Compressible Flow - TME085

Lecture 14

Niklas Andersson

Chalmers University of Technology
Department of Mechanics and Maritime Sciences
Division of Fluid Mechanics
Gothenburg, Sweden

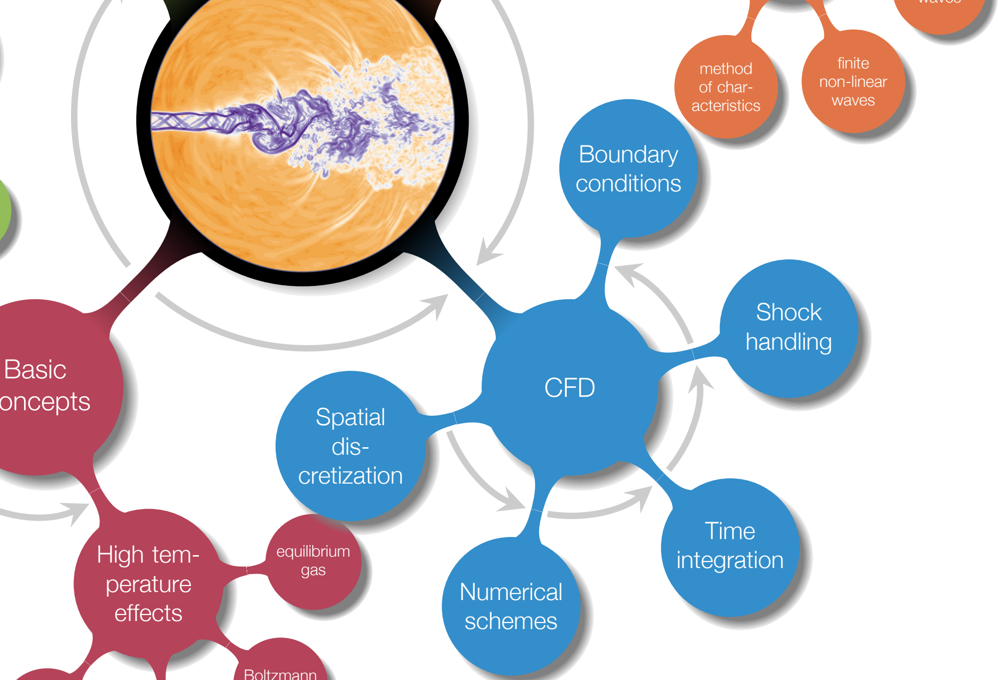
niklas.andersson@chalmers.se

```
#undef __FUNCT__
#define __FUNCT__ "RungeKutta::fwd"
PetscErrorCode RungeKutta::fwd(Domain *dom){
    PetscErrorCode ierr=0;

    ierr=G3DCopy(dom->cons,cons0);CHKERRQ(ierr);

    /* RK1 */

    dom->update();

    dcons->evaluate(dom);

    ierr=G3DWAXPY(dom->cons,1.0,dcons,cons0);CHKERRQ(ierr);
    ierr=G3DAXPBY(cons0,0.5,0.5,dom->cons);CHKERRQ(ierr);

    /* RK2 */

    dom->update();
    dcons->evaluate(dom);

    ierr=G3DWAXPY(dom->cons,0.5,dcons,cons0);CHKERRQ(ierr);

    /* RK3 */
```

Chapter 12 - The Time-Marching Technique

```
    return(ierr);
```

Overview



method of char-acteristics
finite non-linear waves
Boundary conditions
Shock handling
CFD
Spatial dis-cretization
Time integration
Numerical schemes
Basic oncepts
High tem-perature effects
equilibrium gas
Boltzmann

Learning Outcomes

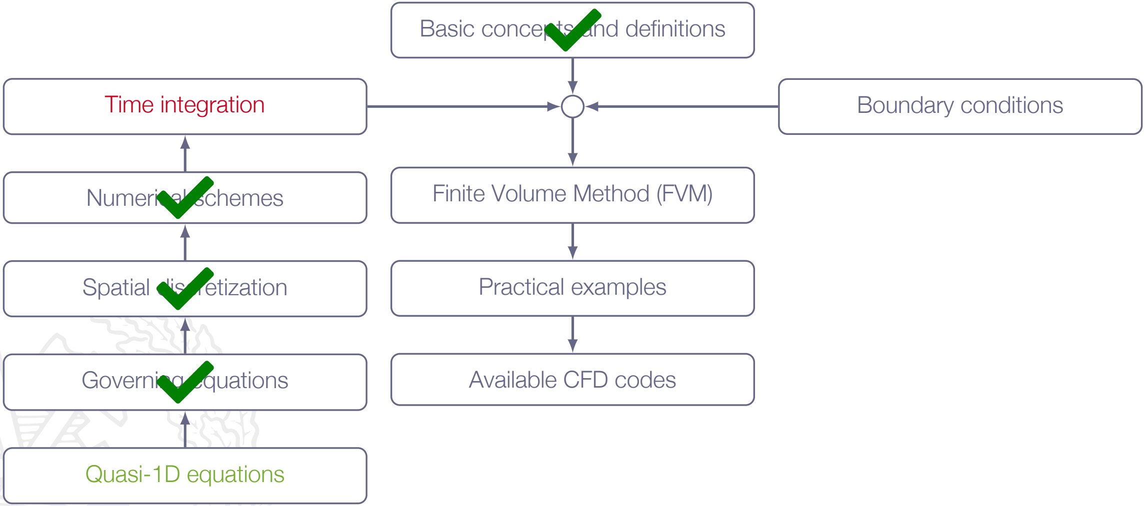
12 **Explain** the main principles behind a modern Finite Volume CFD code and such concepts as explicit/implicit time stepping, CFL number, conservation, handling of compression shocks, and boundary conditions

14 **Analyze** and **verify** the quality of the numerical solution

15 **Explain** the limitations in fluid flow simulation software

time for CFD!

Roadmap - The Time-Marching Technique



Time Stepping

CFLOW

cell-averaged quantity
face-averaged quantity
source term

$$VOL_i \frac{d}{dt}\bar{\rho}_i - \overline{(\rho u)}_{i-\frac{1}{2}} A_{i-\frac{1}{2}} + \overline{(\rho u)}_{i+\frac{1}{2}} A_{i+\frac{1}{2}} = 0$$

$$VOL_i \frac{d}{dt}\overline{(\rho u)}_i - \overline{(\rho u^2 + p)}_{i-\frac{1}{2}} A_{i-\frac{1}{2}} + \overline{(\rho u^2 + p)}_{i+\frac{1}{2}} A_{i+\frac{1}{2}} = \bar{p}_i \left(A_{i+\frac{1}{2}} - A_{i-\frac{1}{2}}\right)$$

$$VOL_i \frac{d}{dt}\overline{(\rho e_o)}_i - \overline{(\rho u h_o)}_{i-\frac{1}{2}} A_{i-\frac{1}{2}} + \overline{(\rho u h_o)}_{i+\frac{1}{2}} A_{i+\frac{1}{2}} = 0$$

Application of these equations to all cells $i \in \{1, 2, ..., N\}$ of the computational domain results in a system of ODEs

Quasi-One-Dimensional Flow - Spatial Discretization

cell-averaged quantity
face-averaged quantity
source term

$$VOL_i \frac{d}{dt}\bar{\rho}_i = \overline{(\rho u)}_{i-\frac{1}{2}} A_{i-\frac{1}{2}} - \overline{(\rho u)}_{i+\frac{1}{2}} A_{i+\frac{1}{2}}$$

$$VOL_i \frac{d}{dt}\overline{(\rho u)}_i = \overline{(\rho u^2 + p)}_{i-\frac{1}{2}} A_{i-\frac{1}{2}} - \overline{(\rho u^2 + p)}_{i+\frac{1}{2}} A_{i+\frac{1}{2}} + \bar{p}_i \left(A_{i+\frac{1}{2}} - A_{i-\frac{1}{2}}\right)$$

$$VOL_i \frac{d}{dt}\overline{(\rho e_o)}_i = \overline{(\rho u h_o)}_{i-\frac{1}{2}} A_{i-\frac{1}{2}} - \overline{(\rho u h_o)}_{i+\frac{1}{2}} A_{i+\frac{1}{2}}$$

cell-averaged quantity
face-averaged quantity
source term

$$\frac{d}{dt}\bar{\rho}_i = \frac{1}{VOL_i}\left[\overline{(\rho u)}_{i-\frac{1}{2}}A_{i-\frac{1}{2}} - \overline{(\rho u)}_{i+\frac{1}{2}}A_{i+\frac{1}{2}}\right]$$

$$\frac{d}{dt}\overline{(\rho u)}_i = \frac{1}{VOL_i}\left[\overline{(\rho u^2+p)}_{i-\frac{1}{2}}A_{i-\frac{1}{2}} - \overline{(\rho u^2+p)}_{i+\frac{1}{2}}A_{i+\frac{1}{2}} + \bar{p}_i\left(A_{i+\frac{1}{2}} - A_{i-\frac{1}{2}}\right)\right]$$

$$\frac{d}{dt}\overline{(\rho e_o)}_i = \frac{1}{VOL_i}\left[\overline{(\rho u h_o)}_{i-\frac{1}{2}}A_{i-\frac{1}{2}} - \overline{(\rho u h_o)}_{i+\frac{1}{2}}A_{i+\frac{1}{2}}\right]$$

$$\frac{d}{dt}\overline{\mathbf{Q}}_i = \mathbf{F}(\overline{\mathbf{Q}}_i) \text{ where } \overline{\mathbf{Q}}_i = [\overline{\rho},\ \overline{\rho u},\ \overline{\rho e_o}]_i,\ i \in \{1 : NCells\}$$

Time Stepping

The system of ODEs obtained from the spatial discretization in vector notation

$$\frac{d}{dt}\mathbf{Q} = \mathbf{F}(\mathbf{Q})$$

$\mathbf{Q}$ is a vector containing all DOFs in all cells

$\mathbf{F}(\mathbf{Q})$ is the **time derivative** of $\mathbf{Q}$ resulting from above mentioned **flux approximations** - *non-linear vector-valued function*

Time Stepping

Three-stage Runge-Kutta - *one example of many*:

Explicit time-marching scheme

Second-order accurate

Time Stepping - Three-stage Runge-Kutta

$$\frac{d}{dt}\mathbf{Q} = \mathbf{F}(\mathbf{Q})$$

Let $\mathbf{Q}^n = \mathbf{Q}(t_n)$ and $\mathbf{Q}^{n+1} = \mathbf{Q}(t_{n+1})$

t_n is the current time level and t_{n+1} is the next time level
$\Delta t = t_{n+1} - t_n$ is the solver time step

Algorithm:

$$\begin{aligned}
1.\ \ \mathbf{Q}^* &= \mathbf{Q}^n + \Delta t\mathbf{F}(\mathbf{Q}^n) \\
2.\ \ \mathbf{Q}^{**} &= \mathbf{Q}^n + \frac{1}{2}\Delta t\mathbf{F}(\mathbf{Q}^n) + \frac{1}{2}\Delta t\mathbf{F}(\mathbf{Q}^*) \\
3.\ \ \mathbf{Q}^{n+1} &= \mathbf{Q}^n + \frac{1}{2}\Delta t\mathbf{F}(\mathbf{Q}^n) + \frac{1}{2}\Delta t\mathbf{F}(\mathbf{Q}^{**})
\end{aligned}$$

Time Stepping - Three-stage Runge-Kutta

```
void RungeKutta::fwd(Domain *dom){
  G3DCopy(dom->cons,cons0);

  /* Runge-Kutta step 1 */

  dom->update();
  if(!G3DMode::constdt){LocalTimeStep(dom);}
  dcons->evaluate(dom);
  G3DWAXPY(dom->cons,1.0,dcons,cons0);
  G3DAXPBY(cons0,0.5,0.5,dom->cons);

  /* Runge-Kutta step 2 */

  dom->update();
  dcons->evaluate(dom);
  G3DWAXPY(dom->cons,0.5,dcons,cons0);

  /* Runge-Kutta step 3 */

  dom->update();
  dcons->evaluate(dom);
  G3DWAXPY(dom->cons,0.5,dcons,cons0);
}
```

Time Stepping - Explicit Schemes

Properties of explicit time-stepping schemes:

+ **Easy to implement** in computer codes
+ **Efficient execution** on most computers
+ Easy to adapt for **parallel execution** on distributed memory systems (*e.g.* Linux clusters)

- **Time step limitation** (CFL number)
- Convergence to steady-state **often slow** (there are, however, some remedies for this)

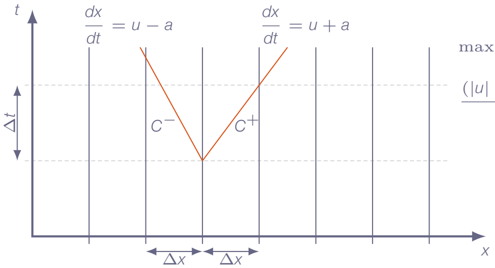
Time Stepping - Explicit Schemes

Courant-Friedrich-Levy **(CFL)** number - *one-dimensional case*:

$$CFL_i = \frac{\Delta t(|u_i| + a_i)}{\Delta x_i} \leq 1$$

Interpretation: The fastest characteristic (C^+ or C^-) must not travel longer than Δx during one time step

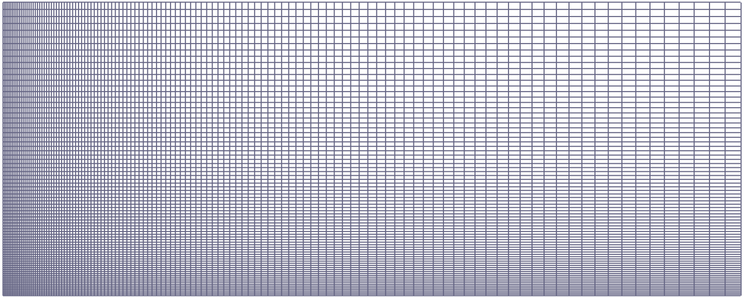
Time Stepping - Explicit Schemes



$$\max(|u-a|,|u+a|)\Delta t = (|u|+a)\Delta t \leq \Delta x \Rightarrow$$

$$\frac{(|u|+a)\Delta t}{\Delta x} = CFL \leq 1$$

Time Stepping - Explicit Schemes



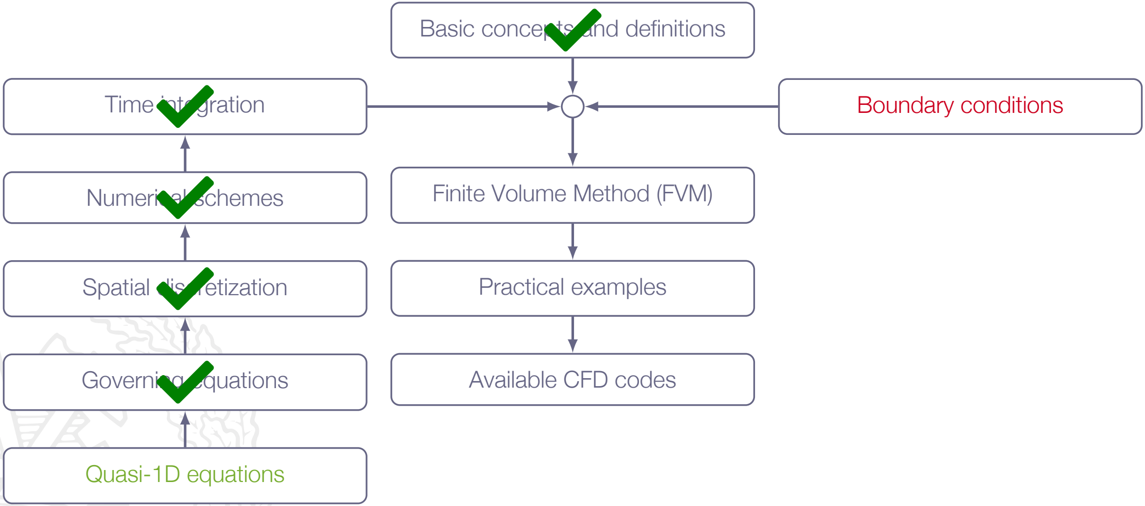
Steady-state problems:

- local time stepping
- each cell has an individual time step
- Δt_i maximum allowed value based on CFL criteria

Unsteady problems:

- time accurate
- all cells have the same time step
- $\Delta t_i = \min\{\Delta t_1, ..., \Delta t_N\}$

Roadmap - The Time-Marching Technique



Boundary Conditions

Boundary Conditions

Boundary conditions are very important for numerical simulation of compressible flows

Main reason: both **flow** and **acoustics** involved!

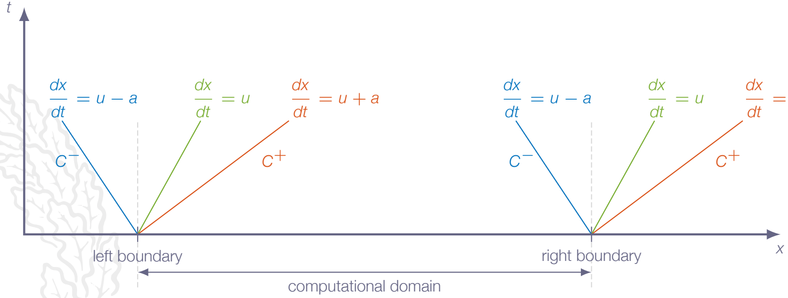
Example 1:
Finite-volume CFD code for Quasi-1D compressible flow (Time-marching procedure)

What boundary conditions should be applied at the left and right ends?

Boundary Conditions

three characteristics:

1. C^+
2. C^-
3. advection



Boundary Conditions

C^+ and C^- characteristics describe the transport of **isentropic pressure waves** (often referred to as **acoustics**)

The advection characteristic simply describes the **transport** of certain quantities **with the fluid itself** (for example **entropy**)

In one space dimension and time, these three characteristics, together with the quantities that are known to be constant along them, give a **complete description** of the time evolution of the flow

We can use the characteristics as a guide to tell us what information that should be specifed at the boundaries

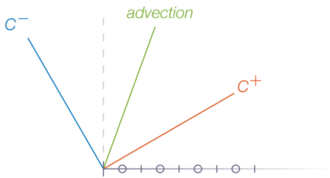
Left Boundary - Subsonic Inflow

we have three PDEs, and are solving for three unknowns

Subsonic inflow: $0 < u < a$

$$u - a < 0$$
$$u > 0$$
$$u + a > 0$$

one outgoing characteristic
two ingoing characteristics



Two variables should be **specified** at the boundary

The third variable must be left free

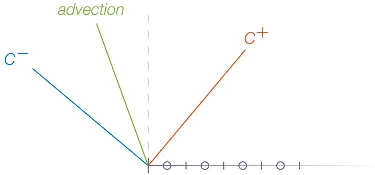
Left Boundary - Subsonic Outflow

we have three PDEs, and are solving for three unknowns

Subsonic outflow: $-a < u < 0$

$$u - a < 0$$
$$u < 0$$
$$u + a > 0$$

two outgoing characteristics
one ingoing characteristic



One variable should be **specified** at the boundary

The second and third variables must be left free

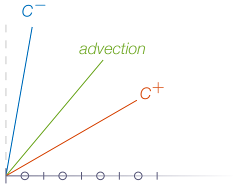
Left Boundary - Supersonic Inflow

we have three PDEs, and are solving for three unknowns

Supersonic inflow: $u > a$

$$u - a > 0$$
$$u > 0$$
$$u + a > 0$$

no outgoing characteristics
three ingoing characteristics



All three variables should be **specified** at the boundary

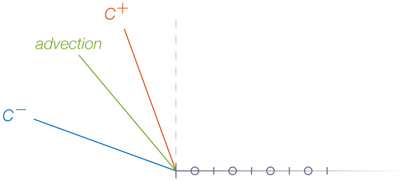
No variables must be left free

Left Boundary - Supersonic Outflow

we have three PDEs, and are solving for three unknowns

Supersonic outflow: $u < -a$

$$u - a < 0$$
$$u < 0$$
$$u + a < 0$$



three outgoing characteristics
no ingoing characteristics

No variables should be **specified** at the boundary

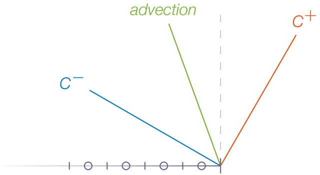
All variables must be left free

Right Boundary - Subsonic Inflow

we have three PDEs, and are solving for three unknowns

Subsonic inflow: $-a < u < 0$

$$u - a < 0$$
$$u < 0$$
$$u + a > 0$$



two ingoing characteristics
one outgoing characteristic

Two variables should be **specified** at the boundary

The third variables must be left free

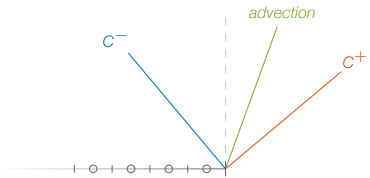
Right Boundary - Subsonic Outflow

we have three PDEs, and are solving for three unknowns

Subsonic outflow: $0 < u < a$

$$u - a < 0$$
$$u > 0$$
$$u + a > 0$$

one ingoing characteristic
two outgoing characteristics



One variable should be **specified** at the boundary

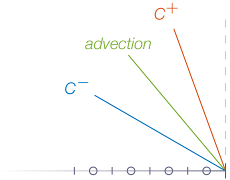
The second and third variables must be left free

Right Boundary - Supersonic Inflow

we have three PDEs, and are solving for three unknowns

Supersonic inflow: $u < -a$

$$u - a < 0$$
$$u < 0$$
$$u + a < 0$$



three ingoing characteristics
no outgoing characteristics

All three variables should be **specified** at the boundary

No variables must be left free

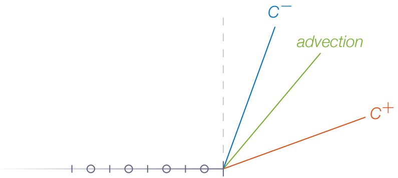
Right Boundary - Supersonic Outflow

we have three PDEs, and are solving for three unknowns

Supersonic outflow: $u > a$

$$u - a > 0$$
$$u > 0$$
$$u + a > 0$$

no ingoing characteristics
three outgoing characteristics



No variables should be **specified** at the boundary

All three variables must be left free

1D Boundary Conditions (Summary)

Characteristic		1D subsonic inflow (left)	1D subsonic inflow (right)
advection	$\mathbf{v} \cdot \mathbf{n}$	$(u,0,0) \cdot (-1,0,0) = -u < 0$	$(-u,0,0) \cdot (1,0,0) = -u < 0$
C^-	$\mathbf{v} \cdot \mathbf{n} - a$	$-u - a < 0$	$-u - a < 0$
C^+	$\mathbf{v} \cdot \mathbf{n} + a$	$-u + a > 0$	$-u + a > 0$
Characteristic		**1D subsonic outflow (left)**	**1D subsonic outflow (right)**
advection	$\mathbf{v} \cdot \mathbf{n}$	$(-u,0,0) \cdot (-1,0,0) = u > 0$	$(u,0,0) \cdot (1,0,0) = u > 0$
C^-	$\mathbf{v} \cdot \mathbf{n} - a$	$u - a < 0$	$u - a < 0$
C^+	$\mathbf{v} \cdot \mathbf{n} + a$	$u + a > 0$	$u + a > 0$
Characteristic		**1D supersonic inflow (left)**	**1D supersonic inflow (right)**
advection	$\mathbf{v} \cdot \mathbf{n}$	$(u,0,0) \cdot (-1,0,0) = -u < 0$	$(-u,0,0) \cdot (1,0,0) = -u < 0$
C^-	$\mathbf{v} \cdot \mathbf{n} - a$	$-u - a < 0$	$-u - a < 0$
C^+	$\mathbf{v} \cdot \mathbf{n} + a$	$-u + a < 0$	$-u + a < 0$
Characteristic		**1D supersonic outflow (left)**	**1D supersonic outflow (right)**
advection	$\mathbf{v} \cdot \mathbf{n}$	$(-u,0,0) \cdot (-1,0,0) = u > 0$	$(u,0,0) \cdot (1,0,0) = u > 0$
C^-	$\mathbf{v} \cdot \mathbf{n} - a$	$u - a > 0$	$u - a > 0$
C^+	$\mathbf{v} \cdot \mathbf{n} + a$	$u + a > 0$	$u + a > 0$

Subsonic Inflow (Left Boundary) - Example

Subsonic inflow: we should specify two variables

Alt	specified variable 1	specified variable 2	well-posed	non-reflective
1	p_o	T_o	X	
2	ρu	T_o	X	
3	s	J^+	X	X

well posed:

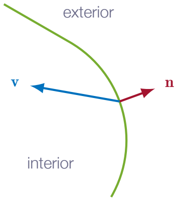
1. the problem has a solution
2. the solution is unique
3. the solution's behaviour changes continuously with initial conditions

Subsonic Outflow (Left Boundary) - Example

Subsonic outflow: we should specify one variable

Alt	specified variable	well-posed	non-reflective
1	p	X	
2	ρu	X	
3	J^+	X	X

Subsonic Inflow 2D/3D



n	unit normal vector
v	fluid velocity at boundary

Subsonic inflow

Assumption:

$-a < \mathbf{v} \cdot \mathbf{n} < 0$

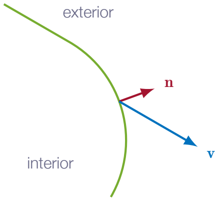
Four ingoing characteristics

One outgoing characteristic

Specify four variables at the boundary:

p_o, T_o, and flow direction (two angles)

Subsonic Outflow 2D/3D



$\mathbf{n}$	unit normal vector
$\mathbf{v}$	fluid velocity at boundary

Subsonic outflow

Assumption:

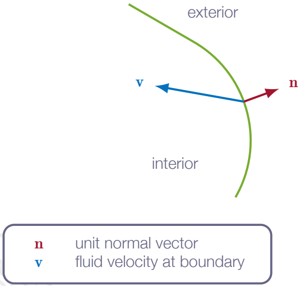
$0 < \mathbf{v} \cdot \mathbf{n} < a$

One ingoing characteristics

Four outgoing characteristic

Specify one variables at the boundary:
static pressure

Supersonic Inflow 2D/3D



Supersonic inflow

Assumption:

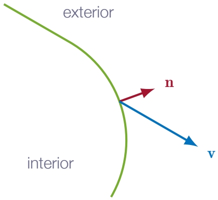
$\mathbf{v} \cdot \mathbf{n} < -a$

Five ingoing characteristics

No outgoing characteristics

Specify five variables at the boundary: solver variables

Supersonic Outflow 2D/3D



$\mathbf{n}$	unit normal vector
$\mathbf{v}$	fluid velocity at boundary

Supersonic outflow

Assumption:

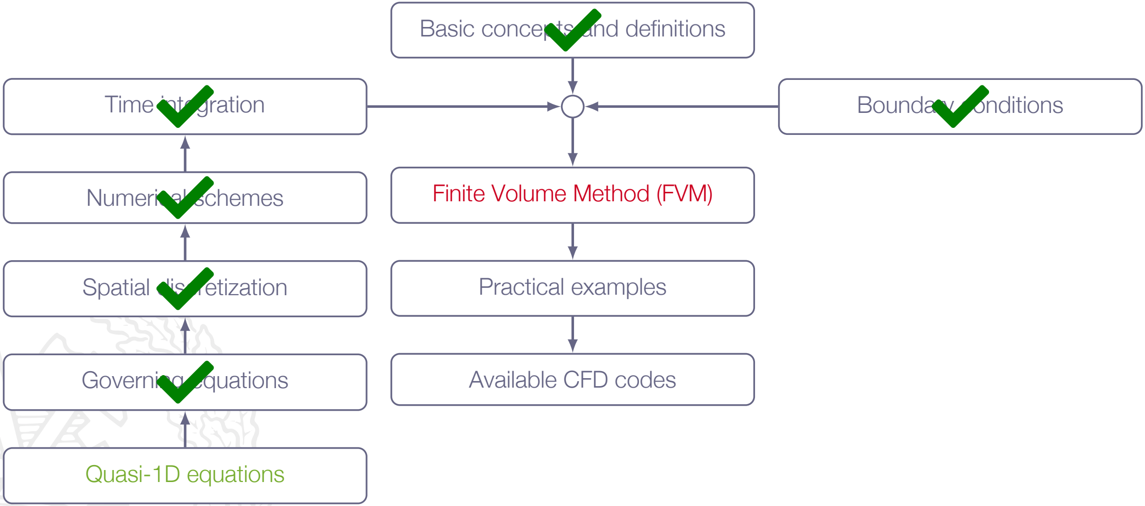
$\mathbf{v} \cdot \mathbf{n} > a$

No ingoing characteristics

Five outgoing characteristics

No variables specified at the boundary

Roadmap - The Time-Marching Technique



Explicit Finite-Volume Method - Summary

The described numerical approach can be categorized as

- Density-based
- Fully coupled
- Structured
- Explicit

with the following features

- High-order convective scheme
- Shock handling (artificial damping)

Explicit Finite-Volume Method - Summary

Spatial discretization:

Control volume formulations of conservation equations are applied to the cells of the discretized domain

Cell-averaged flow quantities $(\overline{\rho}, \overline{\rho u}, \overline{\rho e_o})$ are chosen as degrees of freedom

Flux terms are **approximated** in terms of the chosen degrees of freedom

- high-order, upwind type of flux approximation is used for optimum results

A **fully conservative** scheme is obtained

- the flux leaving one cell is identical to the flux entering the neighboring cell

The result of the spatial discretization is a system of ODEs

Explicit Finite-Volume Method - Summary

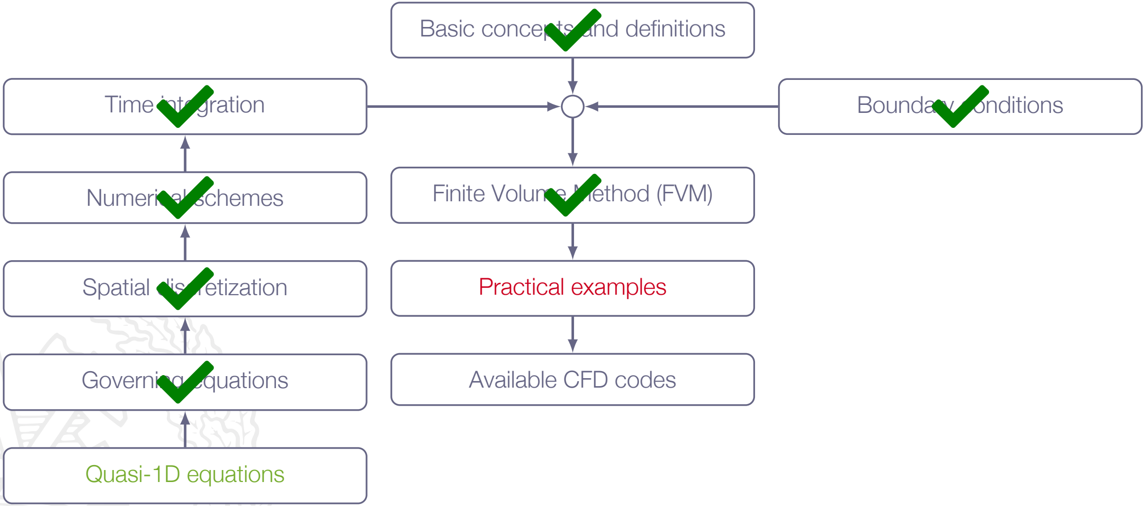
Time marching:

Three-stage, second-order accurate Runge-Kutta scheme

Explicit time-stepping

Time step length **limited by the** *CFL* **condition** ($CFL \leq 1$)

Roadmap - The Time-Marching Technique



Practical Examples: Grid Resolution and Numerical Schemes

Numerical Approach

Code: G3D::Flow (Chalmers in-house CFD code)

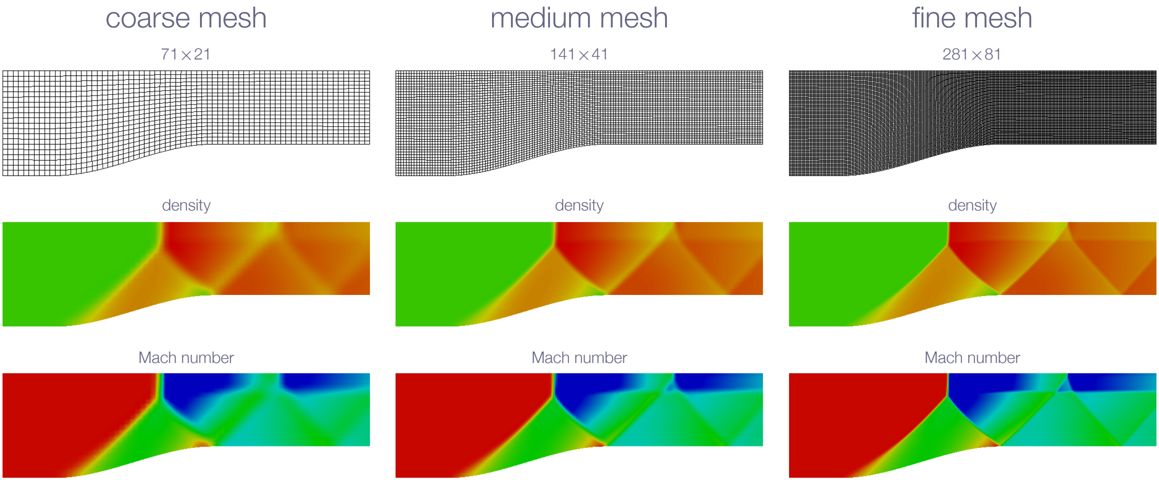
Finite-Volume Method

Three-stage, **second-order** accurate **Runge-Kutta** time stepping

First-order, second-order, and **third-order** characteristic upwinding scheme

Shock handling: TVD and artificial diffusion based on Jameson shock detection

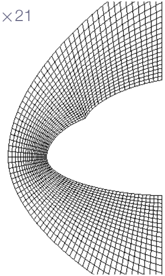
Grid Resolution: Compression Ramp



Grid Resolution: Space Shuttle

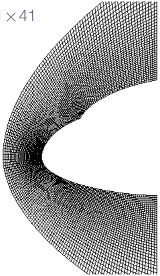
coarse mesh

81 × 21



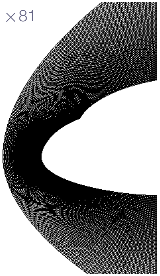
medium mesh

161 × 41

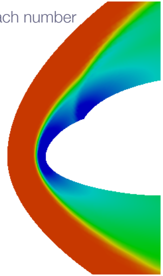


fine mesh

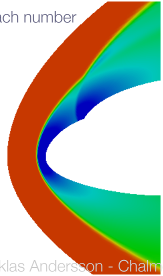
321 × 81



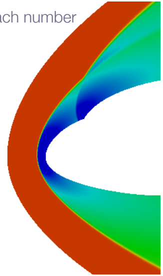
Mach number



Mach number



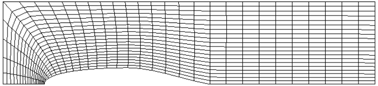
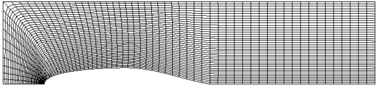
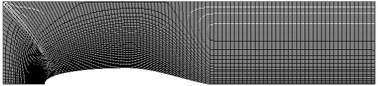
Mach number



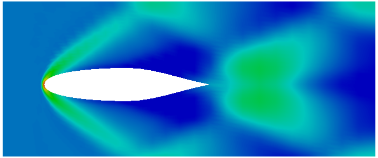
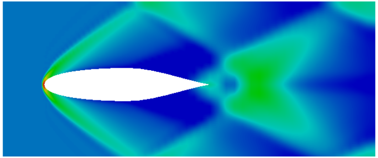
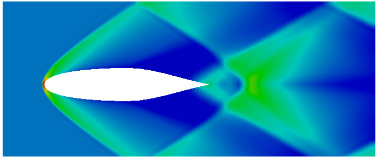
Grid Resolution: Axi-symmetric Slender Body

coarse mesh | medium mesh | fine mesh

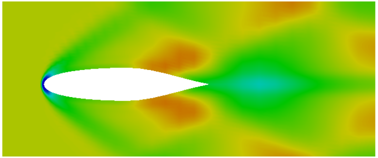
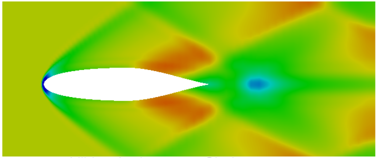
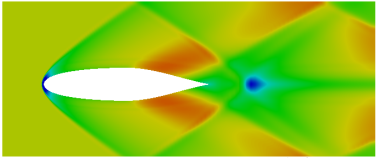
31×21 | 61×41 | 121×81

density | density | density

Mach number | Mach number | Mach number

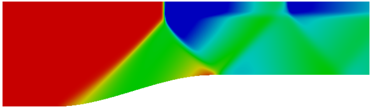




Numerical Scheme: Compression Ramp

first-order upwind

density

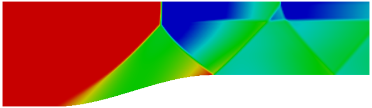
Mach number



second-order upwind

density

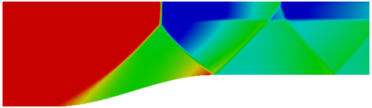
Mach number



third-order upwind

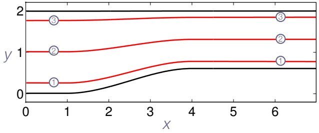
density

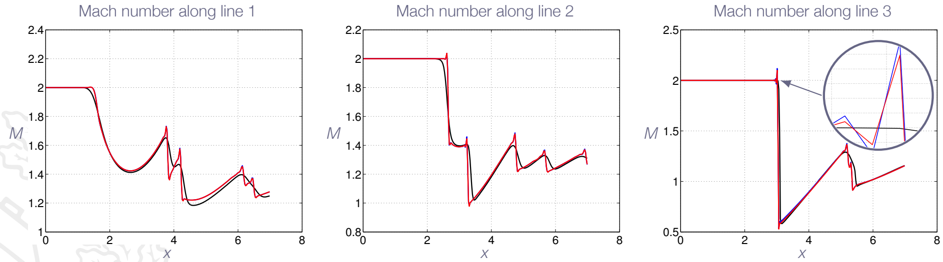
Mach number



Artificial Numerical Damping: Compression Ramp

Low artificial numerical damping



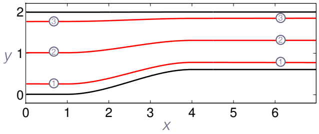


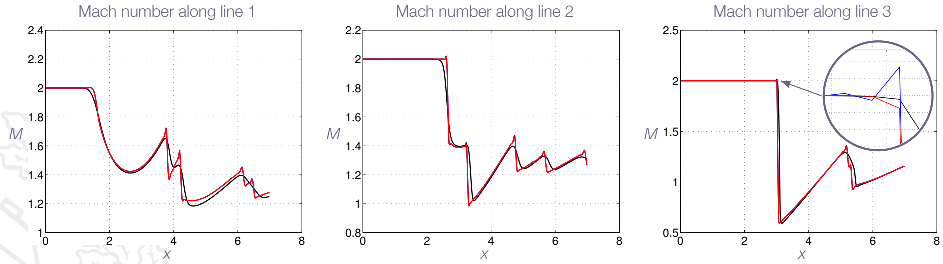
first-order upwind scheme
second-order upwind scheme
third-order upwind scheme

Artificial Numerical Damping: Compression Ramp

G3DFLOW

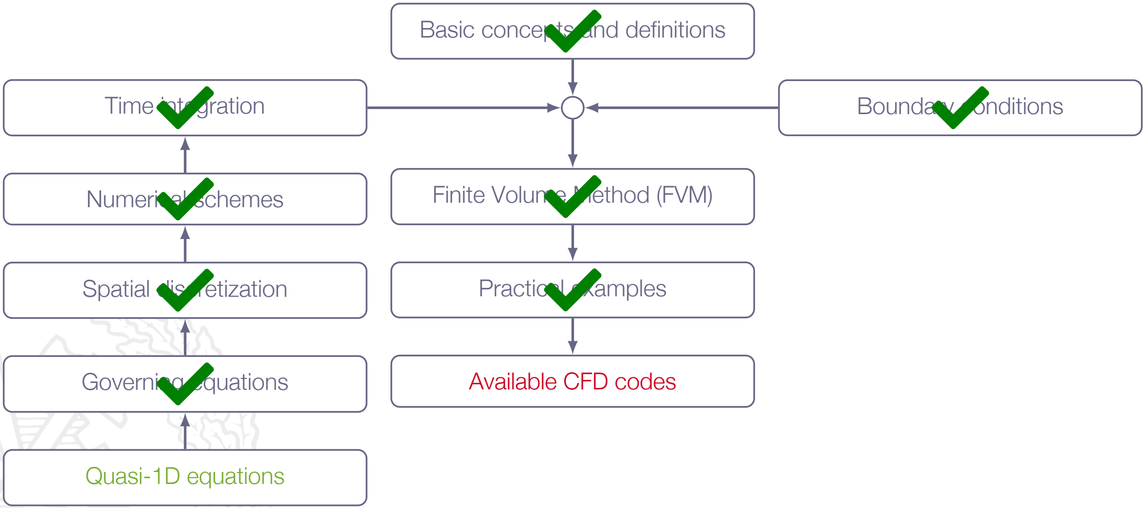
High artificial numerical damping





first-order upwind scheme
second-order upwind scheme
third-order upwind scheme

Roadmap - The Time-Marching Technique



Available CFD Codes

CFD Codes

List of free and commercial CFD codes:

http://www.cfd-online.com/Wiki/Codes

Free codes are in general unsupported and poorly documented

Commercial codes are often claimed to be suitable for all types of flows
The reality is that the user must make sure of this!

CFD Codes - General Guidlines

Simulation of high-speed and/or unsteady compressible flows:

Use correct solver options

otherwise you may obtain completely wrong solution!

1. coupled solver
2. equation of state
3. energy equation included

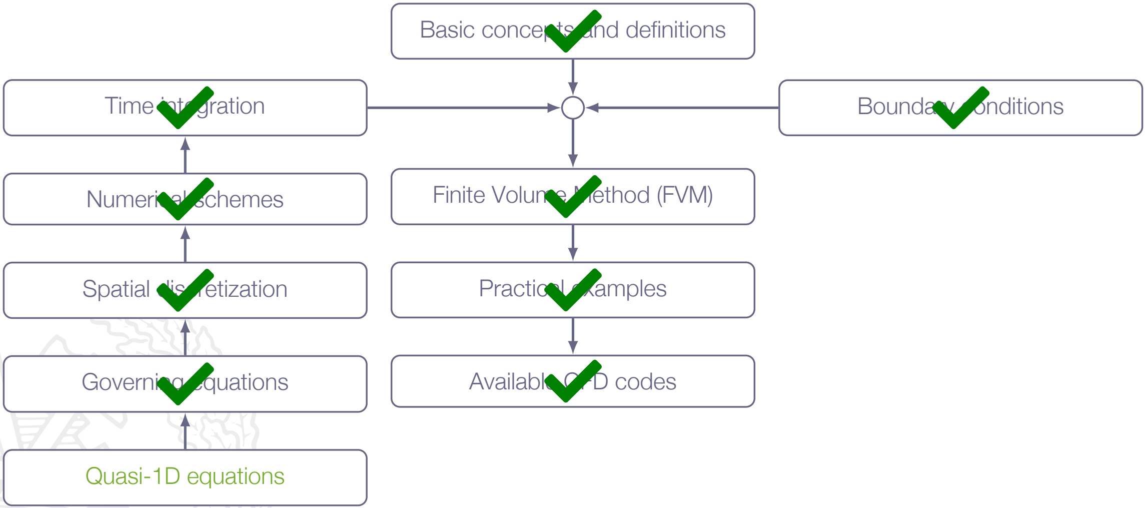
Use a high-quality grid

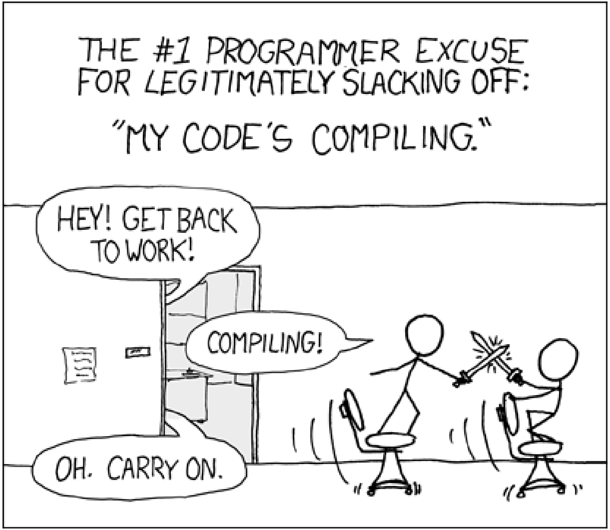
a poor grid will either not give you any solution at all (no convergence) or at best a very inaccurate solution!

ANSYS-FLUENT®/STAR-CCM+® - Typical Experiences

1. Very robust solvers - will almost always give you a solution
2. Accuracy of solution depends a lot on **grid quality**
3. **Shocks** are generally **smeared** more than in specialized codes
4. Solver is generally very **efficient** for **steady-state** problems
5. Solver is less efficient for truly unsteady problems, where both flow and acoustics must be resolved accurately

Roadmap - The Time-Marching Technique




THE #1 PROGRAMMER EXCUSE FOR LEGITIMATELY SLACKING OFF:
"MY CODE'S COMPILING."
HEY! GET BACK TO WORK!
COMPILING!
OH. CARRY ON.