What is Computational Fluid Dynamics (CFD)?

- Grid must be sufficiently fine to resolve the flow

The number of grid points (or control volumes) available determines the complexity of the problem that can be solved and the accuracy of the solution.
Bits and Bytes:
64 bits = 8 bytes = one number with ~16 digits precision
Memory requirement for 3-D calculations
100^3 = 10^6 x 10 bytes x 10 numbers/node = 0.1 GB
200^3 = 8 x 10^6 x 10 bytes x 10 numbers/node = 1 GB
1000^3 = 10^9 x 10 bytes x 10 numbers/node = 100 GB

FLOPS (Floating-point operations per second)
CRAY-1 (1976) - 133 Megaflops
ASCI White (2000) - 12.28 Teraflops
  → Increase by a million in a quarter century!!

CFD is used by many different people for many different things

Industrial problems: The goal is generally to obtain data (quantitative and qualitative) that can be used in design of devices or processes. Often it is necessary to use “subgrid” or closure models for unresolved processes

Academic problems: The goal is to understand the physical aspects of the process, often the goal is to construct “subgrid” or closure models for industrial computations

A few random examples
CFD is now a standard part of the toolkit used both in scientific studies and engineering predictions
The numerical solution yields the velocity and pressure field everywhere. Usually, it is the force on a boundary that is of interest. Looking at the flow field can, however, be very informative

Many website contain information about fluid dynamics and computational fluid dynamics specifically. Those include

NASA site with CFD images
http://www.nas.nasa.gov/SC08/images.html

CFD Online: An extensive collection of information but not always very informative
http://www.cfd-online.com/

eFluids.com is a monitored site with a large number of fluid mechanics material
http://www.e-fluids.com/
Short History of CFD

Numerical Approximations
- Richardson (1910): Early vision of the role of numerical predictions
- Courant, Friedrichs, and Lewy (1928): Stability analysis of the advection equation (CFL condition)
- von Neumann: Role of computations, stability analysis
- Lax and the Courant group: Promoting numerical computations
- Harlow and the Los Alamos group: MAC, CIC, and other methods
- Spalding and industrial applications: CHAM was the first provider of general-purpose CFD software. The original PHOENICS appeared in 1981.
- NCAR and the Supercomputer Centers

Early papers about solutions of the Navier-Stokes equations

Commercial Codes

CHAM (Concentration Heat And Momentum) founded in 1974 by Prof. Brian Spalding was the first provider of general-purpose CFD software. The original PHOENICS appeared in 1981.

The first version of the FLUENT code was launched in October 1983 by Creare Inc. Fluent Inc. was established in 1988.

STAR-CD’s roots go back to the foundation of Computational Dynamics in 1987 by Prof. David Gosman.

The original codes were relatively primitive, hard to use, and not very accurate.
Computational Fluid Dynamics I

What to expect and when to use commercial package:

The current generation of CFD packages generally is capable of producing accurate solutions of simple flows. The codes are, however, designed to be able to handle very complex geometries and complex industrial problems. When used with care by a knowledgeable user CFD codes are an enormously valuable design tool.

Commercial CFD codes are rarely useful for state-of-the-art research due to accuracy limitations, the limited access that the user has to the solution methodology, and the limited opportunities to change the code if needed.

Major current players include

Ansys (Fluent and other codes)  
http://www.fluent.com/  
http://www.ansys.com/

adapco: (starCD)  
http://www.cd-adapco.com/

Others  
CHAM: http://www.cham.co.uk/  

Computational Resources  

Computational Fluid Dynamics has traditionally been one of the most demanding computational application. It has therefore been the driver for the development of the most powerful computers.

Single processor computers

Vector computers (80’s)
Parallel computers (90’s)  
Shared Memory  
Distributed Memory

World’s fastest computers

For up-to-date information about the World’s fastest computers, see:

http://www.top500.org/  
http://www.top500.org/list/2008/11/100

Images:  
http://www.cisl.ucar.edu/computers/bluefire/gallery.jsp  
http://www.nasa.gov/News/Images/columbia_2.html

Introduction to Computational Fluid Dynamics—II

Gretar Tryggvason  
Spring 2010
Computational Fluid Dynamics I

**Coarse Goals:**
Learn how to solve the Navier-Stokes and Euler equations for engineering problems using both customized codes and a commercial code.

Hear about various concepts to allow continuing studies of the literature.

**Ways:**
Detailed coverage of selected topics, such as: simple finite difference methods, accuracy, stability, etc.
Short introduction to FLUENT
Rapid coverage of other topics, such as: multigrid, monotone advection, unstructured grids.

Using CFD to solve a problem:

Preparing the data (preprocessing):
Setting up the problem, determining flow parameters and material data and generating a grid.

Solving the problem

Analyzing the results (postprocessing):
Visualizing the data and estimating accuracy. Computing forces and other quantities of interest.

**CFD is an interdisciplinary topic**

**Background needed:**

Undergraduate Numerical Analysis and Fluid Mechanics

Graduate Level Fluid Mechanics (can be taken concurrently).

Basic computer skills. We will use MATLAB for some of the homework.
Computational Fluid Dynamics I

Course outline

Part I
A brief introduction to CFD and review of fluid mechanics

Part II
Standard Numerical Analysis of partial differential equations

Part III
Advanced topics in CFD

Course homepage: http://users.wpi.edu/~gretar/me612.html

Course outline—Part I
Introduction, what is CFD, examples, computers, elementary numerical analysis, course administration

Elementary numerical analysis, integration of ordinary differential equations

Elementary numerical analysis, accuracy, stability, partial differential equations

Review of fluid mechanics: the governing equations

Finite Difference solution of the Navier-Stokes Equations in vorticity/streamfunction form

Introduction to Commercial CFD codes. Using commercial codes

Course outline—Part II
First order Partial Differential Equations (PDF’s). Characteristics. Classification of Second Order PDF’s.


Algorithms for Parabolic equations.

Algorithms for Elliptic equations.

Putting it together, solving the Navier-Stokes Equations in Primitive Variables, the MAC Method

Course outline—Part III
Complex Domains. Body fitted Coordinates.

Complex Domains. Grid Generation

Introduction to Turbulence, Multiphase flow, and combustion

Parallel Computations, Visualization

Direct Numerical Simulations of Multiphase Flows

Class hours:
MW 6:30pm – 7:50pm, HL 116

Each lecture consists of three 25 minutes (or so) “sessions”

Lecture material will be available on the web and will be handed out in class

The homework consists of problem sets (about one per class) and four computer projects.

Grading:
Four projects, homework, two quizzes.

Collaborations
You are free to discuss the homework problems and the projects with your fellow students, but your solution should be your own work.
Computational Fluid Dynamics I

Project 1
Integrate one-dimensional PDE in time

Project 2
Using FLUENT to solve the Navier-Stokes equations in a given geometry, OR a student proposed problem

Project 3
Solving the Euler equations (1D)

Project 4
Solve the Navier-Stokes equations in the primitive variables (pressure and velocity) for a given 2D geometry

Elementary Numerical Analysis: Finite Difference Approximations-I
Grétar Tryggvason
Spring 2010

Integration of Ordinary Differential Equations in Time

Objectives:

- Introduce the basic concepts needed to solve a partial differential equation using finite difference methods.
- Discuss basic time integration methods, ordinary and partial differential equations, finite difference approximations, accuracy.
- Show the implementation of numerical algorithms into actual computer codes.
Integrating a first-order ordinary differential equation in time

\[ \frac{df}{dt} = g(t, f) \]

The initial condition must also be specified:

\[ f(t_0) = f_0 \]

A numerical solution of

\[ \frac{df}{dt} = g(t, f) \]

consists of discrete values of \( f \) at discrete times:

\[
\begin{align*}
  f^0 &= f(t_0) \\
  f^1 &= f(t_0 + \Delta t) \\
  f^2 &= f(t_0 + 2\Delta t) \\
  f^{-1} &= f(t - \Delta t) \\
  f^0 &= f(t) \\
  f^{n+1} &= f(t + \Delta t)
\end{align*}
\]

\[ f^n = f(t + n\Delta t) \]

To advance:

\[ \frac{df}{dt} = g(f, t) \]

Integrate:

\[
\int_t^{t + \Delta t} g(f, \tau) d\tau = f^{n+1} - f^n = \int_t^{t + \Delta t} g(f, \tau) d\tau
\]

Approximate:

\[
\int_t^{t + \Delta t} g(f, \tau) d\tau = g(t)\Delta t = f^{n+1} - f^n
\]

Forward Euler:

\[ f^{n+1} = f^n + g^n \Delta t \]

Backward Euler:

\[ f^{n+1} = f^n + g^{n+1} \Delta t \]

Approximate:

\[
\int_t^{t + \Delta t} g(f, \tau) d\tau = g(t + \Delta t)\Delta t = f^{n+1} - f^n
\]

Forward Euler:

\[ f^{n+1} = f^n + g^{n+1} \Delta t \]
\[ f_{n+1} = f_n + \frac{1}{2}(g^n + g^{n+1})\Delta t \]

**Summary:**

- **Forward Euler**
  \[ f_{n+1} = f_n + g^n \Delta t \]

- **Backward Euler**
  \[ f_{n+1} = f_n + g^{n+1} \Delta t \]

- **Trapezoidal Rule**
  \[ f_{n+1} = f_n + \frac{1}{2}(g^n + g^{n+1})\Delta t \]

**Example**

The exact solution is

\[ f(t) = e^{-t} \]

**Accuracy**

A short code, using Matlab (EX1)

```matlab
% a simple code for several integration methods
nstep=5;dt=0.5;
f1=zeros(nstep,1);f2=zeros(nstep,1); f3=zeros(nstep,1);
fex=zeros(nstep,1); t=zeros(nstep,1);
t(1)=0;f1(1)=1;f2(1)=1;f3(1)=1;fex(1)=1;
for i=2:nstep
    f1(i)=f1(i-1)-dt*f1(i-1); % Forward Euler
    f2(i)=f2(i-1)/(1.0+dt); % Backward Euler
    f3(i)=f3(i-1)*(1.0-0.5*dt)/(1.0+0.5*dt); % Trapezoidal Rule
    t(i)=t(i-1)+dt;
    fex(i)=exp(-t(i));
end;
plot(t,f1);hold on;plot(t,f2,'r');plot(t,f3,'k');
plot(t,fex, 'r', 'linewidt',3); set(gca,'fontsize',24,'linewidt',2);
```
Clearly the numerical solutions have the same behavior as the exact solution but only the trapezoidal rule results in numerical values that are approximately the same. The obvious question is:

Can we improve the accuracy of the forward and backward Euler method?

Re-run the forward Euler method with smaller time steps.

The error at \( t=2.5 \), for the forward Euler Method, defined by:

\[
E = |f_{\text{num}}(2.5) - \exp(-2.5)|
\]

<table>
<thead>
<tr>
<th>( \Delta t )</th>
<th>( E )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>0.0728</td>
</tr>
<tr>
<td>0.25</td>
<td>0.0303</td>
</tr>
<tr>
<td>0.125</td>
<td>0.0139</td>
</tr>
<tr>
<td>0.0625</td>
<td>0.0067</td>
</tr>
<tr>
<td>0.03125</td>
<td>0.0033</td>
</tr>
</tbody>
</table>

As the time step becomes smaller, it is clear that the error goes to zero.
Analyze the error in evaluating the integral using a Taylor series

\[ g = g^n + \left( \frac{dg}{dt} \right)^n \tau + \cdots \]

substitute

\[ \int_{t^n}^{t^{n+\Delta t}} g \, d\tau = \int_{t^n}^{t^{n+\Delta t}} g^n \, d\tau + \int_{t^n}^{t^{n+\Delta t}} \left( \frac{dg}{dt} \right)^n \tau \, d\tau + \cdots \]

integrate

\[ \int_{t^n}^{t^{n+\Delta t}} gd\tau = g^n \Delta t + \left( \frac{dg}{dt} \right)^n \frac{\Delta t^2}{2} + L \]

Error at each time step

\[ f^{n+1} = f^n + g^n \Delta t + O(\Delta t^2) \]

The error at each time step is important, but it is the total error, when integrating over a given time period \( T \) that is most important. If the number of time steps is \( N \), the total error is

\[ E = N \times O(\Delta t^2) \]

Since the number of time steps is \( N = T / \Delta t \):

\[ E = \frac{T}{\Delta t} \times O(\Delta t^2) = O(\Delta t) \]

First Order Method

If the error is of n-th. order:

\[ E = C \Delta t^n = C \left( \frac{1}{\Delta t} \right)^{-n} \]

Taking the log:

\[ \ln E = \ln C \left( \frac{1}{\Delta t} \right)^{-n} = \ln C - n \ln \frac{1}{\Delta t} \]

On a log-log plot, the \( E \) versus \( 1/\Delta t \) curve should therefore have a slope of \( -n \)

Summary

Integration of an ordinary differential equation (ODE) in time

Implementation in a MATLAB code

Error analysis