Chapter 1

Conventions of MAT*fluids*

1.1 Fundamental Fluid Dynamics Variables

In fluid dynamics, the pressure $p(\mathbf{x}, t)$ and velocity $\mathbf{u}(\mathbf{x}, t)$ fields of a flow are its most fundamental properties. We learn plenty about the physics of a flow by analyzing these two properties in numerous ways. The coordinate system $(\mathbf{x}, t)$ on which the pressure and velocity fields are described is also critical since this not only affects how we visualize and interpret the flow but also how we analyze it. In this section, we consider the most basic and common of coordinate systems, namely, the rectangular (or Cartesian) system $\mathbf{x} = (x, y, z)$.

MAT*fluids* uses the variable `coord` to express the coordinate system. The `coord` variable is a structure array whose fields represent the different coordinate directions. In rectangular coordinates, the fields in the `coord` variable are `t`, `x`, `y` and `z`. Each of these field variables are expressed as column vector arrays whose elements increase monotonically with the array index. MAT*fluids* does not, however, impose any restriction on the spacing or discretization used in the coordinate directions. We show an example `coord` variable in the code snippet below.

```
>> coord.t = (0:0.1:10).';
>> coord.x = (0:0.02:1).';
>> coord.y = [0:0.02:2 2.1:0.1:3].';
```

The pressure and velocity fields are respectively represented as `p` and `vel` in MAT*fluids*. The velocity field variable `vel` is a structure array containing the fields `u`, `v` and `w`, which represent the velocity components in the `x`, `y` and `z` coordinate directions contained in the variable `coord`, respectively. The pressure

and velocity components may be one- to four-dimensional arrays depending on the dimension of the flow data, namely, the number of spatial coordinates being considered and whether the data evolves in time or not.

For Eulerian flow variables given by an n-dimensional array ($n \in \{1, 2, 3, 4\}$), MAT*fluids* uses the convention that the array indices express the spatial coordinates in order (i.e., x, y, z) followed by the time coordinate (t). For example, given a two-dimensional time-dependent flow defined in the (x, y) plane, the pressure value `p(5,2,31)` corresponds to that at the point `coord.x(5)`, `coord.y(2)` and `coord.t(31)`. When plotting a two-dimensional array defined in the (x, y) plane, we note here that MATLAB instead prefers to have the first array dimension correspond to the y coordinate and the second to x. This is the case for all such plotting functions (e.g., `contour`, `contourf`, `pcolor`, `quiver`). Therefore, rather than having the user remember to permute the first and second dimensions when plotting, we offer a set of plotting functions that does this automatically; see Ch. 3.

Chapter 2

Flow Toolbox

2.1 Uniform Flow

The most elementary flow that MAT*fluids* incorporates is the case of a uniform flow, namely, one where the magnitude and direction of velocity is constant everywhere. In rectangular coordinates, the potential (ϕ) and stream (ψ) functions for a uniform flow are given by

$$\phi(x,y) = U_\infty (x\cos(\alpha) + y\sin(\alpha)), \tag{2.1a}$$
$$\psi(x,y) = U_\infty (y\cos(\alpha) - x\sin(\alpha)), \tag{2.1b}$$

where U_∞ denotes the constant velocity magnitude of the flow and α the angle of the flow measured counterclockwise from the positive x axis. The corresponding velocity components are therefore given by

$$u = \frac{\partial \phi}{\partial x} = \frac{\partial \psi}{\partial y} = U_\infty \cos(\alpha), \tag{2.2a}$$
$$v = \frac{\partial \phi}{\partial y} = -\frac{\partial \psi}{\partial x} = U_\infty \sin(\alpha). \tag{2.2b}$$

Given that the magnitude and direction of velocity is constant everywhere in the flow, the velocity gradient is identically zero ($\nabla \mathbf{u} = \mathbf{0}$).

2.2 The Irrotational (or Potential) Vortex

The irrotational (or potential) vortex is a classical example of a potential flow taught in fundamental fluid dynamics courses. In polar coordinates, its potential

and stream functions are defined by

$$\phi(r,\theta) = \frac{\Gamma}{2\pi}\theta, \tag{2.3a}$$

$$\psi(r,\theta) = -\frac{\Gamma}{2\pi}\ln(r), \tag{2.3b}$$

where Γ is the circulation (or strength) of the vortex. The corresponding velocity components in polar coordinates are therefore given by

$$u_r = \frac{\partial\phi}{\partial r} = \frac{1}{r}\frac{\partial\psi}{\partial\theta} = 0, \tag{2.4a}$$

$$u_\theta = \frac{1}{r}\frac{\partial\phi}{\partial\theta} = -\frac{\partial\psi}{\partial r} = \frac{\Gamma}{2\pi r}. \tag{2.4b}$$

In Cartesian coordinates, given that $r^2 = x^2 + y^2$ and $\tan(\theta) = y/x$, the potential and stream functions can be described as

$$\phi(x,y) = \frac{\Gamma}{2\pi}\arctan\left(\frac{y}{x}\right), \tag{2.5a}$$

$$\psi(x,y) = -\frac{\Gamma}{4\pi}\ln\left(x^2+y^2\right), \tag{2.5b}$$

where it should be understood that the 'arctan' in Eq. (2.5a) ought to return an angle respecting a chosen polar coordinate system (e.g., often either between $-\pi$ and π or between 0 and 2π). For these purposes, it is useful to use the `atan2` function (the four-quadrant inverse tangent) available in most programming languages (including MATLAB), which considers the sign of x and y and returns an angle between $-\pi$ and π. The corresponding velocity components in Cartesian coordinates are given by

$$u = -\frac{\Gamma}{2\pi}\frac{y}{x^2+y^2}, \tag{2.6a}$$

$$v = \frac{\Gamma}{2\pi}\frac{x}{x^2+y^2}. \tag{2.6b}$$

The components of the velocity gradient tensor are then given by

$$\frac{\partial u}{\partial x} = \frac{\Gamma}{2\pi}\left(\frac{2xy}{\left(x^2+y^2\right)^2}\right), \tag{2.7a}$$

$$\frac{\partial u}{\partial y} = -\frac{\Gamma}{2\pi}\left(\frac{1}{x^2+y^2} - \frac{y^2}{\left(x^2+y^2\right)^2}\right), \tag{2.7b}$$

$$\frac{\partial v}{\partial x} = \frac{\Gamma}{2\pi}\left(\frac{1}{x^2+y^2} - \frac{x^2}{\left(x^2+y^2\right)^2}\right), \tag{2.7c}$$

$$\frac{\partial v}{\partial y} = -\frac{\Gamma}{2\pi}\left(\frac{2xy}{\left(x^2+y^2\right)^2}\right). \tag{2.7d}$$

In MAT*fluids*, an irrotational vortex can be defined using the `irrotVortex` function. For example,

```
>> coord.x = linspace(−1, 1, 51).';
>> coord.y = coord.x;
>> gamma = 1;
>> [vel,vgt] = irrotVortex(coord, gamma);
```

2.3 The Rotational (or Solid Body) Vortex

Although solid body rotation cannot be associated with a potential function (i.e., it is not a potential flow), it can be associated with the stream function

$$\psi(r,\theta) = -\frac{1}{2}\Omega r^2, \tag{2.8}$$

where Ω represents the rate of rotation of the vortex.

2.4 A Simple Analytical Double Gyre Model

Shadden et al. (2005) described a simple analytical model of a double gyre flow. As the authors state, the model should not be seen as an approximate solution to a real fluid flow. The model should simply be interpreted as a simplified flow pattern that resembles the double gyre patterns seen in geophysical flows, such in Coulliette et al. (2007), and captures some salient features present in more complex models, such as that of Coulliette and Wiggins (2000, 2001).

The stream function of the double gyre model, as defined by Shadden et al. (2005), is given by

$$\psi(x,y,t) = -A\sin(\pi f(x,t))\sin(\pi y), \tag{2.9}$$

where

$$f(x,t) = a(t)\,x^2 + b(t)\,x; \tag{2.10a}$$
$$a(t) = \epsilon\sin(\omega t), \tag{2.10b}$$
$$b(t) = 1 - 2\epsilon\sin(\omega t). \tag{2.10c}$$

The corresponding velocity components are therefore given by

$$u = \frac{\partial\psi}{\partial y} = -A\pi\sin(\pi f(x,t))\cos(\pi y), \tag{2.11a}$$
$$v = -\frac{\partial\psi}{\partial x} = A\pi\frac{\partial f}{\partial x}\cos(\pi f(x,t))\sin(\pi y) \tag{2.11b}$$

where, of course,

$$\frac{\partial f}{\partial x} = 2a\left(t\right)x + b\left(t\right). \tag{2.12}$$

Note that in geophysical flows, the sign convention used in Eqs. (2.11a) and (2.11b) is often the opposite, namely, $u = -\partial\psi/\partial y$ and $v = \partial\psi/\partial x$. If this is the convention you wish to use, the sign of the stream function in Eq. (2.9) should also be changed for consistency. The resulting velocity field satisfies continuity by construction and therefore can be a useful tool to validate many concepts and codes in post-processing. For example, the components of the velocity gradient tensor are easily obtained from the derivatives of the velocity field, i.e.,

$$\frac{\partial u}{\partial x} = -A\pi^2\frac{\partial f}{\partial x}\cos\left(\pi f\left(x,t\right)\right)\cos\left(\pi y\right) \tag{2.13a}$$

$$\frac{\partial u}{\partial y} = A\pi^2\sin\left(\pi f\left(x,t\right)\right)\sin\left(\pi y\right) \tag{2.13b}$$

$$\frac{\partial v}{\partial x} = A\pi\sin\left(\pi y\right)\left[\frac{\partial^2 f}{\partial x^2}\cos\left(\pi f\left(x,t\right)\right) - \pi\left(\frac{\partial f}{\partial x}\right)^2\sin\left(\pi f\left(x,t\right)\right)\right] \tag{2.13c}$$

$$\frac{\partial v}{\partial y} = A\pi^2\frac{\partial f}{\partial x}\cos\left(\pi f\left(x,t\right)\right)\cos\left(\pi y\right). \tag{2.13d}$$

Chapter 3

Plot Toolbox

Chapter 4

Derivative Schemes

In the case of analytical flows, obtaining various physical quantities, such as vorticity and strain rate, is generally an exercise in computing derivatives of the velocity field. However, in the case of data generated by numerical simulations or experimental measurements, the derivatives must be computed from the data points. MAT*fluids* uses various schemes to achieve this. Due to the number of included derivative estimation functions, a nomenclature was devised that make the properties of the scheme evident. For example, a derivative estimation function may have the name `FDO2ExD1uCtr`. The first two characters denote the underlying method, where '`FD`' in the example refers to 'finite differences'. The second two characters denote the order of the scheme being used with an '`O`' followed by a number representing the order (i.e., the example function represents a second-order scheme). The following two characters denote the nature of the method, namely, whether it is explicit '`Ex`' or implicit '`Im`'. The next three characters represent the order of the derivative being approximated (i.e., a '`D`' followed by a number representing the order) as well as whether the grid is uniform '`u`' or variable '`v`'. Any remaining characters are intended to provide additional information, such as '`Ctr`' in the example suggesting that a centred scheme is primarily used.

A wide range of finite difference schemes can be derived using multiple Taylor series expansions centred at a point of interest. For instance, say we wish to compute derivatives of some dependent variable f with respect to some independent variable x numerically. If we are interested in the derivative at any particular point x_j (assume for the moment a uniform discretization of size Δx), we can express the value of f at x_{j+n} using a Taylor series expansion centred at x_j as

$$f\left(x_{j+n}\right) = f\left(x_j\right) + n\Delta x \left.\frac{\partial f}{\partial x}\right|_j + \frac{\left(n\Delta x\right)^2}{2!} \left.\frac{\partial^2 f}{\partial x^2}\right|_j + \frac{\left(n\Delta x\right)^3}{3!} \left.\frac{\partial^3 f}{\partial x^3}\right|_j + \cdots \quad (4.1)$$

and, likewise, for the value of f at x_{j-n} this amounts to

$$f\left(x_{j-n}\right) = f\left(x_j\right) - n\Delta x \left.\frac{\partial f}{\partial x}\right|_j + \frac{\left(n\Delta x\right)^2}{2!} \left.\frac{\partial^2 f}{\partial x^2}\right|_j - \frac{\left(n\Delta x\right)^3}{3!} \left.\frac{\partial^3 f}{\partial x^3}\right|_j + \cdots . \quad (4.2)$$

Although it may get tedious, by combining several expressions like Eqs. (4.1) and (4.2), a scheme can be derived that is accurate up to some order of truncation error.

4.1 Second-Order Explicit Schemes

4.1.1 Estimation of First Derivatives

Perhaps the most common derivative schemes used in post-processing, at least as a first attempt, are the second-order schemes. For points that share a neighbour on each side, the second-order centred scheme often provides a good derivative estimation and is given by

$$\left.\frac{\partial f}{\partial x}\right|_j = \frac{f\left(x_{j+1}\right) - f\left(x_{j-1}\right)}{2\Delta x} + O\left(\Delta x^2\right), \quad (4.3)$$

which can be derived by subtracting Eq. (4.2) from Eq. (4.1), both with $n = 1$, and solving for $\partial f/\partial x$. Unfortunately, points that lie at a boundary of some sort will not have neighbouring points on each side and therefore the second-order centred scheme will not work, at least not without some added condition (e.g., periodicity or symmetry of the domain at the boundary). In such a case, if there are two points ahead of the boundary node, a second-order forward scheme can be used,

$$\left.\frac{\partial f}{\partial x}\right|_j = \frac{-3f\left(x_j\right) + 4f\left(x_{j+1}\right) - f\left(x_{j+2}\right)}{2\Delta x} + O\left(\Delta x^2\right), \quad (4.4)$$

and if there are two points behind the boundary node, a second-order forward scheme can be used, namely,

$$\left.\frac{\partial f}{\partial x}\right|_j = \frac{3f\left(x_j\right) - 4f\left(x_{j-1}\right) + f\left(x_{j-2}\right)}{2\Delta x} + O\left(\Delta x^2\right). \quad (4.5)$$

In MAT*fluids*, a complete second-order scheme is included in `FDO2ExD1uCtr`, which primarily uses the second-order centred scheme of Eq. (4.3) along with the forward and backward differences of Eqs. (4.4) and (4.5) only at the boundaries. As a measure of the accuracy of the method, let's use the potential vortex defined below.

```
» coord.x = linspace(-1, 1, 51).';
» coord.y = coord.x;
» gamma = 1;
» [vel,vgt] = irrotVortex(coord, gamma);
```

Now, let's compute the derivatives of the velocity field '`vel`' using the `FDO2ExD1uCtr` function and verify the accuracy against the analytical derivatives contained in '`vgt`'.

```
» ds = coord.x(2) - coord.x(1);
» ux = FDO2ExD1uCtr(vel.u, ds, 2);
» uy = FDO2ExD1uCtr(vel.u, ds, 1);
» vx = FDO2ExD1uCtr(vel.v, ds, 2);
» vy = FDO2ExD1uCtr(vel.v, ds, 1);
» sqrt(sum((vgt.ux - ux).^2, 'all', 'omitnan')/numel(vgt.ux))
ans =
    0.2232
» sqrt(sum((vgt.uy - uy).^2, 'all', 'omitnan')/numel(vgt.uy))
ans =
    0.8057
» sqrt(sum((vgt.vx - vx).^2, 'all', 'omitnan')/numel(vgt.vx))
ans =
    0.8057
» sqrt(sum((vgt.vy - vy).^2, 'all', 'omitnan')/numel(vgt.vy))
ans =
    0.2232
```

Chapter 5

Efficient MATLAB Code

In general, in scientific programming, we often strive to write a code in the simplest and most elegant way possible. Fortunately, MATLAB has many capabilities and functions that allow us to achieve this goal and a proper understanding of them can lead to pretty efficient codes.

5.1 Loops and Test Statements

5.2 Order of Mathematical Operations

The order in which mathematical operations are executed in a code can have a huge impact on the efficiency of a code. Consider the following simple code:

```
>> n = 10000;
>> x = rand(n, 1);
>> A = rand(n, 10);
>> tic; B = A*A'*x; toc;
Elapsed time is 0.371698 seconds.
>> tic; C = A*(A'*x); toc;
Elapsed time is 0.000251 seconds.
```

5.3 Low-Memory Matrix Products

When dealing with large datasets, it is important to understand different ways of dealing with matrix products, particularly when at least one of the matrices is so large that it needs to be partitioned and stored in multiple files. Given an $p \times n$ matrix $\mathbf{X}$, two of the simplest and particularly useful partitions we could use are columnwise and rowwise partitions, i.e.,

$$\mathbf{X} = \begin{bmatrix} | & & | \\ \mathbf{x}_1 & \cdots & \mathbf{x}_n \\ | & & | \end{bmatrix} = \begin{bmatrix} — & \widetilde{\mathbf{x}}_1 & — \\ & \vdots & \\ — & \widetilde{\mathbf{x}}_p & — \end{bmatrix}. \tag{5.1}$$

Say we have another $n \times q$ matrix

$$\mathbf{Y} = \begin{bmatrix} | & & | \\ \mathbf{y}_1 & \cdots & \mathbf{y}_q \\ | & & | \end{bmatrix} = \begin{bmatrix} — & \widetilde{\mathbf{y}}_1 & — \\ & \vdots & \\ — & \widetilde{\mathbf{y}}_n & — \end{bmatrix}, \tag{5.2}$$

and we are interested in the product $\mathbf{A} = \mathbf{XY}$. If $q \gg \max(p, n)$, a particularly useful way of representing the product is

$$\mathbf{A} = \begin{bmatrix} | & & | \\ \mathbf{X}\mathbf{y}_1 & \cdots & \mathbf{X}\mathbf{y}_q \\ | & & | \end{bmatrix}, \tag{5.3}$$

the reason being that if the matrix $\mathbf{X}$ can be held in memory, then the matrix $\mathbf{A}$ can be computed column by column by loading successive columns of $\mathbf{Y}$ (i.e., $\mathbf{a}_j = \mathbf{X}\mathbf{y}_j$). If the matrix $\mathbf{Y}$ is stored in groups of columns, then the matrix $\mathbf{A}$ can also be computed in groups of columns (i.e., $\mathbf{A}_{j_1}^{j_2} = \mathbf{X}\mathbf{Y}_{j_1}^{j_2}$). If instead $p \gg \max(q, n)$, it is more useful to represent the product as

$$\mathbf{A} = \begin{bmatrix} | & & | \\ \mathbf{X}\mathbf{y}_1 & \cdots & \mathbf{X}\mathbf{y}_q \\ | & & | \end{bmatrix}. \tag{5.4}$$

References

Coulliette, C., Lekien, F., Paduan, J. D., Haller, G., & Marsden, J. E. (2007, September). Optimal pollution mitigation in Monterey Bay based on coastal radar data and nonlinear dynamics. *Environmental Science & Technology*, **41**(18), 6562–6572.

Coulliette, C., & Wiggins, S. (2000, June). Intergyre transport in a wind-driven, quasigeostrophic double gyre: An application of lobe dynamics. *Nonlinear Processes in Geophysics*, **7**(1/2), 59–85.

Coulliette, C., & Wiggins, S. (2001, April). Intergyre transport in a wind-driven, quasigeostrophic double gyre: An application of lobe dynamics. *Nonlinear Processes in Geophysics*, **8**(1/2), 69–94.

Shadden, S. C., Lekien, F., & Marsden, J. E. (2005, December). Definition and properties of Lagrangian coherent structures from finite-time Lyapunov exponents in two-dimensional aperiodic flows. *Physica D: Nonlinear Phenomena*, **212**(3–4), 271–304.