده و با آن همسایه می‌باشند، بعنوان سلول‌های مرزی در نظر گرفته می‌شوند. بنابر آنچه در بخش قبل و در نحوهٔ گسسته‌سازی و اعمال شرایط مرزی مطرح گردید، در اطراف دامنهٔ