Structure of the lectures

1. Fundamentals of finite volume methods
   - Shock-capturing schemes, higher-order methods
   - Discussion of mesh adaptation approaches

2. Structured AMR for hyperbolic problems
   - Presentation of all algorithmic components
   - Parallelization

3. Numerical methods for combustion research
   - Consideration of non-Cartesian geometries
   - Numerical methods for the inviscid reactive equations

4. Detonation simulation
   - Examples of ignition and detonation structure simulation
   - Extensions to viscous reactive equations

5. Fluid-structure interaction (FSI) simulation
   - Examples of detonation-driven FSI
   - Adaptive Lattice-Boltzmann method

Structure of the lectures - II

6. The AMROC software system
   - Practical implementation, discussion of other SAMR systems
   - AMROC code for examples from the lectures

7. Demo of AMROC
   - Installation on student computers
   - Running examples, etc.

8. Supplementary material
   - FV multigrid with SAMR codes

Useful references I

Finite volume methods for hyperbolic problems


Structured Adaptive Mesh Refinement


Useful references II

Combustion, detonations and shockwave theory

Shock-capturing schemes for combustion

Useful references III

Fluid-structure interaction and further applications (from my own work only)

Useful references IV

Implementation, parallelization

Useful references V

Supplementary: Adaptive multigrid (finite difference and finite element based in textbooks)
Lecture 1

Finite volume methods

Course Block-structured Adaptive Finite Volume Methods for Shock-Induced Combustion Simulation

Outline

Conservation laws
- Mathematical background
- Examples

Finite volume methods
- Basics of finite difference methods
- Splitting methods, second derivatives

Upwind schemes
- Flux-difference splitting
- Flux-vector splitting
- High-resolution methods

Meshes and adaptation
- Elements of adaptive algorithms
- Adaptivity on unstructured meshes
- Structured mesh refinement techniques

Mathematical background

$$\frac{\partial}{\partial t} q(x, t) + \sum_{n=1}^{d} \frac{\partial}{\partial x_n} f_n(q(x, t)) = s(q(x, t)), \quad D \subset \{(x, t) \in \mathbb{R}^d \times \mathbb{R}_+^M\}$$

$q = q(x, t) \in S \subset \mathbb{R}^M$ - vector of state, $f_n(q) \in C^1(S, \mathbb{R}^M)$ - flux functions,
$s(q) \in C^1(S, \mathbb{R}^M)$ - source term

Definition (Hyperbolicity)

$A(q, \nu) = \nu_1 A_1(q) + \cdots + \nu_d A_d(q)$ with $A_n(q) = \partial f_n(q)/\partial q$ has $M$ real eigenvalues $\lambda_1(q, \nu) \leq \cdots \leq \lambda_M(q, \nu)$ and $M$ linear independent right eigenvectors $r_n(q, \nu)$.

If $f_n(q)$ is nonlinear, classical solutions $q(x, t) \in C^1(D, S)$ do not generally exist, not even for $q_0(x) \in C^1(\mathbb{R}^d, S)$ [Majda, 1984], [Godlewski and Raviart, 1996], [Kröner, 1997]

Example: Euler equations
**Weak solutions**

Integral form (Gauss's theorem):
\[
\int_\Omega q(x, t + \Delta t) \, dx - \int_\Omega q(x, t) \, dx + \sum_{n=1}^{d} \int_{t}^{t+\Delta t} f_n(q(o(t))) \sigma_n(o) \, do \, dt = \int_\Omega s(q(x, t)) \, dx
\]

Theorem (Weak solution)
If \( q_0 \in L^\infty_{loc}(\mathbb{R}^d, S) \), \( q \in L^\infty_{loc}(D, S) \) is weak solution if \( q \) satisfies
\[
\int_0^t \int_{\mathbb{R}^d} \left[ \frac{\partial \varphi}{\partial t} \cdot q + \sum_{n=1}^{d} \frac{\partial \varphi}{\partial x_n} \cdot f_n(q) - \varphi \cdot s(q) \right] \, dx \, dt + \int_{\mathbb{R}^d} \varphi(x, 0) \cdot q_0(x) \, dx = 0
\]
for any test function \( \varphi \in C^1_0(D, S) \)

**Entropy solutions**

Select physical weak solution as \( \lim_{\varepsilon \to 0} q_\varepsilon = q \) almost everywhere in \( D \) of
\[
\frac{\partial q}{\partial t} + \sum_{n=1}^{d} \frac{\partial f_n(q)}{\partial x_n} - \varepsilon \sum_{n=1}^{d} \frac{\partial^2 q}{\partial x_n^2} = s(q), \quad x \in \mathbb{R}^d, \quad t > 0
\]

Theorem (Entropy condition)
Assume existence of entropy \( \eta \in C^1(D, \mathbb{R}) \) and entropy fluxes \( \psi_n \in C^1(D, \mathbb{R}) \) that satisfy
\[
\frac{\partial \eta(q)}{\partial t} + \sum_{n=1}^{d} \frac{\partial \psi_n}{\partial x_n}(q) \leq \frac{\partial \eta(q)}{\partial q} \cdot s(q)
\]
in the sense of distributions. Proof: [Godlewski and Raviart, 1996]

**Entropy solutions II**

Definition (Entropy solution)
Weak solution \( q \) is called an entropy solution if \( q \) satisfies
\[
\int_0^t \int_{\mathbb{R}^d} \left[ \frac{\partial \varphi}{\partial t} \cdot q + \sum_{n=1}^{d} \frac{\partial \varphi}{\partial x_n} \cdot f_n(q) - \varphi \cdot s(q) \right] \\
\eta(q_0(x)) \, dx \, dt + \int_{\mathbb{R}^d} \varphi(x, 0) \eta(q_0(x)) \, dx \geq 0
\]
for all entropy functions \( \eta(q) \) and all test functions \( \varphi \in C^1_0(D, \mathbb{R}^d), \varphi \geq 0 \)

Theorem (Jump conditions)
An entropy solution \( q \) is a classical solution \( q \in C^1(D, S) \) almost everywhere and satisfies the Rankine-Hugoniot (RH) jump condition
\[
(q^+ - q^-) \sigma_t + \sum_{n=1}^{d} (f_n(q^+) - f_n(q^-)) \sigma_n = 0
\]
and the jump inequality
\[
(\eta(q^+) - \eta(q^-)) \sigma_t + \sum_{n=1}^{d} (\psi_n(q^+) - \psi_n(q^-)) \sigma_n \leq 0
\]
along discontinuities. Proof: [Godlewski and Raviart, 1996]

**Examples**

Euler equations
\[
\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_n}(\rho u_n) = 0
\]
\[
\frac{\partial}{\partial t}(\rho u_k) + \frac{\partial}{\partial x_n}(\rho u_k u_n + \delta_{kn} p) = 0, \quad k = 1, \ldots, d
\]
with polytrope gas equation of state
\[
p = (\gamma - 1)(\rho E - \frac{1}{2} \rho u_n u_n)
\]
have structure
\[
\partial_t q(x, t) + \nabla \cdot f(q(x, t)) = 0
\]
Examples II

Navier-Stokes equations

\[ \partial_t p + \frac{\partial}{\partial x_k} (p u_k) = 0 \]
\[ \partial_t (p u_k) + \frac{\partial}{\partial x_k} (p u_k u_j + \delta_{kj} p - \tau_{kn}) = 0, \quad k = 1, \ldots, d \]
\[ \frac{\partial}{\partial t} (\rho E) + \frac{\partial}{\partial x_k} (u_k (\rho E + p) + q_n - \tau_{nj} u_j) = 0 \]

Type can be either hyperbolic or parabolic

Derivation

Assume \( \partial_t \mathbf{q} + \partial_x \mathbf{f} (\mathbf{q}) + \partial_x \mathbf{h} (\mathbf{q}, \partial_x \mathbf{q}) = \mathbf{s} (\mathbf{q}) \)

Time discretization \( t_n = n \Delta t \), discrete volumes \( l_j = [x_j - 1/2 \Delta x, x_j + 1/2 \Delta x] = [x_{j-1/2}, x_{j+1/2}] \)

Using approximations \( Q_j (t) \approx \frac{1}{|l_j|} \int_{l_j} q(x,t) \, dx \), \( s(Q_j (t)) \approx \frac{1}{|l_j|} \int_{l_j} s(q(x,t)) \, dx \)

and numerical fluxes \( \mathbf{F} (Q_j (t), Q_{j+1} (t)) \approx \mathbf{f} (q(x_{j+1/2}, t)) \), \( \mathbf{H} (Q_j (t), Q_{j+1} (t)) \approx \mathbf{h} (q(x_{j+1/2}, t), \nabla q(x_{j+1/2}, t)) \)

yields after integration (Gauss theorem)

\[ Q_j (t_{n+1}) = Q_j (t_n) - \frac{1}{\Delta t} \int_{t_n}^{t_{n+1}} \left[ \mathbf{F} (Q_j (t), Q_{j+1} (t)) - \mathbf{F} (Q_{j-1} (t), Q_j (t)) \right] dt - \]
\[ \frac{1}{\Delta x} \int_{t_n}^{t_{n+1}} \left[ \mathbf{H} (Q_j (t), Q_{j+1} (t)) - \mathbf{H} (Q_{j-1} (t), Q_j (t)) \right] dt + \int_{t_n}^{t_{n+1}} s(Q_j (t)) \, dt \]

For instance:

\[ Q_j^{n+1} - Q_j^n - \frac{\Delta t}{\Delta x} \left[ \mathbf{F} (Q_j^n, Q_{j+1}^n) - \mathbf{F} (Q_{j-1}^n, Q_j^n) \right] - \]
\[ \frac{\Delta t}{\Delta x} \left[ \mathbf{H} (Q_j^n, Q_{j+1}^n) - \mathbf{H} (Q_{j-1}^n, Q_j^n) \right] + \Delta t s(Q_j^n) \]

Outline

Conservation laws
- Mathematical background
- Examples

Finite volume methods
- Basics of finite difference methods
- Splitting methods, second derivatives

Upwind schemes
- Flux-difference splitting
- Flux-vector splitting
- High-resolution methods

Meshes and adaptation
- Elements of adaptive algorithms
- Adaptivity on unstructured meshes
- Structured mesh refinement techniques

Some classical definitions

\( (2s + 1) \)-point difference scheme of the form

\[ Q_j^{n+1} = \mathcal{H}^{(\Delta t)} (Q_{j-s}^n, \ldots, Q_{j+s}^n) \]

Definition (Stability)

For each time \( \tau \) there is a constant \( C_\tau \) and a value \( n_0 \in \mathbb{N} \) such that

\[ \| \mathcal{H}^{(\Delta t)} (Q^n) \| \leq C_\tau \] for all \( n \Delta t \leq \tau \), \( n < n_0 \)

Definition (Consistency)

If the local truncation error

\[ \mathcal{L}^{(\Delta t)} (\cdot, t) := \frac{1}{\Delta t} \left[ q(x, t + \Delta t) - \mathcal{H}^{(\Delta t)} (q(\cdot, t)) \right] \]

satisfies \( \| \mathcal{L}^{(\Delta t)} (\cdot, t) \| \to 0 \) as \( \Delta t \to 0 \)

Definition (Convergence)

If the global error \( \mathcal{E}^{(\Delta t)} (x, t) := Q(x, t) - q(x, t) \) satisfies \( \| \mathcal{E}^{(\Delta t)} (\cdot, t) \| \to 0 \) as \( \Delta t \to 0 \) for all admissible initial data \( q_0 (x) \)
Some classical definitions II

Definition (Order of accuracy)
\( \mathcal{H}(\cdot) \) is accurate of order \( o \) if for all sufficiently smooth initial data \( q_0(x) \), there is a constant \( C_o \), such that the local truncation error satisfies
\[ ||\mathcal{L}^{(o)}(\cdot, t)|| \leq C_o \Delta t^o \text{ for all } \Delta t < \Delta t_0, \ t \leq \tau \]

Definition (Conservative form)
If \( \mathcal{H}(\cdot) \) can be written in the form
\[ Q_i^{n+1} = Q_i^n - \frac{\Delta t}{\Delta x} (F_i^{n-1} - F_i^{n+1}) \]

A conservative scheme satisfies
\[ \sum_{j \in \Omega} Q_j^{n+1} = \sum_{j \in \Omega} Q_j^n \]

Definition (Consistency of a conservative method)
If the numerical flux satisfies \( F_i(q, \ldots, q) = f(q) \) for all \( q \in S \)

**Splitting methods**

Solve homogeneous PDE and ODE successively!
\[ \mathcal{H}((\Delta t)) : \quad \partial_t q + \nabla \cdot f(q) = 0, \quad IC: Q(t_m) \xrightarrow{\Delta t} \tilde{Q} \]
\[ S((\Delta t)) : \quad \partial_t q = s(q), \quad IC: \tilde{Q} \xrightarrow{\Delta t} Q(t_m + \Delta t) \]

1st-order Godunov splitting: \( Q(t_m + \Delta t) = S^{((\Delta t))} \mathcal{H}^{((\Delta t))}(Q(t_m)) \)
2nd-order Strang splitting: \( Q(t_m + \Delta t) = S^{(\frac{1}{2} \Delta t)} \mathcal{H}^{((\Delta t))} S^{(\frac{1}{2} \Delta t)}(Q(t_m)) \)

1st-order dimensional splitting for \( \mathcal{H}(\cdot) \):
\[ \mathcal{H}^{(1)}: \quad \partial_t q + \partial_x f_1(q) = 0, \quad IC: Q(t_m) \xrightarrow{\Delta t} \tilde{Q}_1^{1/2} \]
\[ \mathcal{H}^{(2)}: \quad \partial_t q + \partial_x f_2(q) = 0, \quad IC: \tilde{Q}_1^{1/2} \xrightarrow{\Delta t} \tilde{Q} \]

[Toro, 1999]

Conservative scheme for diffusion equation
Consider \( \partial_t q - c\Delta q = 0 \) with \( c \in \mathbb{R}^+ \), which is readily discretized as
\[ Q_i^{n+1} = Q_i^n + c \frac{\Delta t}{\Delta x_1} (Q_i^{n-1} + 2Q_i^n - Q_{i+1}^n - Q_{i-1}^n) \]

or conservatively
\[ Q_i^{n+1} = Q_i^n + c \frac{\Delta t}{\Delta x_1} \left( H_i^{n+1} - H_i^{n-1} \right) \]

Von Neumann stability analysis: Insert single eigenmode \( \tilde{Q}(t) e^{ikx} e^{\imath \xi x} \) into discretization
\[ \tilde{Q}^{n+1} = \tilde{Q}^n + C_1 \left( \tilde{Q}^n e^{ik\Delta x_1} - 2\tilde{Q}^n + \tilde{Q}^n e^{-ik\Delta x_1} \right) + C_2 \left( \tilde{Q}^n e^{ik\Delta x_2} - 2\tilde{Q}^n + \tilde{Q}^n e^{-ik\Delta x_2} \right) \]

with \( C_1 = \frac{\Delta t}{\Delta x_1} \), \( C_2 = \frac{\Delta t}{\Delta x_2} \), \( \imath = 1, 2 \), which gives after inserting \( e^{ikx_i} = \cos(k_i x_i) + \imath \sin(k_i x_i) \)
\[ \tilde{Q}^{n+1} = \tilde{Q}^n (1 + 2C_1 (\cos(k_1 \Delta x_1) - 1) + 2C_2 (\cos(k_2 \Delta x_2) - 1)) \]

Stability requires
\[ \left| 1 + 2C_1 (\cos(k_1 \Delta x_1) - 1) + 2C_2 (\cos(k_2 \Delta x_2) - 1) \right| \leq 1 \]

i.e.
\[ |1 - 4C_1| \leq |4C_2| \leq 1 \]

from which we derive the stability condition
\[ 0 \leq c \left( \frac{\Delta t}{\Delta x_1^2} + \frac{\Delta t}{\Delta x_2^2} \right) \leq \frac{1}{2} \]

Linear upwind schemes
Consider Riemann problem
\[ \frac{\partial}{\partial t} q(x,t) + \frac{\partial}{\partial x} f(q(x,t)) = 0, \quad x \in \mathbb{R}, \quad t > 0 \]

Has exact solution
\[ q(x,t) = q_L + \sum_{\lambda_m \leq x/t} a_{m\lambda_m} f_{\lambda_m} = q_R - \sum_{\lambda_m \geq x/t} a_{m\lambda_m} f_{\lambda_m} \]

Use Riemann problem to evaluate numerical flux \( F(q_L, q_R) := f(q(0,t)) = A q(0,t) \) as
\[ F(q_L, q_R) = A q_L + \sum_{\lambda_m \leq \lambda_m^*} a_{m\lambda_m} f_{\lambda_m} = A q_R - \sum_{\lambda_m \geq \lambda_m^*} a_{m\lambda_m} f_{\lambda_m} \]

Use \( \lambda_m^* = \max(\lambda_m, 0) \), \( \lambda_m = \min(\lambda_m, 0) \)
to define
\[ \Lambda^+ := \text{diag}(\lambda^+_1, \ldots, \lambda^+_M), \quad \Lambda^- := \text{diag}(\lambda^-_1, \ldots, \lambda^-_M) \]

and
\[ \Lambda^+ := R \Lambda^+ R^{-1}, \quad \Lambda^- := R \Lambda^- R^{-1} \]

which gives
\[ F(q_L, q_R) = A q_L + \Lambda^- \Delta q = A q_R - (\Lambda^- + \Lambda^+) q_L + \Lambda^- q_R \]

with \( \Delta q := q_R - q_L \)
Flux difference splitting

Godunov-type scheme with $\Delta Q^p_{j+1/2} = Q^p_{j+1} - Q^p_j$

$$Q^p_{j+1} = Q^p_j - \frac{\Delta t}{\Delta x} \left( \hat{A}^{-}(Q^p_j) \Delta Q^p_{j+1/2} + \hat{A}^{+}(Q^p_{j-1}) \Delta Q^p_{j-1/2} \right)$$

Use linearization $\tilde{f}(q) = \hat{A}(q_L, q_R)q$ and construct scheme for nonlinear problem as

$$Q^{n+1}_{j} = Q^n_j - \frac{\Delta t}{\Delta x} \left( \hat{A}^{-}(Q^n_j, Q^{n+1}_{j+1}) \Delta Q^n_{j+1/2} + \hat{A}^{+}(Q^n_{j-1}, Q^n_j) \Delta Q^n_{j-1/2} \right)$$

stability condition

$$\max_{j \in \mathbb{Z}} |\hat{\lambda}_{m,j+1/2}| \frac{\Delta t}{\Delta x} \leq 1 \quad \text{for all } m = 1, \ldots, M$$

[LeVeque, 1992]

Roe’s approximate Riemann solver

Choosing $\hat{A}(q_L, q_R)$ [Roe, 1981]:

(i) $\hat{A}(q_L, q_R)$ has real eigenvalues

(ii) $\hat{A}(q_L, q_R) = \frac{\partial f(q)}{\partial q}$ as $q_L, q_R \to q$

(iii) $\hat{A}(q_L, q_R) \Delta q = f(q_R) - f(q_L)$

For Euler equations:

$$\hat{\rho} = \sqrt{\rho_L \rho_R} + \sqrt{\rho_L \rho_L} \quad \text{and} \quad \hat{\nu} = \sqrt{\nu_L \nu_R} + \sqrt{\nu_L \nu_L} \quad \text{for } \nu = u_n, H$$

Wave decomposition: $\Delta q = q_L - q_R = \sum m \hat{\lambda}_m \hat{a}_m \hat{r}_m$

$$F(q_L, q_R) = f(q_L) + \sum_{\hat{\lambda}_m < 0} \hat{\lambda}_m \hat{a}_m \hat{r}_m = f(q_R) - \sum_{\hat{\lambda}_m \geq 0} \hat{\lambda}_m \hat{a}_m \hat{r}_m$$

$$= \frac{1}{2} \left( f(q_L) + f(q_R) - \sum m \hat{\lambda}_m \hat{a}_m \hat{r}_m \right)$$

Harten-Lax-Van Leer (HLL) approximate Riemann solver

Flux vector splitting

Splitting

$$f(q) = f^+(q) + f^-(q)$$

derived under restriction $\hat{\lambda}_m \geq 0$ and $\hat{\lambda}_m \leq 0$ for all $m = 1, \ldots, M$ for

$$\hat{A}^+(q) = \frac{\partial f^+(q)}{\partial q} \quad \hat{A}^-(q) = \frac{\partial f^-(q)}{\partial q}$$

plus reproduction of regular upwinding

$$f^+(q) = f(q) \quad f^-(q) = 0 \quad \text{if } \lambda_m \geq 0 \quad \text{for all } m = 1, \ldots, M$$

$$f^+(q) = 0 \quad f^-(q) = f(q) \quad \text{if } \lambda_m \leq 0 \quad \text{for all } m = 1, \ldots, M$$

Then use

$$F(q_L, q_R) = f^+(q_L) + f^-(q_R)$$

[Toro, 1999], HLLC: [Toro et al., 1994]
Steger-Warming

Required \( f(q) = A(q)q \)

\[
\lambda^+_m = \frac{1}{2} (\lambda_m + |\lambda_m|) \quad \lambda^-_m = \frac{1}{2} (\lambda_m - |\lambda_m|)
\]

\[
A^+(q) := R(q) \Lambda^+(q) R^{−1}(q) \quad A^−(q) := R(q) \Lambda^−(q) R^{−1}(q)
\]

Gives

\[
f(q) = A^+(q)q + A^−(q)q
\]

and the numerical flux

\[
F(q_L, q_R) = A^+(q_L)q_L + A^−(q_R)q_R
\]

Jacobians of the split fluxes are identical to \( A^\pm(q) \) only in linear case

\[
\frac{\partial f^\pm(q)}{\partial q} = \frac{\partial (A^\pm(q)q)}{\partial q} = A^\pm(q) + \frac{\partial A^\pm(q)}{\partial q} q
\]

Further methods: Van Leer FVS [Toro, 1999], AUSM [Wada and Liou, 1997]

High-resolution methods

Objective: Higher-order accuracy in smooth solution regions but no spurious oscillations near large gradients

Consistent monotone methods converge toward the entropy solution, but

**Theorem**

A monotone method is at most first order accurate.

Proof: [Harten et al., 1976]

**Definition (TVD property)**

Scheme \( T^{\text{(L)}}(Q^n) \) TVD if \( TV(Q^{n+1}) \leq TV(Q^n) \) is satisfied for all discrete sequences \( Q^n \). Herein, \( TV(Q^n) := \sum_{j \in Z} |Q^{n+1}_j - Q^n_j| \).

TVD schemes: no new extrema, local minima are non-decreasing, local maxima are non-increasing (termed *monotonicity-preserving*). Monotonicity-preserving higher-order schemes are at least 5-point methods. Proofs: [Harten, 1983]

TVD concept is proven [Godlewski and Raviart, 1996] for scalar schemes only but nevertheless used to construct high resolution schemes. Monotonicity-preserving scheme can converge toward non-physical weak solutions.

**MUSCL slope limiting**

Monotone Upwind Schemes for Conservation Laws [van Leer, 1979]

\[
\begin{align*}
Q^+_{j+1/2} &= Q^+_j + \frac{1}{4} \left[ (1 - \omega) \Phi^+_j \Delta_j + (1 + \omega) \Phi^-_j \Delta_j \right] \\
Q^-_{j+1/2} &= Q^-_j - \frac{1}{4} \left[ (1 - \omega) \Phi^+_j \Delta_j + (1 + \omega) \Phi^-_j \Delta_j \right]
\end{align*}
\]

with \( \Delta_{j+1/2} = Q^+_{j+1} - Q^-_{j+1} \) and slope limiters, e.g., Minmod

\[
\Phi(r) = \max(0, \min(r, 1))
\]

Using a midpoint rule for temporal integration, e.g.,

\[
Q^* = Q^n - \frac{1}{2} \Delta t \left( F(Q^{n+1}_j, Q^+_j) - F(Q^n_j, Q^-_j) \right)
\]

and constructing limited values from \( Q^* \) to be used in FV scheme gives a TVD method if

\[
\frac{1}{2} \left[ (1 - \omega) \Phi(r) + (1 + \omega) r \Phi \left( \frac{1}{r} \right) \right] < \min(2, 2r)
\]

is satisfied for \( r > 0 \). Proof: [Hirsch, 1988]

**Wave Propagation with flux limiting**

Wave Propagation Method [LeVeque, 1997] is built on the flux differencing approach

\[
A^\pm \Delta := \bar{A}^\pm(q_j, q_R) \Delta q \quad \text{and the waves} \quad \mathcal{V}_m := \eta_m \mathcal{V}_m, \quad \mathcal{A}^\pm \Delta q = \sum_{\lambda_m < 0} \lambda_m \mathcal{V}_m
\]

Wave Propagation 1D:

\[
Q^{n+1} = Q^n + \frac{\Delta t}{\Delta x} \left( A^- \Delta_j + A^+ \Delta_{j+1} \right) - \frac{\Delta t}{\Delta x} \left( \bar{F}_{j+1/2} - \bar{F}_{j-1/2} \right)
\]

with

\[
\bar{F}_{j+1/2} = \frac{1}{2} |A| \left[ 1 - \frac{\Delta t}{\Delta x} |A| \right] \Delta_{j+1/2} = \frac{1}{2} \sum_{m=1}^{M} |\lambda|^2 \left( 1 - \frac{\Delta t}{\Delta x} \lambda_{j+1/2} \right) \mathcal{V}_m
\]

and wave limiter

\[
\mathcal{V}_{\text{lim}}^{\pm \Delta}_j = \Phi(\Theta^m_{j+1/2}) \mathcal{V}_m
\]

with

\[
\Theta^m_{j+1/2} = \left\{ \begin{array}{ll}
\eta_{m-1} & \quad \lambda_{m-1} \geq 0, \\
\eta_{m+1} & \quad \lambda_{m+1} < 0
\end{array} \right.
\]
Wave Propagation Method in 2D

Writing \( \bar{\lambda} = \bar{A}^{+} \Delta j_{j+1/2} \) one can develop a truly two-dimensional one-step method [Langseth and LeVeque, 2000]

\[
\begin{align*}
Q_{j+1/2}^{n+1} &= \frac{\Delta t}{\Delta x_1} \left( \bar{A}^{-} \Delta j_{j+1/2} - \frac{\Delta t}{2 \Delta x_2} \left[ A^{-} \bar{B}^{-} \Delta j_{j+1,k+1/2} + A^{-} \bar{B}^{+} \Delta j_{j+1,k-1/2} \right] + \\
&\quad - \frac{\Delta t}{2 \Delta x_1} \left[ B^{-} \bar{A}^{-} \Delta j_{j,k+1} + B^{-} \bar{A}^{+} \Delta j_{j,k+1/2} \right] \\
&\quad + \frac{\Delta t}{2 \Delta x_1} \left[ B^{+} \bar{A}^{-} \Delta j_{j,k-1} + B^{+} \bar{A}^{+} \Delta j_{j,k-1/2} \right] \right)
\end{align*}
\]

that is stable for

\[
\left\{ \max_{j \in \mathbb{Z}} |\bar{\lambda}_m,j+1/2| \frac{\Delta t}{\Delta x_1}, \max_{k \in \mathbb{Z}} |\bar{\lambda}_m,j+1/2| \frac{\Delta t}{\Delta x_2} \right\} \leq 1, \quad \text{for all } m = 1, \ldots, M
\]

Further high-resolution methods

Some further high-resolution methods (good overview in [Laney, 1998]):

- FCT: 2nd order [Oran and Boris, 2001]
- ENO/WENO: 3rd order [Shu, 97]
- PPM: 3rd order [Colella and Woodward, 1984]

3rd order methods must make use of strong-stability preserving Runge-Kutta methods [Gottlieb et al., 2001] for time integration that use a multi-step update

\[
\bar{Q}_j^0 = \alpha_0 Q_j^n + \beta_0 \bar{Q}_j^{n-1} + \gamma_0 \frac{\Delta t}{\Delta x} \left( F_{j+1/2} \bar{Q}_j^{n-1} - F_{j-1/2} \bar{Q}_j^{n-1} \right)
\]

with \( \bar{Q}_j^0 := Q_j^n, \alpha_1 = 1, \beta_1 = 0; \) and \( Q_j^{n+1} := \bar{Q}^T \) after final stage \( T \)

Typical storage-efficient SSPRK(3,3):

\[
\bar{Q}_j^T = Q_j^n + \Delta t F(Q_j^n), \quad \bar{Q}_j^2 = \frac{3}{4} Q_j^n + \frac{1}{4} \bar{Q}_j^1 + \frac{1}{4} \Delta t F(\bar{Q}_j^1),
\]

\[
Q_j^{n+1} = \frac{1}{3} Q_j^n + \frac{2}{3} \bar{Q}_j^1 + \frac{2}{3} \Delta t F(\bar{Q}_j^1)
\]
Adaptivity on unstructured meshes

- Coarse cells replaced by finer ones
- Global time-step
- Cell-based data structures
- Neighborhoods have to stored
  + Geometric flexible
  + No hanging nodes
  + Easy to implement
    - Higher order difficult to achieve
    - Cell aspect ratio must be considered
    - Fragmented data
    - Cache-reuse / vectorization nearly impossible
  - Complex load-balancing
  - Complex synchronization

Structured mesh refinement techniques

- Block-based data of equal size
- Block stored in a quad-tree
- Time-step refinement
- Global index coordinate system
- Neighborhoods need not be stored
  + Numerical scheme only for single regular block necessary
  + Easy to implement
  + Simple load-balancing
  + Parent/Child relations according to tree
  +/- Cache-reuse / vectorization only in data block

Block-structured adaptive mesh refinement (SAMR)

- Refined block overlay coarser ones
- Time-step refinement
- Block (aka patch) based data structures
- Global index coordinate system
- Numerical scheme only for single patch necessary
- Efficient cache-reuse / vectorization possible
- Simple load-balancing
- Minimal synchronization overhead
  - Cells without mark are refined
  - Hanging nodes unavailable
  - Cluster-algorithm necessary
  - Difficult to implement

References


References II


Lecture 2

**The SAMR method for hyperbolic problems**

Course *Block-structured Adaptive Finite Volume Methods for Shock-Induced Combustion Simulation*

Outline

The serial Berger-Colella SAMR method
- Block-based data structures
- Numerical update
- Conservative flux correction
- Level transfer operators
- The basic recursive algorithm
- Cluster algorithm
- Refinement criteria

Parallel SAMR method
- Domain decomposition
- A parallel SAMR algorithm
- Partitioning

Examples
- Euler equations
**Outline**

The serial Berger-Colella SAMR method

- Block-based data structures
- Numerical update
- Conservative flux correction
- Level transfer operators
- The basic recursive algorithm
- Cluster algorithm
- Refinement criteria

Parallel SAMR method

- Domain decomposition
- A parallel SAMR algorithm
- Partitioning

Examples

- Euler equations

**Examples**

- Euler equations

**References**

- Serial SAMR method
- Parallel SAMR method
- Examples
- References

---

**Refinement data**

- Resolution: $\Delta t_l := \frac{\Delta t_l-1}{r_l}$ and $\Delta x_{n,l} := \frac{\Delta x_{n,l-1}}{r_l}$
- Refinement factor: $r_l \in \mathbb{N}, r_l \geq 2$ for $l > 0$ and $r_0 = 1$
- Integer coordinate system for internal organization [Bell et al., 1994]:
  $$\Delta x_{n,l} \approx \prod_{k=1}^{l_{max}} r_k$$
- Computational Domain: $G_0 = \bigcup_{m=1}^{M_0} G_{0,m}$
- Domain of level $l$: $G_l := \bigcup_{m=1}^{M_l} G_{l,m}$ with $G_{l,m} \cap G_{l,n} = \emptyset$ for $m \neq n$
- Refinements are properly nested: $G_l^k \subset G_{l-1}$
- Assume a FD scheme with stencil radius $s$. Necessary data:
  - Vector of state: $Q^l := \bigcup_m Q(G_{l,m})$
  - Numerical fluxes: $F^{n,l} := \bigcup_m F^n(G_{l,m})$
  - Flux corrections: $\delta F^{n,l} := \bigcup_m \delta F^n(G_{l,m})$

**Setting of ghost cells**

- Notations:
  - Boundary: $\partial G_{l,m}$
  - Hull: $\bar{G}_{l,m} = G_{l,m} \cup \partial G_{l,m}$
  - Ghost cell region: $\tilde{G}_{l,m} = G_{l,m}^\text{int} \setminus \bar{G}_{l,m}$

- Complete grid with ghost cells - $G_{l,m}^\text{ghost}$
- Interior grid with buffer cells - $G_{l,m}^\text{int}$
- Synchronization with $G_l - \tilde{S}_{l,m}^p = \bar{G}_{l,m}^p \cap G_l$
- Physical boundary conditions - $\tilde{P}_{l,m}^p = \bar{G}_{l,m}^p \setminus G_0$
- Interpolation from $G_{l-1} - \tilde{T}_{l,m}^p = \bar{G}_{l,m}^p \setminus \tilde{S}_{l,m}^p \cup \tilde{P}_{l,m}^p$
**Numerical update**

Time-explicit conservative finite volume scheme

\[
\gamma(t) : Q_{jk}(t+\Delta t) = Q_{jk}(t) - \frac{\Delta t}{\Delta x_1} \left( F^{j+\frac{1}{2},k} - F^{j-\frac{1}{2},k} \right) - \frac{\Delta t}{\Delta x_2} \left( F^{jk+\frac{1}{2}} - F^{jk-\frac{1}{2}} \right)
\]

**Update Level \( l \)**

For all \( m = 1 \) to \( M_l \), do

\[
Q(G_{l,m}, t) : \gamma'(Q_{j,l+1}) = Q(G_{l,m}, t + \Delta t) \cdot F^n(\tilde{G}_{l,m}, t)
\]

If level \( l > 0 \)

- Add \( F^n(\partial G_{l,m}, t) \) to \( \delta F^n \)

If level \( l + 1 \) exists

- Init \( \delta F^{n,l+1} \) with \( F^n(\tilde{G}_{l,m} \cap \partial G_{l+1}, t) \)

**Conservative flux correction**

Example: Cell \( j, k \)

\[
\bar{Q}_{jk}(t + \Delta t) = Q_{jk}(t) - \frac{\Delta t}{\Delta x_1} \left( F^{j+\frac{1}{2},k} - F^{j-\frac{1}{2},k} \right) - \frac{\Delta t}{\Delta x_2} \left( F^{jk+\frac{1}{2}} - F^{jk-\frac{1}{2}} \right)
\]

**Correction pass:**

1. \( \delta F^{j+1/2} := -F^{j+1/2} \)

2. \( \delta F^{j+1/2} := \delta F^{j+1/2} + \frac{1}{r_{l+1}} \sum_{i=0}^{\eta_{l+1}-1} F^{i+1/2}_{l+1} (t + \kappa \Delta t_{l+1}) \)

3. \( \bar{Q}_{jk}(t + \Delta t) := \bar{Q}_{jk}(t + \Delta t) + \frac{\Delta t}{\Delta x_1} \delta F^{jk+1/2} \)

**Conservative flux correction II**

- Level \( l \) cells needing correction \( (G^{i+1}_{l+1} \cap G_l) \cap G_l \)
- Corrections \( \delta F^{n,l+1} \) stored on level \( l + 1 \) along \( \partial G_{l+1} \) (lower-dimensional data coarsened by \( r_{l+1} \))
- Init \( \delta F^{n,l+1} \) with level \( l \) fluxes \( F^n(\tilde{G}_l \cap \partial G_{l+1}) \)
- Add level \( l + 1 \) fluxes \( F^{n,l+1}(\partial G_{l+1}) \) to \( \delta F^{n,l+1} \)

**Cells to correct**

- \( F^n \)
- \( F^{n,l+1} \)
- \( \delta F^{n,l+1} \)

**Level transfer operators**

Conservative averaging (restriction):

Replace cells on level \( l \) covered by level \( l + 1 \), i.e. \( G_l \cap G_{l+1} \), by

\[
\bar{Q}_{jk} := \frac{1}{(r_{l+1})^2} \sum_{i=0}^{\eta_{l+1}-1} \sum_{j=0}^{\eta_{l+1}-1} Q_{i+1,l,k+1} \delta x^{i+1}_{l+1} \delta x^{j+1}_{l+1}
\]

Bilinear interpolation (prolongation):

\[
\bar{Q}^{i+1}_{j+1} := (1 - f_1)(1 - f_2) Q^{i-1,j-1} + f_1(1 - f_2) Q^{i+1,j-1} + (1 - f_1)f_2 Q^{i+1,j+1} + f_1 f_2 Q^{i+1,j}
\]

with factors \( f_1 := \frac{x_j - x_{j+1}^{i-1}}{\delta x_{1,l}} \), \( f_2 := \frac{x_k - x_{k+1}^{j-1}}{\delta x_{2,l}} \) derived from the spatial coordinates of the cell centers \( (x_{1,l,j,k}^{i,j}) \) and \( (x_{1,l,j,k}^{i+1,j+1}) \).

For boundary conditions on \( \tilde{l} \):

Linear time interpolation

\[
\bar{Q}^{i+1}_{j+1}(t + \kappa \Delta t_{l+1}) := \left( 1 - \frac{\kappa}{r_{l+1}} \right) \bar{Q}^{i+1}_{j+1}(t) + \frac{\kappa}{r_{l+1}} \bar{Q}^{i+1}_{j+1}(t + \Delta t_{l+1}) \quad \text{for } \kappa = 0, \ldots, r_{l+1}
The basic recursive algorithm

Recursive integration order

- Space-time interpolation of coarse data to set $I_l^i, l > 0$
- Regridding:
  - Creation of new grids, copy existing cells on level $l > 0$
  - Spatial interpolation to initialize new cells on level $l > 0$

\[ BERGER \ and \ COLELLA, \ 1988 \]
\[ BERGER \ and \ OLiger, \ 1984 \]

Regridding algorithm

Regrid($l$) - Regrid all levels $i > l$

For $i = l_f$ Down to $l$ Do
  Flag $N^i$ according to $Q^i(t)$
  If level $i+1$ exists?
    Flag $N^i$ below $G^{i+2}$
    Flag buffer zone on $N^i$
    Generate $G^{i+1}$ from $G^i$
    $\tilde{G}_i := G_i$
  For $i = l$ To $l_f$ Do
    $C\tilde{G}_i := G_0 \backslash \tilde{G}_i$
    $\tilde{G}_{i+1} := \tilde{G}_{i+1} \backslash C\tilde{G}_i$

Recompose($l$)

- Refinement flags:
  $N^i := \bigcup_m N(\partial G_{i,m})$
- Activate flags below higher levels
- Flag buffer cells of $b > \kappa_f$ cells,$\kappa_f$ steps between calls of
  Regrid($l$)
- Special cluster algorithm
- Use complement operation to ensure proper nesting condition

Recomposition of data

Recompose($l$) - Reorganize all levels $i > l$

For $i = l_f + 1$ To $l_f + 1$ Do
  Interpolate $Q^{-1}(t)$ onto $\tilde{Q}^i(t)$
  Copy $Q^i(t)$ onto $\tilde{Q}^i(t)$
  Set ghost cells of $\tilde{Q}^i(t)$
  $Q^i(t) := \tilde{Q}^i(t)$, $G_i := \tilde{G}_i$
- Creates max. 1 level above $l_f$, but can remove multiple level if $\tilde{G}_i$
  empty (no coarsening!)
- Use spatial interpolation on entire data $\tilde{Q}^i(t)$
- Overwrite where old data exists
- Synchronization and physical boundary conditions
### Clustering by signatures

<table>
<thead>
<tr>
<th>T</th>
<th>Flagged cells per row/column</th>
</tr>
</thead>
<tbody>
<tr>
<td>Δ</td>
<td>Second derivative of T, Δ = T_{n+1} - 2T_n + T_{n-1}</td>
</tr>
</tbody>
</table>

Technique from image detection: [Bell et al., 1994], see also [Berger and Rigoutsos, 1991], [Berger, 1986]

### Refinement criteria

Heuristic error estimation [Berger, 1982]:

Local truncation error of scheme of order o

\[ q(x, t + \Delta t) - H^{(\Delta t)}(q(t, t)) = C_\Delta t^{o+1} + O(\Delta t^{o+2}) \]

For q smooth after 2 steps \( \Delta t \)

\[ q(x, t + \Delta t) - H^{(\Delta t)^2}(q(t, t - \Delta t)) = 2C_\Delta t^{o+1} + O(\Delta t^{o+2}) \]

and after 1 step with \( 2\Delta t \)

\[ q(x, t + \Delta t) - H^{(2\Delta t)}(q(t, t - \Delta t)) = 2^{o+1}C_\Delta t^{o+1} + O(\Delta t^{o+2}) \]

Gives

\[ H^{(\Delta t)}(q(t, t - \Delta t)) - H^{(2\Delta t)}(q(t, t - \Delta t)) = (2^{o+1} - 2)C_\Delta t^{o+1} + O(\Delta t^{o+2}) \]

### Heuristic error estimation for FV methods

1. Error estimation on interior cells

2. Create temporary Grid coarsened by factor 2

   Initialize with fine-grid solutions

   \[ H^{\Delta t}(H^{\Delta t} Q(t, t - \Delta t)) = H^{\Delta t}(H^{\Delta t} Q(t, t - \Delta t)) + H^{\Delta t}(\tilde{Q}(t, t - \Delta t)) \]

3. Compare temporary solutions

   \[ H^{2\Delta t}(\tilde{Q}(t, t - \Delta t)) \]
Usage of heuristic error estimation

Current solution integrated tentatively 1 step with $\Delta t_i$ and coarsened

$$Q(t_i + \Delta t) := \text{Restrict} \left( \mathcal{T}_2^{\Delta t_i} Q'(t_i - \Delta t) \right)$$

Previous solution coarsened and integrated 1 step with $2\Delta t_i$

$$Q(t_i + \Delta t) := \mathcal{T}^{2\Delta t_i} \text{Restrict} (Q'(t_i - \Delta t))$$

Local error estimation of scalar quantity $w$

$$\gamma_{jk}^w := \frac{|w(\bar{Q}_{jk}(t + \Delta t)) - w(Q_{jk}(t + \Delta t))|}{2^{\alpha+1} - 2}$$

In practice [Deiterding, 2003] use

$$\max(|w(\bar{Q}_{jk}(t + \Delta t))|, S_w) > \eta_w$$

Parallelization strategies

Decomposition of the hierarchical data

- Distribution of each grid
- Separate distribution of each level, cf. [Rendleman et al., 2000]
- Rigorous domain decomposition
  - Data of all levels resides on same node
  - Grid hierarchy defines unique "floor-plan"
  - Redistribution of data blocks during reorganization of hierarchical data
  - Synchronization when setting ghost cells

Parallel machine with $P$ identical nodes. $P$ non-overlapping portions $G^p_0$, $p = 1, \ldots, P$ as

$$G_0 = \bigcup_{p=1}^{P} G^p_0 \quad \text{with} \quad G^p_0 \cap G^q_0 = \emptyset \quad \text{for} \quad p \neq q$$

Higher level domains $G_i$ follow decomposition of root level

$$G_i^p := G_i \cap G^p_0$$

With $\mathcal{N}(\cdot)$ denoting number of cells, we estimate the workload as

$$\mathcal{W}(\Omega) = \max_{i=0}^{n} \left[ \mathcal{N}(G_i \cap \Omega) \prod_{k=0}^{i} \mathcal{r}_k \right]$$

Equal work distribution necessitates

$$\mathcal{L}^p := \frac{P \cdot \mathcal{W}(G^p_0)}{\mathcal{W}(G_0)} \approx 1 \quad \text{for all} \quad p = 1, \ldots, P$$

[Deiterding, 2005]
The recursive algorithm in parallel

\textbf{AdvanceLevel}(l)

Repeat \( r_l \) times

Set ghost cells of \( Q^l(t) \)

If time to regrid?

\textbf{Regrid}(l)

\textbf{UpdateLevel}(l)

If level \( l > 1 \) exists?

Set ghost cells of \( Q^l(t + \Delta t_l) \)

\textbf{AdvanceLevel}(l + 1)

Average \( Q^{l+1}(t + \Delta t_l) \) onto \( Q^l(t + \Delta t_l) \)

Correct \( Q^l(t + \Delta t_l) \) with \( \delta F^{l+1} \)

\( t := t + \Delta t_l \)

\textbf{UpdateLevel}(l)

For all \( m = 1 \) to \( M_l \) do

\( Q(G_{l,m}^r, t) \rightarrow Q(G_{l,m}, t + \Delta t_l), F^n(G_{l,m}, t) \)

If level \( l > 0 \)

Add \( F^n(\partial G_{l,m}, t) \) to \( \delta F^n(l) \)

If level \( l > 1 \) exists?

Init \( \delta F^{n+1}_l \) with \( F^n(G_{l,m}, t + \Delta t_{l+1}, t) \)

Regrid algorithm in parallel

\textbf{Regrid}(l) - Regrid all levels \( l > l \)

For \( l = l \) Down to \( l \)

Flag \( N^l \) according to \( Q^l(t) \)

If level \( l > 1 \) exists?

Flag \( N^l \) below \( G_{l+1} \)

Generate \( G_{l+1} \) from \( N^l \)

\( G_l := G_l \)

For \( l = l \) To \( l \)

\( C_{l+1} := G_l \cap G_{l+1} \)

\( G_{l+1} := G_{l+1} \setminus C_{l+1} \)

Recompose(l)

Parallel SAMR method

Physical boundary

Interpolation

Local synchronization

Parallel synchronization

Ghost cell values:
Recomposition algorithm in parallel

Recompose(λ) – Reorganize all levels

Generate \( G^0_l \) from \( \{G_0, \ldots, G_l, G_{l+1}, \ldots, G_N \} \)

For \( \ell = l + 1 \) to \( l + 1 \)

If \( \ell > l \)

\[ G^0_{\ell} := G_{\ell} \cap G^0_0 \]

Interpolate \( Q^{\ell-1}(t) \) onto \( Q^\ell(t) \)

else

\[ G^0_{\ell} := G_{\ell} \cap G^0_0 \]

If \( \ell > 0 \)

Copy \( \delta F^{n,\ell} \) onto \( \delta F^{n,0} \)

\[ \delta F^{n,\ell} := \delta F^{n,0} \]

If \( \ell > l \) then \( \kappa = 0 \) else \( \kappa = 1 \)

For \( \kappa = 0 \) to \( \kappa \), Do

Copy \( Q^\ell(t) \) onto \( Q^\ell(t) \)

Set ghost cells of \( Q^\ell(t) \)

\[ Q^\ell(t) := Q^\ell(t) \]

\[ G_{\ell} := G_{\ell} \]

Space-filling curve algorithm

Necessary domain of Space-Filling Curve

Calculation domain

Proc. 1 [ ] High Workload
Proc. 2 [ ] Medium Workload
Proc. 3 [ ] Low Workload

Outline

The serial Berger-Colella SAMR method
Block-based data structures
Numerical update
Conservative flux correction
Level transfer operators
The basic recursive algorithm
Cluster algorithm
Refinement criteria

Parallel SAMR method
Domain decomposition
A parallel SAMR algorithm
Partitioning

Examples
Euler equations

SAMR accuracy verification

Gaussian density shape

\[ \rho(x_1, x_2) = 1 - \left( \frac{\sqrt{x_1^2 + x_2^2}}{R} \right)^2 \]

is advected with constant velocities \( u_1 = u_2 = 1 \), \( \rho_0 = 1 \), \( R = 1/4 \)

- Domain \([-1, 1] \times [-1, 1] \), periodic boundary conditions, \( t_{end} = 2 \)
- Two levels of adaptation with \( r_{n,2} = 2 \), finest level corresponds to \( N \times N \) uniform grid

Use locally conservative interpolation

\[ Q_{v,w}^\ell := Q_{v,j}^\ell + f_1(Q_{v+1,j}^\ell - Q_{v-1,j}^\ell) + f_2(Q_{j+1,v}^\ell - Q_{j-1,v}^\ell) \]

with factor \( f_1 = \frac{x_{v+1,j} - x_{v,j}}{2\Delta x_{v,j}} \), \( f_2 = \frac{x_{j+1,v} - x_{j,v}}{2\Delta x_{j,v}} \) to also test flux correction

This prolongation operator is not monotonicity preserving! Only applicable to smooth problems.
SAMR accuracy verification: results

### VanLeer flux vector splitting with dimensional splitting, Minmod limiter

<table>
<thead>
<tr>
<th>$N$</th>
<th>Unigrid</th>
<th>SAMR - fixup</th>
<th>SAMR - no fixup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error</td>
<td>Order</td>
<td>$\Delta\rho$</td>
<td>Error</td>
</tr>
<tr>
<td>20</td>
<td>0.10946400</td>
<td>1.369</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>0.04239430</td>
<td>1.590</td>
<td>0.052669</td>
</tr>
<tr>
<td>80</td>
<td>0.01408160</td>
<td>1.514</td>
<td>0.00165516</td>
</tr>
<tr>
<td>160</td>
<td>0.00492945</td>
<td>1.754</td>
<td>1.503</td>
</tr>
<tr>
<td>320</td>
<td>0.00146132</td>
<td>1.805</td>
<td>0.00051513</td>
</tr>
<tr>
<td>640</td>
<td>0.00041809</td>
<td>1.807</td>
<td>0.00046587</td>
</tr>
</tbody>
</table>

### Fully two-dimensional Wave Propagation Method, Minmod limiter

<table>
<thead>
<tr>
<th>$N$</th>
<th>Unigrid</th>
<th>SAMR - fixup</th>
<th>SAMR - no fixup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error</td>
<td>Order</td>
<td>$\Delta\rho$</td>
<td>Error</td>
</tr>
<tr>
<td>20</td>
<td>0.10620000</td>
<td>1.380</td>
<td>0.01536580</td>
</tr>
<tr>
<td>40</td>
<td>0.04079600</td>
<td>1.598</td>
<td>0.00505406</td>
</tr>
<tr>
<td>80</td>
<td>0.01482500</td>
<td>1.513</td>
<td>0.00147218</td>
</tr>
<tr>
<td>160</td>
<td>0.0047301</td>
<td>1.513</td>
<td>0.00044500</td>
</tr>
</tbody>
</table>

### Benchmark run 2: point-explosion in 3D
- Benchmark from the Chicago workshop on AMR methods, September 2003
- Sedov explosion - energy deposition in sphere of radius 4 finest cells
- 3D-Wave-Prop. Method with hybrid Roe-HLL scheme
- Base grid $32^3$
- Refinement factor $n = 2$
- Effective resolutions: $128^3$, $256^3$, $512^3$, $1024^3$
- Grid generation efficiency $\eta_{tol} = 85\%$
- Proper nesting enforced
- Buffer of 1 cell

### Benchmark run 2: visualization of refinement
Benchmark run 2: performance results

<table>
<thead>
<tr>
<th>$l_{max}$</th>
<th>Grids</th>
<th>Cells</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>28,768</td>
<td>32</td>
</tr>
<tr>
<td>3</td>
<td>28,768</td>
<td>32</td>
</tr>
<tr>
<td>4</td>
<td>190,944</td>
<td>580,568</td>
</tr>
<tr>
<td>5</td>
<td>2,234,272</td>
<td>8,429,624</td>
</tr>
</tbody>
</table>

Breakdown of CPU time on 8 nodes SGI Altix 3000 (Linux-based shared memory system)

<table>
<thead>
<tr>
<th>Task [%]</th>
<th>$l_{max} = 2$</th>
<th>$l_{max} = 3$</th>
<th>$l_{max} = 4$</th>
<th>$l_{max} = 5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integration</td>
<td>73.7</td>
<td>77.2</td>
<td>72.9</td>
<td>37.8</td>
</tr>
<tr>
<td>Fixup</td>
<td>2.6</td>
<td>46</td>
<td>3.1</td>
<td>58</td>
</tr>
<tr>
<td>Boundary</td>
<td>10.1</td>
<td>79</td>
<td>6.3</td>
<td>78</td>
</tr>
<tr>
<td>Reconversion</td>
<td>7.4</td>
<td>8.0</td>
<td>15.1</td>
<td>50.4</td>
</tr>
<tr>
<td>Clustering</td>
<td>0.5</td>
<td>0.6</td>
<td>0.7</td>
<td>1.0</td>
</tr>
<tr>
<td>Output/Misc</td>
<td>5.7</td>
<td>4.0</td>
<td>3.6</td>
<td>1.7</td>
</tr>
<tr>
<td>Time [min]</td>
<td>0.5</td>
<td>5.1</td>
<td>13.0</td>
<td>2100.0</td>
</tr>
<tr>
<td>Uniform [min]</td>
<td>5.4</td>
<td>160</td>
<td>~5,000</td>
<td>~180,000</td>
</tr>
<tr>
<td>Factor of AMR savings</td>
<td>11</td>
<td>31</td>
<td>69</td>
<td>86</td>
</tr>
<tr>
<td>Time steps</td>
<td>15</td>
<td>27</td>
<td>52</td>
<td>115</td>
</tr>
</tbody>
</table>

References I


References II


Lecture 3

Numerical methods for combustion research

Course Block-structured Adaptive Finite Volume Methods for Shock-Induced Combustion Simulation

Ralf Deiterding

German Aerospace Center (DLR)
Institute for Aerodynamics and Flow Technology
Bunsenstr. 10, Göttingen, Germany

E-mail: ralf.deiterding@dlr.de
Outline

Complex geometry
- Boundary aligned meshes
- Cartesian techniques
- Implicit geometry representation
- Accuracy / verification

Combustion
- Equations
- Upwind schemes for combustion
- Tests with one-step chemistry
- Shock-induced combustion with real chemistry

SAMR on boundary aligned meshes
- Analytic or stored geometric mapping of the coordinates (graphic from [Yamaleev and Carpenter, 2002])
  - Topology remains unchanged and thereby entire SAMR algorithm
  - Patch solver and interpolation need to consider geometry transformation
  - Handles boundary layers well

Cut-cell techniques
- Accurate embedded boundary method
  \[ V_j^{n+1/2}Q_j^{n+1} = V_j^nQ_j^n + \Delta t \left( A_{j+1/2}^{n+1/2} F(Q,j) - A_{j-1/2}^{n+1/2} F(Q,j-1) \right) \]
  - Methods that represent the boundary sharply:
    - Cut-cell approach constructs appropriate finite volumes
    - Conservative by construction. Correct boundary flux
    - Key question: How to avoid small-cell time step restriction in explicit methods?
      - Cell merging: [Quirk, 1994a]
    - Usually explicit geometry representation used [Aftosmis, 1997], but can also be implicit, cf. [Nourgaliev et al., 2003], [Murman et al., 2003]
Embedded boundary techniques

Volume of fluid methods that resemble a cut-cell technique on purely Cartesian mesh

- Redistribution of boundary flux achieves conservation and bypasses time step restriction: [Pember et al., 1999], [Berger and Helzel, 2002]

Methods that diffuse the boundary in one cell (good overview in [Mittal and Iaccarino, 2005]):
- Related to the immersed boundary method by Peskin, cf. [Roma et al., 1999]
- Boundary prescription often by internal ghost cell values, cf. [Tseng and Ferziger, 2003]
- Not conservative by construction but conservative correction possible
- Usually combined with implicit geometry representation

Level-set method for boundary embedding

- Implicit boundary representation via distance function \( \phi \), normal \( n = \nabla \phi / |\nabla \phi| \)
- Complex boundary moving with local velocity \( w \), treat interface as moving rigid wall
- Construction of values in embedded boundary cells by interpolation / extrapolation

\[
\bar{x} = x + 2w \cdot n
\]

Velocity in ghost cells

\[
u' = (2w \cdot n - u \cdot n)n + (u \cdot t)t = 2((w - u) \cdot n) n + u
\]

Closest point transform algorithm

The signed distance \( \phi \) to a surface \( I \) satisfies the eikonal equation [Sethian, 1999]

\[|\nabla \phi| = 1 \quad \text{with} \quad \phi|_I = 0\]

Solution smooth but non-differentiable across characteristics.

Distance computation trivial for non-overlapping elementary shapes but difficult to do efficiently for triangulated surface meshes:

- Geometric solution approach with closest-point-transform algorithm [Mauch, 2003]

\[b\text{-rep}\]

Surface mesh \( I \)

Distance \( \phi \)

Normal to closest point

The characteristic / scan conversion algorithm

1. Build the characteristic polyhedrons for the surface mesh
2. For each face/edge/vertex
   2.1 Scan convert the polyhedron.
   2.2 Compute distance to that primitive for the scan converted points
3. Computational complexity.
   - \( O(m) \) to build the b-rep and the polyhedra.
   - \( O(n) \) to scan convert the polyhedra and compute the distance, etc.
4. Problem reduction by evaluation only within specified max. distance

[Mauch, 2003], see also [Deiterding et al., 2006]
Complex geometry
Combustion
References
Accuracy / verification

Accuracy / verification
Shock reflection: SAMR solution for Euler equations
∆x = 25 mm
Δx = 12.5 mm
Δx = 3.125 mm

Verification: shock reflection

- Reflection of a Mach 2.38 shock in nitrogen at 43° wedge
- 2nd order MUSCL scheme with Roe solver, 2nd order multidimensional wave propagation method

Stationary vortex: results
Compute one full rotation, Roe solver, embedded slip wall boundary conditions x_{c,1} = 0.5, x_{c,2} = 0.5, R = 0.4, f_{wall} = 1, ∆h = ∆x_1 = ∆x_2 = 1/N, α = Rπ

No embedded boundary
Marginal shear flow along embedded boundary, α = Rπ, R_c = R, U_W = 0

Major shear flow along embedded boundary, α = Rπ, R_c = R, U_W = 0

Numerical methods for combustion research

Wave Propagation
Godunov Splitting

<table>
<thead>
<tr>
<th>N</th>
<th>Wave Propagation</th>
<th>Godunov Splitting</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>0.0120056</td>
<td>0.0179236</td>
</tr>
<tr>
<td>40</td>
<td>0.0035074</td>
<td>0.0011898</td>
</tr>
<tr>
<td>80</td>
<td>0.0014193</td>
<td>0.0001588</td>
</tr>
<tr>
<td>160</td>
<td>0.0005032</td>
<td>1.50546e-05</td>
</tr>
</tbody>
</table>

Error
Mass loss

<table>
<thead>
<tr>
<th>N</th>
<th>Wave Propagation</th>
<th>Godunov Splitting</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>0.0423925</td>
<td>0.0423925</td>
</tr>
<tr>
<td>40</td>
<td>0.0358735</td>
<td>0.0358735</td>
</tr>
<tr>
<td>80</td>
<td>0.0212340</td>
<td>0.0212340</td>
</tr>
<tr>
<td>160</td>
<td>0.0212340</td>
<td>0.0212340</td>
</tr>
</tbody>
</table>

Error
Mass loss

Verification: shock reflection

Shock reflection: SAMR solution for Euler equations

No embedded boundary

Marginal shear flow along embedded boundary, α = Rπ, R_c = R, U_W = 0

Major shear flow along embedded boundary, α = Rπ, R_c = R, U_W = 0

Numerical methods for combustion research

Wave Propagation
Godunov Splitting

<table>
<thead>
<tr>
<th>N</th>
<th>Wave Propagation</th>
<th>Godunov Splitting</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>0.0120056</td>
<td>0.0179236</td>
</tr>
<tr>
<td>40</td>
<td>0.0035074</td>
<td>0.0011898</td>
</tr>
<tr>
<td>80</td>
<td>0.0014193</td>
<td>0.0001588</td>
</tr>
<tr>
<td>160</td>
<td>0.0005032</td>
<td>1.50546e-05</td>
</tr>
</tbody>
</table>

Error
Mass loss

<table>
<thead>
<tr>
<th>N</th>
<th>Wave Propagation</th>
<th>Godunov Splitting</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>0.0423925</td>
<td>0.0423925</td>
</tr>
<tr>
<td>40</td>
<td>0.0358735</td>
<td>0.0358735</td>
</tr>
<tr>
<td>80</td>
<td>0.0212340</td>
<td>0.0212340</td>
</tr>
<tr>
<td>160</td>
<td>0.0212340</td>
<td>0.0212340</td>
</tr>
</tbody>
</table>

Error
Mass loss

Stationary vortex: results
Compute one full rotation, Roe solver, embedded slip wall boundary conditions x_{c,1} = 0.5, x_{c,2} = 0.5, R = 0.4, f_{wall} = 1, ∆h = ∆x_1 = ∆x_2 = 1/N, α = Rπ

Accuracy test: stationary vortex
Construct non-trivial radially symmetric and stationary solution by balancing hydrodynamic pressure and centripetal force per volume element, i.e.

\[ \frac{d}{dr}p(r) = \frac{\rho(r)}{r} \frac{U(r)^2}{r} \]

For \( \rho_0 \equiv 1 \) and the velocity field

\[ U(r) = \alpha \left\{ \begin{array}{ll}
2r/R & \text{if } 0 < r < R/2, \\
2(1 - r/R) & \text{if } R/2 \leq r \leq R, \\
0 & \text{if } r > R,
\end{array} \right. \]

one gets with boundary condition \( p(R) = \rho_0 \equiv 2 \) the pressure distribution

\[ p(r) = \rho_0 + 2\rho_0 \alpha^2 \left\{ \begin{array}{ll}
r^2/R^2 + 2 \log 2 & \text{if } 0 < r < R/2, \\
r^2/R^2 + 3 - 4r/R + 2 \log(r/R) & \text{if } R/2 \leq r \leq R, \\
0 & \text{if } r > R.
\end{array} \right. \]

Entire solution for Euler equations reads

\[ \rho(x_1, x_2, t) = \rho_0, \quad u_1(x_1, x_2, t) = -U(r) \sin \phi, \quad u_2(x_1, x_2, t) = U(r) \cos \phi, \quad p(x_1, x_2, t) = p(r) \]

for all \( t \geq 0 \) with \( r = \sqrt{(x_1 - x_{c,1})^2 + (x_2 - x_{c,2})^2} \) and \( \phi = \arctan \frac{x_2 - x_{c,2}}{x_1 - x_{c,1}} \)

Verification: shock reflection

- Reflection of a Mach 2.38 shock in nitrogen at 43° wedge
- 2nd order MUSCL scheme with Roe solver, 2nd order multidimensional wave propagation method

Cartesian base grid 360 × 160 cells on domain of 36 mm × 16 mm with up to 3 refinement levels with \( \eta = 2, 4, 4 \) and \( \Delta x_{1,2} = 3.125 \mu m, 38 \) h CPU

GFM base grid 390 × 330 cells on domain of 26 mm × 22 mm with up to 3 refinement levels with \( \eta = 2, 4, 4 \) and \( \Delta x_{1,2} = 2.849 \mu m, 200 \) h CPU

vt//amroc/clawpack/applications/euler/2d/Ramp
vt//amroc/clawpack/applications/euler/2d/RampGFM
Shock reflection: solution for Navier-Stokes equations

- No-slip boundary conditions enforced
- Conservative 2nd order centered differences to approximate stress tensor and heat flow

![Graphs showing different shock reflections](images)

Governing equations for premixed combustion

Euler equations with reaction terms
\[
\frac{\partial \rho_i}{\partial t} + \frac{\partial}{\partial x_a}(\rho_i u_n) = \dot{\omega}_i, \quad i = 1, \ldots, K
\]
\[
\frac{\partial}{\partial t}(\rho u_k) + \frac{\partial}{\partial x_a}(\rho u_k u_n + \delta_{n a} p) = 0, \quad k = 1, \ldots, d
\]

\[
\frac{\partial}{\partial t}(\rho E) + \frac{\partial}{\partial x_a}(u_n(\rho E + p)) = 0
\]

Ideal gas law and Dalton's law for gas-mixtures
\[
p(\rho_1, \ldots, \rho_K, T) = \sum_{i=1}^{K} \rho_i = \sum_{i=1}^{K} \rho_i \frac{R}{W_i} T = \rho \frac{R}{W} T \quad \text{with} \quad \sum_{i=1}^{K} \rho_i = \rho, \quad Y_i = \frac{\rho_i}{\rho}
\]

Caloric equation
\[
h(Y_1, \ldots, Y_K, T) = \sum_{i=1}^{K} Y_i h_i(T) \quad \text{with} \quad h_i(T) = h_i^0 + \int_0^T c_{\mu i}(s) ds
\]

Computation of \( T = T(\rho_1, \ldots, \rho_K, e) \) from implicit equation
\[
\sum_{i=1}^{K} \rho_i h_i(T) - R T \sum_{i=1}^{K} \frac{\rho_i}{W_i} - \rho e = 0
\]

for thermally perfect gases with \( \gamma_i(T) = c_{\rho i}(T)/c_{u i}(T) \)

Chemistry

Arrhenius-Kinetics:
\[
\dot{\omega}_i = \sum_{j=1}^{M} (\nu_r^{ij} - \nu_f^{ij}) \left( k_i \prod_{n=1}^{K} \left( \frac{\rho_n}{W_n} \right)^{\nu_m^{ij}} - k_f^{ij} \prod_{n=1}^{K} \left( \frac{\rho_n}{W_n} \right)^{\nu_m^{ij}} \right) \quad i = 1, \ldots, K
\]

- Parsing of mechanisms with Chemkin-II
- Evaluation of \( \dot{\omega}_i \) with automatically generated optimized Fortran-77 functions in the line of Chemkin-II

Integration of reaction rates: ODE integration in \( S(\cdot) \) for Euler equations with chemical reaction

- Standard implicit or semi-implicit ODE-solver subcycles within each cell
- \( \rho, \ e, \ u_k \) remain unchanged!

\[
\frac{\partial}{\partial t} \rho_i = W_i \dot{\omega}_i(\rho_1, \ldots, \rho_K, T) \quad i = 1, \ldots, K
\]

Use Newton or bisection method to compute \( T \) iteratively.
Hyperbolicity of the homogeneous equations

Consider Jacobian $A_1(q) = \partial f_1(q)/\partial q$

$$
A_1(q) = \begin{bmatrix}
-q_1(1 - Y_1) & -q_1 Y_1 - q_2 Y_1 & \cdots & -q_1 Y_1 & Y_1 & 0 & 0 \\
-q_2(1 - Y_2) & -q_2 Y_2 & \cdots & -q_2 Y_2 & 0 & 0 & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\
-q_K(1 - Y_K) & -q_K Y_K & \cdots & -q_K Y_K & 0 & 0 & 0 \\
\end{bmatrix}
$$

with

$$
\frac{\partial p}{\partial \rho_j} = \gamma \left( \frac{u_j^2}{2} - h_j \right) + \gamma R_j = \phi_j \quad \text{and} \quad \frac{\partial p}{\partial E} = \gamma - 1 = \tilde{\gamma}
$$

Eigenvalues of $A_1(q)$ are

$\{u_1 - c, u_1, \ldots, u_1, u_1 + c\}$. The system is hyperbolic, if the frozen speed of sound $c$ given by

$$
c^2 = \left( \frac{\partial p}{\partial \rho} \right)_{i,Y_i} = \sum_{i=1}^{K} Y_i \phi_i - \tilde{\gamma} u^2 + \tilde{\gamma} H = \gamma \frac{p}{\rho} > 0.
$$

is real.

Empirical argument: $\partial_t Y_i + u_1 \partial_x Y_i = 0$

Van Leer flux vector splitting

$$
f^\pm(q) = \pm \frac{\rho}{4c} (u_1 \pm c)^2
$$

Stability condition

$$
C^\text{VL}_{\text{CFL}} := \max_{j \in \Omega} \left\{ \left[ (u_1, j) + c \right] \right\} \frac{\Delta t}{\Delta x} \leq 1
$$

with $\Omega_j = \begin{cases} 
\gamma_j + 3 & \text{if } |u_1, j| < c_j, \\
2 & \text{otherwise}.
\end{cases}

Roe solver

Roe averages $\bar{\mu}$, $\bar{u}$, $\bar{\psi}$, $\bar{h}$, $\bar{\tilde{\tau}}$, $\bar{\tilde{\xi}}$, $\bar{Y}_i$

Define $\bar{\psi}_i = \frac{1}{\bar{T}_i - \bar{T}_j} \int_{\bar{T}_j}^{\bar{T}_i} \psi_i(\tau) d\tau$, $\bar{\tau}_i = \frac{1}{\bar{T}_i - \bar{T}_j} \int_{\bar{T}_j}^{\bar{T}_i} \tau \psi_i(\tau) d\tau$

$$\Delta t := f(q_{k+1}) - f(q_k) = \sum_{m=1}^{M} \tilde{a}_m \lambda_m(q_k) (r_m(q_{k+1}) - r_m(q_k))$$

with $\Delta q := q_{k+1} - q_k = \sum_{m=1}^{M} \tilde{a}_m r_m(q)$.

With matrix of right eigenvectors

$$
R(q) = \begin{bmatrix}
Y_1 & 1 & 0 & \cdots & 0 & 0 & 0 & Y_1 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & \cdots & 1 & 0 & 0 & Y_1 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\
u_1 - c & u_1 & \cdots & u_1 & 0 & 0 & u_1 + c & u_1 + c \\
u_2 & u_2 & \cdots & u_2 & 1 & 0 & 0 & u_2 \\
u_3 & u_3 & \cdots & u_3 & 0 & 1 & 0 & u_3 \\
u_4 & u_4 & \cdots & u_4 & 0 & 0 & 1 & u_4 \\
\end{bmatrix}
$$

and evaluating $R^{-1}(\Delta \tilde{q})$ one gets the wave strengths eventually as

$$
a_1, a_{k+1} = \frac{\Delta \tilde{q} \cdot \tilde{\Delta} u_k}{2 \tilde{\Delta} u_k}, \quad a_{k+1} = \Delta \tilde{q}_i - \bar{\psi}_i \frac{\tilde{\Delta} u_k}{2 \tilde{\Delta} u_k}.$$

References

Complex geometry

Equations

Hyperbolicity of the homogeneous equations

Steger-Warming flux vector splitting
Roe solver - fixes

Mass fraction positivity: Calculate numerical fluxes of partial densities by [Larrouituro, 1993]

\[ F_i^*(\rho L, \rho R) = F_i(\rho L, \rho R), \quad F_i(\rho L, \rho R) > 0, \]
\[ Y_{i,L}, \quad F_i(\rho L, \rho R) < 0. \]

to ensure positivity of \( Y_i \).

Example: Mass fraction for a typical Riemann problem after 1 time step with the Roe solver

![Graph showing mass fraction vs. x with f=1.0 and f=1.8]

Further: Switch from Roe to HLL scheme near vacuum state to avoid unphysical values.

Riemann solver for combustion

(S1) Calculate standard Roe-averages \( \hat{\rho}, \hat{u}, \hat{R}, \hat{Y}_i, \hat{\gamma} \).

(S2) Compute \( \hat{\gamma} := \frac{\hat{\gamma}_i}{2} \), with \( \hat{\gamma}_{i+1/2} = \frac{1}{R_L - R_R} \int_{R_L}^{R_R} \hat{\gamma}(\tau) d\tau \).

(S3) Calculate \( \hat{\gamma}_i := (\hat{\gamma} - 1) \left( \frac{\hat{E}}{\hat{\rho}} - \hat{h}_i \right) + \hat{\gamma} R \), \( \hat{\gamma} \) with standard Roe-averages \( \hat{\rho}, \hat{u}, \hat{R} \).

(S4) Calculate \( \hat{\rho} := \sum_{n=1}^{K} \hat{\rho}_n, \hat{u} \), \( \hat{\rho} = \sum_{n=1}^{K} \hat{\rho}_n, \hat{u} \) and \( \Delta \rho \) to compute the wave strengths \( \lambda_m \).

(S5) Calculate \( W_1 = a \hat{\rho}, W_2 = \sum_{n=1}^{K} a \hat{\rho}_n, W_3 = \sum_{n=1}^{K} a \hat{\rho}_n \).

(S6) Evaluate \( s_1 = \hat{u} - \hat{\gamma}, s_2 = \hat{u} + \hat{\gamma}, s_3 = \hat{u} + \hat{\gamma} \).

(S7) Evaluate \( \rho u_{L/R}, \rho u_{L/R}^*, \rho u_{L/R} \) from \( \hat{\rho}_L = \hat{\rho}_L, \hat{\rho}_R = \hat{\rho}_R \).

(S8) If \( \rho u_{L/R} \leq 0 \) or \( \rho u_{L/R}^* \leq 0 \) use \( \rho u_{L/R} = \rho u_{L/R} \) and go to (S12).

(S9) Entropy correction: Evaluate \( \hat{\rho}, \hat{u}, \hat{E} \).

(S10) Positivity correction: Replace \( F_i \) by \( F_i^* = F_i \).

(S11) Evaluate maximal signal speed by \( S = \max(|s_1|, |s_2|). \)

ZND Detonation Model with Simplified Chemistry

Assume a stationary 1D detonation with irreversible reaction

\[ A \rightarrow B \]

with energy release \( q_0 = -\Delta h^0 \) and \( k^f(T) = K \exp(-E_A/R T) \).

Simplified Kinetics \( W_A \omega_A = -K_{PA} \exp(-E_A/R T), \quad W_B \omega_B = -W_A \omega_A \)

With \( \gamma_A = \gamma_B \) the equation of state is \( p(\rho_A, \rho_B, \gamma) = (\gamma - 1) (\rho_A - \rho_B) \).

Integration of the stationary 1D Euler equations

\[ \frac{d\rho}{dx} \rho u^2 + p = 0, \quad \frac{d\rho}{dx} = \frac{\partial}{\partial x} \left( \rho u \right) = 0 \]

\[ \frac{d\rho}{dx} = \frac{K\rho(1-\lambda)}{\exp(-E_A\rho/p)} = \frac{K\rho(1-\lambda)}{\exp(-E_A\rho/p)} \frac{d\rho}{dx} \]

gives for \( \lambda \), the mass fraction of the product \( B \),

\[ \frac{d\lambda}{dx} = K(1-\lambda) \exp\left(-\frac{E_A(\rho(1-\lambda))}{\rho}\right) = f(\lambda), \]

i.e.

\[ \int_0^\gamma f(\lambda) \) d\lambda = \lambda(x') \]

\( D_C \) is the minimal detonation velocity. The overdrive-parameter \( f = (D/D_C)^2 \geq 1 \) determines stability.

Normalization:

\[ L_{1/2} = \int_0^\frac{1}{2} \frac{d\lambda}{f(\lambda)} \]

References

[Harten, 1983]

[Quirk, 1994b]

[Deiterding, 2003]
Unstable ZND Detonation

\[ \gamma = 1.2, \quad E^+ = 50, \quad q_0 = 50, \quad f = 1.6, \quad CFL = 0.9 \]

Comparison of FV Schemes: MUSCL, Van Albada-limiter

\[ \gamma = 1.2, \quad E = 50, \quad q_0 = 50, \quad f = 3.0, \quad 40 \text{ Pts}/L_{1/2} \]

Comparison of FV Schemes - II

\[ \gamma = 1.2, \quad E = 10, \quad q_0 = 50, \quad f = 1.2, \quad 40 \text{ Pts}/L_{1/2} \]

Detonations - motivation for SAMR

- Extremely high spatial resolution in reaction zone necessary.
- Minimal spatial resolution: \( 7 - 8 \text{ Pts}/l_g \rightarrow \Delta x_1 \approx 0.2 - 0.175 \text{ mm} \)
- Uniform grids for typical geometries: \( > 10^7 \text{ Pts} \) in 2D, \( > 10^9 \text{ Pts} \) in 3D \( \rightarrow \) Self-adaptive finite volume method (AMR)

Approximation of \( \text{H}_2 : \text{O}_2 \) detonation at \( \sim 1.5 \text{ Pts}/l_g \) (left) and \( \sim 24 \text{ Pts}/l_g \) (right)
**Detonation ignition in a shock tube**

- Shock-induced detonation ignition of \(H_2 : O_2 : Ar\) mixture at molar ratios 2:1:7 in closed 1d shock tube
- Insufficient resolution leads to inaccurate results
- Reflected shock is captured correctly by FV scheme, detonation is resolution dependent
- Fine mesh necessary in the induction zone at the head of the detonation

**Numerical methods for combustion research**

<table>
<thead>
<tr>
<th>Method</th>
<th>System</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniform</td>
<td>Clawpack</td>
<td>Euler Chem 2d/Sphere</td>
</tr>
<tr>
<td>Adaptive</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Detonation ignition in 1d - adaptive vs. uniform**

<table>
<thead>
<tr>
<th>(\Delta x [\mu m])</th>
<th>Uniform</th>
<th>Adaptive</th>
<th>(\Delta T [\mu s])</th>
<th>(t_{max})</th>
<th>(T_{max})</th>
<th>(t_{max})</th>
</tr>
</thead>
<tbody>
<tr>
<td>400</td>
<td>300</td>
<td>166.1</td>
<td>91</td>
<td>63</td>
<td>172.6</td>
<td>99</td>
</tr>
<tr>
<td>200</td>
<td>600</td>
<td>172.6</td>
<td>90</td>
<td>2</td>
<td>2</td>
<td>172.6</td>
</tr>
<tr>
<td>100</td>
<td>1200</td>
<td>175.5</td>
<td>277</td>
<td>3</td>
<td>2.2</td>
<td>175.8</td>
</tr>
<tr>
<td>50</td>
<td>2400</td>
<td>176.9</td>
<td>858</td>
<td>4</td>
<td>2.22</td>
<td>177.3</td>
</tr>
<tr>
<td>25</td>
<td>4800</td>
<td>177.8</td>
<td>2713</td>
<td>4</td>
<td>2.24</td>
<td>177.9</td>
</tr>
<tr>
<td>12.5</td>
<td>9600</td>
<td>178.3</td>
<td>9472</td>
<td>5</td>
<td>2.2,2,4</td>
<td>178.3</td>
</tr>
<tr>
<td>6.25</td>
<td>19200</td>
<td>178.6</td>
<td>35712</td>
<td>5</td>
<td>2.2,4,4</td>
<td>178.6</td>
</tr>
</tbody>
</table>

**Refinement criteria:**

- \(Y_{i} \cdot 10^{-3}\)
- \(n_{ij} \cdot 10^{-3}\)

<table>
<thead>
<tr>
<th>(Y_{i})</th>
<th>(S_{ij} \cdot 10^{-4})</th>
<th>(n_{ij} \cdot 10^{-3})</th>
</tr>
</thead>
<tbody>
<tr>
<td>(O_2)</td>
<td>10.0</td>
<td>2.0</td>
</tr>
<tr>
<td>(H_2O)</td>
<td>7.8</td>
<td>8.0</td>
</tr>
<tr>
<td>(H)</td>
<td>0.16</td>
<td>5.0</td>
</tr>
<tr>
<td>(O)</td>
<td>1.0</td>
<td>5.0</td>
</tr>
<tr>
<td>(OH)</td>
<td>1.8</td>
<td>5.0</td>
</tr>
<tr>
<td>(H_2)</td>
<td>1.3</td>
<td>2.0</td>
</tr>
</tbody>
</table>

\(\epsilon_{\rho} = 0.07 \, \text{kg} \, \text{m}^{-3}, \, \epsilon_{p} = 50 \, \text{kPa}\)

**Shock-induced combustion around a sphere**

- Spherical projectile of radius 1.5 mm travels with constant velocity \(v_i = 2170.6 \, \text{m/s}\) through \(H_2 : O_2 : Ar\) mixture (molar ratios 2:1:7) at 6.67 kPa and \(T = 298 \, \text{K}\)
- Cylindrical symmetric simulation on AMR base mesh of 70 x 40 cells
- Comparison of 3-level computation with refinement factors 2.2 (~ 5 Pts/\(l_{kg}\)) and a 4-level computation with refinement factors 2,2,4 (~ 19 Pts/\(l_{kg}\)) at \(t = 350 \, \mu s\)
- Higher resolved computation captures combustion zone visibly better and at slightly different position (see below)

**Non-equilibrium mechanism for hydrogen-oxygen combustion**

\[
\begin{array}{c|c|c|c}
\hline
\text{Species} & \xi & \beta & \text{Eact} [\text{cal mol}^{-1}] \\
\hline
H + O_2 & O + OH & 1.86 \times 10^{14} & 0.00 & 16790.0 \\
O + O_2 & H + O_2 & 4.83 \times 10^{13} & 0.00 & 18350.0 \\
H_2 + O & H + OH & 8.32 \times 10^{13} & 0.00 & 6950.0 \\
H_2O + O & OH + OH & 3.39 \times 10^{13} & 0.00 & 1100.0 \\
OH + OH & H_2O + O & 3.16 \times 10^{13} & 0.00 & 20350.0 \\
H_2O + H & H_2O + H & 9.55 \times 10^{13} & 0.00 & 20350.0 \\
H_2 + OH & H_2O + H & 2.19 \times 10^{13} & 0.00 & 5150.0 \\
H_2O_2 + OH & H_2 + O & 1.00 \times 10^{13} & 1.00 & 1850.0 \\
H_2O + HO_2 & H_2O_2 + OH & 2.82 \times 10^{13} & 0.00 & 32790.0 \\
\hline
\end{array}
\]

Third body efficiencies: \(f(O_2) = 0.40, \, f(H_2O) = 6.50\) [Westbrook, 1982]
### References I


### References II


### References III


References IV


References V


Outline

**Lecture 4**

**Detonation simulation**

Course *Block-structured Adaptive Finite Volume Methods for Shock-Induced Combustion Simulation*

- Detonation simulation
  - Detonation structures

- Combustion with viscous terms
  - Combustion induced by projectiles
  - Finite volume scheme

- Higher order schemes
  - Hybrid methods
  - Large-eddy simulation
### Planar ZND Structure

Steady situation under Galilean transformation:

\[
\frac{\partial}{\partial x'} (\rho u') = 0 \\
\frac{\partial}{\partial x'} (\rho u'^2 + p) = 0 \\
\frac{\partial}{\partial x'} (u' \rho H) = 0 \\
\frac{\partial Y_i}{\partial x'} = W_i \left( \frac{\rho u' \gamma_i}{\gamma}, \frac{\rho \gamma_i}{\gamma} \right)
\]

CJ-detonation of H\(_2\) : O\(_2\) : Ar with molar ratios 2 : 1 : 7 at \(T_0 = 298\) K and \(p_0 = 6.67\) kPa, \(d_{CJ} \approx 1627\) m/s.

\(t_{ig} \approx 3.55\) μs, \(u'_{ig} \approx 395.5\) m/s, \(l_{ig} \approx 0.14\) cm.

* Cf. vtf/amroc/clawpack/applications/eulerChem/1d/ModelDetonation

### Detonation cell structure in 2D - Regular instability

### Transverse detonation structure - irregular instability

### Simulation of regular structures

- CJ detonation for H\(_2\) : O\(_2\) : Ar (2:1:7) at \(T_0 = 298\) K and \(p_0 = 10\) kPa, cell width 1.6 cm
- Perturb 1d ZND solution with unreacted high-pressure pocket behind front
- Triple point trajectories by tracking \(\max |\omega|\) on auxiliary mesh shifted through grid with CJ velocity, \(\omega = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}\)
- SAMR simulation with 4 additional levels (2,2,2,4), 67.6 Pts/\(l_{ig}\)
- Configuration similar to Oran et al., J. Combustion and Flame 113, 1998.
Detonation structures

Detonation simulation
Combustion with viscous terms
Higher order schemes
References

Detonation simulation

Shock polar analysis of triple points in detonations

- Neglect reaction, but consider \( c_p(T) \)
- Data extracted point-wise from simulation
- Primary triple point \( T \) travels exactly at tip of Mach stem → use oblique shock relations between \( A \) and \( B \)

\[
\begin{align*}
\rho_A u_A \sin(\phi_B) &= \rho_B u_B \sin(\phi_B - \theta_B), \\
\rho_A + \rho_A u_A^2 \sin^2(\phi_B) &= \rho_B + \rho_B u_B^2 \sin^2(\phi_B - \theta_B)
\end{align*}
\]

\[
to \text{evaluate inflow velocity as } u_A = \frac{1}{\sin \phi_B} \sqrt{\frac{\rho_B(\rho_B - \rho_A)}{\rho_A(\rho_B - \rho_A)}}
\]

- Measure inflow angle \( \phi_B \) between Mach stem and triple point trajectory
- Velocity \( a \) of \( T' \) relative to \( T \) cannot be derived that easily: Oblique shock relations across \( C \) and \( D \) hold true both in frame of reference for \( T \) and \( T' \)

\[
\begin{align*}
\rho_C (u_{C,n} - a_n) &= \rho_D (u_{D,n} - a_n), \\
p_C + p_C (u_{C,n} - a_n)^2 &= p_D + p_D (u_{D,n} - a_n)^2, \\
u_{C,t} - a_t &= u_{D,t} - a_t, \\
h_C + \frac{1}{2} (u_{C,n} - a_n)^2 &= h_D + \frac{1}{2} (u_{D,n} - a_n)^2
\end{align*}
\]

\( \rightarrow a_n = 0, a_t \) arbitrary

Estimate \( a_t = \frac{L_t}{t_{hit}} \)

Final distribution to 48 nodes and density distribution on four refinement levels

Detonation simulation

Detonation diffraction - adaptation

Detonation diffraction

CJ detonation for
\[ \text{H}_2 : \text{O}_2 : \text{Ar} = 2 : 1 : 7 \text{ at} \]
\[ T_0 = 298 \text{K and } p_0 = 10 \text{ kPa.} \]
Cell width \( \lambda_c = 1.6 \text{ cm} \)

Adaption criteria (similar as before):
1. Scaled gradients of \( \rho \) and \( p \)
2. Error estimation in \( Y_i \), by Richardson extrapolation

- 25 Points/log, 5 refinement levels (2,2,2,4).
- Adaptive computations use up to \( \sim 2.2 \text{M} \) instead of \( \sim 150 \text{M} \) cells (uniform grid)
- \( \sim 3850 \text{h CPU (} \sim 80 \text{h real time) on 48 nodes Athlon 1.4GHz} \)

Triple point analysis

Double Mach reflection structure shortly before the next collision

\[
\begin{array}{|c|c|c|c|c|}
\hline
\rho / p_0 & p / p_0 & \rho / \rho_0 & T [\text{K}] & u[t/m/s] & M \\
\hline
A & 1.00 & 1.00 & 296 & 1775 & 5.078 \\
B & 31.45 & 4.17 & 2248 & 447 & 0.477 \\
C & 31.69 & 5.32 & 1775 & 965 & 1.153 \\
D & 19.17 & 3.84 & 1487 & 1178 & 1.533 \\
E & 35.61 & 5.72 & 1856 & 901 & 1.053 \\
F & 40.61 & 6.09 & 1987 & 777 & 0.880 \\
\hline
\end{array}
\]
Detonation propagation through pipe bends

- 2D Simulation of CJ detonation for $H_2 : O_2 : Ar = 1 : 1 : 7$ at $T_0 = 298$ K and $p_0 = 10$ kPa.
  - Tube width of 5 detonation cells
- AMR base grid $1200 \times 992$. 4 additional refinement levels (2,2,2,4). 67.6 Pts/lc.
- Adaptive computations use up to $7.1 \cdot 10^6$ cells (4.8 - $10^6$ on highest level) instead of $1.22 \cdot 10^9$ cells (uniform grid)
- $\sim 70,000$ h CPU on 128 CPUs

Detonation simulation
Combustion with viscous terms
Higher order schemes
References

Detonation structures

Triple point tracks

Slight overdrive decreases cell size

Marginal detonation

Mach reflection, high overdrive, structure disappears

Re-ignition with transverse detonation

Detonation failure

$\varphi = 15^\circ$ (left, top), $\varphi = 30^\circ$ (left, bottom), and $\varphi = 60^\circ$ (right)

Detonation simulation
Combustion with viscous terms
Higher order schemes
References

Detonation structures

Triple point structures – $\varphi = 15^\circ$

- Triple point re-initiation after bend with change from transitional to Double Mach reflection

Triple point structures – $\varphi = 30^\circ$

- Triple point quenching and failure as single Mach reflection
Transition criteria

Solve system of oblique shock relations numerically and determine transition boundaries [Ben-Dor, 2007].

- Regular reflection (RR): $M_D^T < 1$
- Single Mach reflection (SMR): $M_C^T < 1$ and $M_D^T > 1$
- Transitional Mach reflection: $M_C^T < 1$ and $M_D^T > 1$
- Double Mach reflection: $M_C^T > 1$ and $M_D^T > 1$
- Here: Nonreactive $H_2 : O_2 : Ar$ mixture at initially 298 K and 10 kPa

For detonations:

$$S := \frac{pC - pD}{pD}$$

[Deiterding, 2011]

Triple point structures

Re-ignition with strong DMR and transverse detonation, $\varphi = 30$, $S = 0.338$

Detonation cell structure in 3D

- Simulation of only one quadrant
- $44.8 \text{Pts} / l$ for $H_2 : O_2 : Ar$. CJ detonation
- SAMR base grid 400x24x24, 2 additional refinement levels (2, 4)
- Simulation uses $\sim 18$ M CPU on 128 CPU Compaq Alpha. $\mathcal{H}$: 37.6 %, $S$: 25.1 %
Detonation cell structure in 3D - II

Schlieren plots of density, mirrored for visualization

Schematic front view of the periodic triple point line structure right plot at the same time.

Axisymmetric Navier-Stokes equations with chemical reaction

\[
\frac{\partial q}{\partial t} + \frac{\partial (f - f_y)}{\partial x} + \frac{\partial (g - g_y)}{\partial y} = \alpha \left( c - g + g_y \right) + s
\]

\[
q = \begin{bmatrix} \rho \\ \rho \mu \\ \rho u \\ \rho v \\ \rho E \end{bmatrix}, \quad f = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho u v \\ \rho v^2 + p \\ \rho (uE + p) \end{bmatrix}, \quad g = \begin{bmatrix} 0 \\ 0 \\ \frac{\rho v}{\rho E + p} \end{bmatrix}, \quad c = \begin{bmatrix} \rho - \tau_{xy} \\ 0 \\ 0 \end{bmatrix}, \quad s = \begin{bmatrix} \omega_i \\ 0 \\ 0 \end{bmatrix}
\]

Chemistry and transport properties

Arrhenius kinetics:

\[
\omega_i = \sum_{j=1}^{M} \left( \nu_{ij}^f - \nu_{ij}^l \right) \left[ k_j \prod_{n=1}^{K} \left( \frac{\rho_n}{W_n} \right)^{\nu_j n} - k_j \prod_{n=1}^{K} \left( \frac{\rho_n}{W_n} \right)^{\nu_j n} \right]
\]

\( i = 1, \ldots, K \)

- Parsing of mechanisms and evaluation of \( \omega_i \) with Chemkin-II
- \( c_T(T) \) and \( h(T) \) tabulated, linear interpolation between values

Mixture viscosity \( \mu = \mu(T, Y) \) with Wilke formula

\[
\mu = \sum_{i=1}^{K} \frac{Y_i \mu_i}{W_i + \sum_{n=1}^{K} Y_n \Phi_{in}/W_n}
\]

with \( \Phi_{im} = \frac{1}{\sqrt{8}} \left( 1 + \frac{W_i}{W_m} \right)^{-\frac{3}{2}} \left( 1 + \left( \frac{\mu_i}{\mu_m} \right)^{\frac{1}{2}} \left( \frac{W_m}{W_i} \right)^{\frac{3}{2}} \right)^2 \)

Mixture thermal conductivity \( k = k(T, Y) \) following Mathur

\[
k = \frac{1}{2} \left( W \sum_{i=1}^{K} \frac{Y_i k_i}{W_i} + \frac{1}{W} \sum_{j=1}^{K} Y_j / (W k_j) \right)
\]

Mixture diffusion coefficients \( D_i = D_i(T, p, Y) \) from binary diffusion \( D_{mi}(T, p) \) as

\[
D_i = \frac{W \sum_{m=1}^{K} Y_m / (W_m D_m)}{1 - Y_i}
\]

- Evaluation with Chemkin-II Transport library
Splitting method

\[
\partial_t q + \partial_x (f - f_v) + \partial_y (g - g_v) = \frac{\alpha}{Y} (c - g + g_v) + s
\]

Dimensional splitting for PDE

\[
\lambda(\Delta t): \quad \partial_t q + \partial_x (f(q) - f_v(q)) = 0, \quad \text{IC: } Q(t_m) \quad \overset{\Delta t}{\longrightarrow} \quad \dot{Q}^{1/2}
\]

\[
\gamma(\Delta t): \quad \partial_t q + \partial_y (g(q) - g_v(q)) = 0, \quad \text{IC: } \dot{Q}^{1/2} \quad \overset{\Delta t}{\longrightarrow} \quad \dot{Q}
\]

Treat right-hand side as source term

\[
\chi(\Delta t): \quad \partial_t q = \frac{\alpha}{Y} (c(q) - g(q) + g_v(q)), \quad \text{IC: } \dot{Q} \quad \overset{\Delta t}{\longrightarrow} \quad Q(t_m + \Delta t)
\]

Chemical source term

\[
S_t(\Delta t): \quad \partial_t q = s(q), \quad \text{IC: } \dot{Q} \quad \overset{\Delta t}{\longrightarrow} \quad Q(t_m + \Delta t)
\]

Formally 1st-order algorithm

\[
Q(t_m + \Delta t) = S_t(\Delta t) \chi(\Delta t) \lambda(\Delta t) Q(t_m)
\]

but all sub-operators 2nd-order accurate or higher.

Finite volume discretization – cont.

Symmetry source term \( \chi(\Delta t) \): Use

\[
Q_{jk}^{n+1} = Q_{jk}^{n} + \Delta t \left( \frac{\alpha}{Y} (c(Q_{jk}^{n}) - g(Q_{jk}^{n})) + \frac{1}{2} G_v (Q_{jk}^{n}, Q_{jk,k+1}^{n}) + G_v (Q_{jk,k-1}^{n}, Q_{jk}^{n})) \right)
\]

within explicit 2nd-order accurate Runge-Kutta method

- Gives 2nd-order central difference approximation of \( G_v \)
- Transport properties \( \mu, k, D_i \) are stored in vector of state \( Q \) and kept constant throughout entire time step

Chemical source term \( S_t(\cdot) \):

- 4th-order accurate semi-implicit ODE-solver subcycles within each cell

\[
\partial_t \rho_i = W_i \dot{\omega}_i (\rho_1, \ldots, \rho_K, T) \quad i = 1, \ldots, K
\]

Finite volume discretization

Time discretization \( t_n = n \Delta t \), discrete volumes \( l_{jk} = [x_{jk} - \frac{1}{2} \Delta x, x_{jk} + \frac{1}{2} \Delta x] \times [y_{jk} - \frac{1}{2} \Delta y, y_{jk} + \frac{1}{2} \Delta y] \times z =: [x_{jk-1/2}, x_{jk+1/2}] \times [y_{jk-1/2}, y_{jk+1/2}] \)

Approximation \( Q_{jk}(t) \approx \frac{1}{l_{jk}} \int_{l_{jk}} f(q(x, t)) \, dx \) and numerical fluxes

\[
F(Q_{jk}(t), Q_{jk+1,k}(t)) = f(q(x_{jk+1/2}, y_{jk}, t)),
\]

\[
F_v(Q_{jk}(t), Q_{jk-1,k}(t)) = f_v(q(x_{jk-1/2}, y_{jk}, t), \nabla q(x_{jk-1/2}, y_{jk}, t))
\]

yield (for simplicity)

\[
Q_{jk}^{n+1} = Q_{jk}^{n} - \frac{\Delta t}{\Delta x} \left[ F(Q_{jk}^{n}, Q_{jk+1,k}^{n}) - F(Q_{jk-1,k}^{n}, Q_{jk}^{n}) \right] + \frac{\Delta t}{\Delta x} \left[ F_v(Q_{jk}^{n}, Q_{jk+1,k}^{n}) - F_v(Q_{jk-1,k}^{n}, Q_{jk}^{n}) \right]
\]

- Riemann solver to approximate \( F(Q_{jk}^{n}, Q_{jk+1,k}^{n}) \)
- 1st-order finite differences for \( F_v(Q_{jk}^{n}, Q_{jk+1,k}^{n}) \) yield 2nd-order accurate central differences in \( \alpha \)

Stability condition used:

\[
\max_{i,j,k} \left( \frac{\Delta t}{\Delta x} |u_{jk}| + c_{jk} \right) + \frac{8}{3} \frac{\mu_{jk} \Delta t}{\rho_{jk} \Delta x^2} \left( \frac{\Delta t}{\Delta x} |u_{jk}| + c_{jk} \right) + \frac{2 k \mu_{jk} \Delta t}{c_{jk} \rho_{jk} \Delta x^2} \left( \frac{\Delta t}{\Delta x} |u_{jk}| + c_{jk} \right) + \frac{D_{jk} \Delta t}{\Delta x^2} \right) \leq 1
\]

Lehr’s ballistic range experiments

- Spherical-nosed projectile of radius 1.5 mm travels with constant velocity through stoichiometric \( H_2 : O_2 : N_2 \) mixture (molar ratios 2:1:3.76) at 42.663 kPa and \( T = 293 \) K [Lehr, 1972]
- Axisymmetric Navier-Stokes and Eulers simulations on AMR base mesh of 400 \times 200 cells, physical domain size 6 cm \times 3 cm
- 4-level computations with refinement factors 2,2,4 to final time \( t = 170 \mu s \). Refinement downstream removed.
- Main configurations
  - Velocity \( v_i = 1931 \) m/s (\( M = 4.79 \)), \( \sim 40 \) Pts/\( l_g \)
  - Velocity \( v_i = 1806 \) m/s (\( M = 4.48 \)), \( \sim 60 \) Pts/\( l_g \)
- Various previous studies with not entirely consistent results. E.g. [Yungster and Radhakrishnan, 1996], [Axdaal et al., 2011]
- Stagnation point location and pressure tracked in every time step
- All computations were on 32 cores requiring \( \sim 1500 \) h CPU each
Viscous case – $M = 4.79$

- 5619 iterations with CFL=0.9 to $t = 170 \mu s$
- Oscillation frequency in last 20 $\mu s$: $\sim 722$ kHz (viscous), $\sim 737$ kHz (inviscid)
- Experimental value: $\sim 720$ kHz

Viscous case – $M = 4.79$ – mesh adaptation

Comparison of temperature field

Inviscid

Viscous case – $M = 4.48$

- 5432 iterations with CFL=0.9 to $t = 170 \mu s$
- Oscillation frequency in last 20 $\mu s$: $\sim 417$ kHz
- Experimental value: $\sim 425$ kHz
Oscillation mechanism

Inviscid case – $M = 4.48$

- 4048 iterations with CFL=0.9 to $t = 170 \mu$s
- Oscillation frequency in last 20 $\mu$s: $\sim 395$ kHz
- Experimental value: $\sim 425$ kHz

Schlieren plot of density

Perturbed oscillation mechanism

Hybrid method

Convective numerical flux is defined as

$$ F^n_{inv} = \begin{cases} F^n_{inv-WENO^*} & \text{in } C, \\ F^n_{inv-CD^*} & \text{in } C', \end{cases} $$

- For LES: 3rd order WENO method, 2nd order TCD [Hill and Pullin, 2004]
- For DNS: Symmetric 6th order WENO, 6th-order CD scheme [Ziegler et al., 2011]

Use WENO scheme to only capture shock waves but resolve interface between species. Shock detection based on using two criteria together:

1. Lax-Liu entropy condition $|u_R \pm a_R| < |u_\star \pm a_\star| < |u_L \pm a_L|$ tested with a threshold to eliminate weak acoustic waves. Used intermediate states at cell interfaces:

$$ u_\star = \frac{\sqrt{\rho_R u_L + \sqrt{\rho_R \rho_L}}}{\sqrt{\rho_L} + \sqrt{\rho_R}}, \quad a_\star = \sqrt{(\gamma_\star - 1)(h_\star - \frac{1}{2}u_\star^2)}, \ldots $$

2. Limiter-inspired discontinuity test based on mapped normalized pressure gradient

$$ \phi(\theta_j) = \frac{2\theta_j}{(1 + \theta_j)^2} \quad \text{with} \quad \theta_j = \frac{|p_{j+1} - p_j|}{|p_{j+1} + p_j|}, \quad \phi(\theta_j) > \alpha_{Map} $$

Small perturbations can quickly create numerous triple points
**SAMR flux correction for Runge-Kutta method**

Recall Runge-Kutta temporal update

\[ \tilde{Q}^n = \alpha_v Q^n + \beta_v \tilde{Q}^{n-1} + \gamma_v \frac{\Delta t}{\Delta x_0} \Delta F^n(\tilde{Q}^{n-1}) \]

Rewrite scheme as

\[ Q^{n+1} = Q^n - \sum_{i=1}^{Y} \varphi_v \frac{\Delta t}{\Delta x_0} \Delta F^n(\tilde{Q}^{n-1}) \]

with \( \varphi_v = \gamma_v \prod_{i=1}^{Y} \beta_v \)

Flux correction to be used

1. \( \delta F^{1,4}_{i+0.5} := -\varphi_v \frac{F^{1,0}_{i+0.5}}{\Delta x} (\tilde{Q}^n) \), \( \delta F^{1,4}_{i+1} := \delta F^{1,4}_{i+0.5} - \sum_{i-2}^{Y} \varphi_v F^{1,0}_{i+0.5} (\tilde{Q}^{n-1}) \)

2. \( \delta F^{1,4}_{i+0.5} := \delta F^{1,4}_{i+0.5} + \frac{\Delta t}{\Delta x_0} \sum_{i=0}^{Y} \varphi_v F^{1,0}_{i+0.5} (\tilde{Q}^{n-1} + \kappa \Delta t_{i+1}) \)

Storage-efficient SSPRK(3,3):

\[ \begin{array}{cccc} v & 1 & 2 & 3 \\
\alpha_v & \varphi_v & \beta_v & \gamma_v \\
\end{array} \]

[Pantano et al., 2007]

**DNS of shear layer in detonation triple point**

- Calorically perfect two-species model with \( \gamma = 1.29499 \) and \( h_0 = 54,000 \) J/mol and one-step Arrhenius reaction with parameters \( E_a = 30,000 \) J/mol, \( A = 6 \cdot 10^9 \) s\(^{-1} \), \( W = 0.029 \) kg/mol
- 1d ZND theory predicts \( d_{c,1} = 1587.8 \) m/s
- For dynamic viscosity, heat conductivity, and mass diffusion simple Sutherland models are used
- Distance \( L(t) = d_{c,1} \sin(\theta) t \) is used to define a Reynolds number as \( Re = \frac{\gamma \rho_0 L(t)}{\mu} \)
- Viscous shear layer thickness, thermal heat conduction layer thickness, and mass diffusion layer thickness grow as \( \delta_{visc} \approx \sqrt{\frac{F}{F}} t, \delta_{cond} \approx \sqrt{k_{cond} \rho}, \delta_{mass} \approx \sqrt{\frac{D}{F}} t \)
- Only shock thickness not resolved — “pseudo-DNS”
- Computations with WENO/CD/RK3 use SAMR base mesh 320 × 160 and up to 8 levels refined by factor 2, domain: 40 mm × 20 mm
- Computations with MUSCL scheme use base mesh 590 × 352 and up to 7 levels refined by factor 2, domain: 40 mm × 22 mm

**Computational results for shear layer**

<table>
<thead>
<tr>
<th>WENO/CD - 6 levels</th>
<th>WENO/CD - 7 levels</th>
<th>WENO/CD - 8 levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \Delta x_{\text{min}} = 3.91 \cdot 10^{-6} ) m</td>
<td>( \Delta x_{\text{min}} = 1.95 \cdot 10^{-6} ) m</td>
<td>( \Delta x_{\text{min}} = 9.77 \cdot 10^{-7} ) m</td>
</tr>
<tr>
<td>MUSCL - 7 levels</td>
<td>MUSCL - 7 levels - Euler</td>
<td>Usage of WENO for WENO/CD - 8 levels</td>
</tr>
<tr>
<td>( \Delta x_{\text{min}} = 1.05 \cdot 10^{-6} ) m</td>
<td>( \Delta x_{\text{min}} = 1.05 \cdot 10^{-6} ) m</td>
<td></td>
</tr>
</tbody>
</table>

- WENO/CD/RK3 gives results comparable to 4x finer resolved optimal 2nd-order scheme, but CPU times with SAMR 2-3x larger
- Gain in CPU time from higher-order scheme roughly one order

**Favre-averaged Navier-Stokes equations**

\[ \frac{\partial}{\partial t} \tilde{\rho} \tilde{u} + \frac{\partial}{\partial x_k} (\tilde{\rho} \tilde{u}_k \tilde{u}_k + \tilde{\rho}_k \tilde{u}_k - \tilde{\tau}_{kk} + \sigma_{kk}) = 0 \]

\[ \frac{\partial}{\partial t} \tilde{E} + \frac{\partial}{\partial x_k} \left( \tilde{\rho} \tilde{u}_k \tilde{E} + \tilde{\rho} \tilde{E} + \tilde{\rho}_k \tilde{u}_k + \sigma_{kk} \right) = 0 \]

\[ \frac{\partial}{\partial t} (\tilde{\rho} \tilde{Y}) + \frac{\partial}{\partial x_k} (\tilde{\rho} \tilde{Y}_k + \tilde{\rho}_k \tilde{Y} + \sigma_{kk}) = 0 \]

with stress tensor

\[ \tilde{\tau}_{kk} = \tilde{\mu} \left( \frac{\partial \tilde{u}_k}{\partial x_k} + \frac{\partial \tilde{u}_k}{\partial x_k} - \frac{2}{3} \tilde{\rho} \frac{\partial \tilde{\rho}}{\partial x_k} \delta_{kk} \right) \]

Heat conduction

\[ \tilde{q}_k = -\lambda \frac{\partial \tilde{T}}{\partial x_k} \]

and inter-species diffusion

\[ \tilde{J}_k = -\tilde{\rho} \tilde{D}_{ij} \frac{\partial \tilde{Y}_j}{\partial x_k} \]

Favre-filtering

\[ \hat{\phi} = \frac{\phi \tilde{\rho}}{\tilde{\rho}} \] with \( \hat{\phi}(x, t; \Delta_c) = \int_{\Omega} G(x - x' ; \Delta_c) \phi(x', t) dx' \)
Numerical solution approach

- Subgrid terms $\sigma_{\text{sub}}, \sigma_{\text{sub}}^1, \sigma_{\text{sub}}^2$ are computed by Pullin’s stretched-vortex model
- Cutoff $\Delta x_i$ is set to local SAMR resolution $\Delta x_i$
- It remains to solve the Navier-Stokes equations in the hyperbolic regime
  - 3rd order WENO method (hybridized with a tuned centered difference stencil) for convection
  - 2nd order conservative centered differences for diffusion

Example: Cylindrical Richtmyer-Meshkov instability

- Sinusoidal interface between two gases hit by shock wave
- Objective is correctly predict turbulent mixing
- Embedded boundary method used to regularize apex
- AMR base grid $95 \times 95 \times 64$ cells, $n_{1,2,3} = 2$
- $\sim 70,000$ h CPU on 32 AMD 2.5GHz-quad-core nodes

Planar Richtmyer-Meshkov instability

- Perturbed Air-SF6 interface shocked and re-shocked by Mach 1.5 shock
- Containment of turbulence in refined zones
- 96 CPUs IBM SP2-Power3
- WENO-TCD scheme with LES model
- AMR base grid $172 \times 56 \times 56$, $n_{1,2} = 2$, $10M$ cells in average instead of $3M$ (uniform)

<table>
<thead>
<tr>
<th>Task</th>
<th>2ms (%)</th>
<th>5ms (%)</th>
<th>10ms (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integration</td>
<td>45.3</td>
<td>65.9</td>
<td>52.0</td>
</tr>
<tr>
<td>Boundary setting</td>
<td>44.3</td>
<td>28.6</td>
<td>41.9</td>
</tr>
<tr>
<td>Flux correction</td>
<td>7.2</td>
<td>3.4</td>
<td>4.1</td>
</tr>
<tr>
<td>Interpolation</td>
<td>0.9</td>
<td>0.4</td>
<td>0.3</td>
</tr>
<tr>
<td>Reorganization</td>
<td>1.6</td>
<td>1.2</td>
<td>1.2</td>
</tr>
<tr>
<td>Misc</td>
<td>0.6</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>Max. imbalance</td>
<td>1.25</td>
<td>1.23</td>
<td>1.30</td>
</tr>
</tbody>
</table>

References I


References II


Lecture 5

Fluid-structure interaction simulation

Course Block-structured Adaptive Finite Volume Methods for Shock-Induced Combustion Simulation

Outline

Fluid-structure interaction
- Coupling to a solid mechanics solver
- Rigid body motion
- Thin elastic and deforming thin structures
- Deformation from water hammer
- Real-world example

Adaptive Lattice Boltzmann method with FSI
- Adaptive LBM
- Realistic static embedded geometries
- Simulation of wind turbines

Construction of coupling data

Moving boundary/interface is treated as a moving contact discontinuity and represented by level set [Fedkiw, 2002][Arienti et al., 2003]

One-sided construction of mirrored ghost cell and new FEM nodal point values

FEM ansatz-function interpolation to obtain intermediate surface values

Explicit coupling possible if geometry and velocities are prescribed for the more compressible medium [Specht, 2000]

$u^S_n := u^F_n(t) |_I$
$\sigma^S_{nn} := \rho^F(t + \Delta t) |_I$
$\sigma^S_{nm} := 0 |_I$

Coupling conditions on interface

Coupling to a solid mechanics solver

Ralf Deiterding
German Aerospace Center (DLR)
Institute for Aerodynamics and Flow Technology
Bunsenstr. 10, Göttingen, Germany
E-mail: ralf.deiterding@dlr.de

E-mail: ralf.deiterding@dlr.de
Usage of SAMR

- Eulerian SAMR + non-adaptive Lagrangian FEM scheme
- Exploit SAMR time step refinement for effective coupling to solid solver
  - Lagrangian simulation is called only at level \( l \leq l_{\text{max}} \)
  - SAMR refines solid boundary at least at level \( l_c \)
  - Additional levels can be used resolve geometric ambiguities
- Nevertheless: Inserting sub-steps accommodates for time step reduction from the solid solver within an SAMR cycle
- Communication strategy:
  - Updated boundary info from solid solver must be received before regridding operation
  - Boundary data is sent to solid when highest level available
- Inter-solver communication (point-to-point or globally) managed on the fly special coupling module

Fluid and solid update / exchange of time steps

**FluidStep**

\[
\Delta t_F := \min_{i=0, \ldots, l_{\text{max}}} \left( R_i \cdot \text{StableFluidTimeStep}(l), \Delta t_S \right) \\
\Delta t_l := \Delta t_F / R_l \quad \text{for} \quad l = 0, \ldots, L \\
\text{ReceiveInterfaceData}(I_l, u_i^S|I_l) \\
\text{AdvanceLevel}(0)
\]

**SolidStep**

\[
\Delta t_S := \min( K \cdot R_0, \text{StableSolidTimeStep}(), \Delta t_S) \\
\text{Repeat} \quad R_0 \quad \text{times} \\
\text{t}_{\text{end}} := t + \Delta t_S / R_0, \quad \Delta t := \Delta t_S / (KR_0) \\
\text{While} \quad t < t_{\text{end}} \\
\quad \text{SendInterfaceData}(I(t), \bar{u}_i|I(t)) \\
\quad \text{ReceiveInterfaceData}(\rho_i^S|I) \\
\quad \text{UpdateSolid}(\rho_i^S|I, \Delta t) \\
\quad t := t + \Delta t \\
\quad \Delta t := \min(\text{StableSolidTimeStep}(), t_{\text{end}} - t)
\]

with \( R_l = \prod_{i=0}^{l} R_i \)

Parallelization strategy for coupled simulations

**SAMR algorithm for FSI coupling**

**AdvanceLevel(\( l \))**

\[\text{Repeat} \ n \ \text{times} \]

- Set ghost cells of \( Q'(t) \)
- \( \text{CPT}(\rho', C', I, \bar{u}) \)
- \( \text{If to regrid?} \)
- \( \text{Regrid}(l) \)
- \( \text{UpdateLevel}(l) \)
- \( \text{If level} \ l + 1 \ \text{exists?} \)
  - Set ghost cells of \( Q'(t + \Delta t_l) \)
  - \( \text{Average} \ Q^{l+1}(t + \Delta t_l) \) \( \rightarrow \) \( Q'(t + \Delta t_l) \)
- \( \text{If} \ l = l_c? \)
- \( \text{SendInterfaceData}(\rho_i^S(t + \Delta t_l)|I) \)
- \( \text{If} \ (t + \Delta t_l) < (t_0 + \Delta t_0)? \)
- \( \text{ReceiveInterfaceData}(I, u_i^S|I) \)
- \( t := t + \Delta t_l \)

**Fluid and solid data exchange**

**Coupling of an Eulerian FV fluid Solver and a Lagrangian FEM Solver:**

- Distribute both meshes seperately and copy necessary nodal values and geometry data to fluid nodes
- Setting of ghost cell values becomes strictly local operation
- Construct new nodal values strictly local on fluid nodes and transfer them back to solid nodes
- Only surface data is transferred
- Asynchronous communication ensures scalability
- Generic encapsulated implementation guarantees reusability
**Eulerian/Lagrangian communication module**

1. Put bounding boxes around each solid processors piece of the boundary and around each fluid processors grid
2. Gather, exchange and broadcast of bounding box information
3. Optimal point-to-point communication pattern, non-blocking

**Coupling elements**

**Boundary Conditions:**
- Inlet (U), Outlet (U), Walls, Velocities
- Initial Conditions: Density, Pressure, Velocity

**Solid Domain**

**Fluid Domain**

**Proximal bodies in hypersonic flow**

Flow modeled by Euler equations for a single polytropic gas with \(\rho = (\gamma - 1) \rho_e\)

\[
\partial_t p + \partial_x (\rho u) = 0, \quad \partial_t (\rho u) + \partial_x (\rho u u + \delta_{kn} p) = 0, \quad \partial_t (\rho E) + \partial_x (\rho u (\rho E + p)) = 0
\]

Numerical approximation with

- Finite volume flux-vector splitting scheme with MUSCL reconstruction, dimensional splitting
- Spherical bodies, force computation with overlaid latitude-longitude mesh to obtain drag and lift coefficients \(C_D, L = \frac{2F_{D, L}}{\rho v^2 \pi r^2}\)
- Inflow \(M = 10\), \(C_D\) and \(C_L\) on secondary sphere, lateral position varied, no motion

**Lift-up of a spherical body**

Cylindrical body hit by Mach 3 shockwave, 2D test case by [Falcovitz et al., 1997]

Schlieren plot of density

Refinement levels

[vtf/amrc/clawpack/applications/euler/2d/SphereLiftOff]
Verification and validation

Static force measurements, $M = 10$:
[Laurence et al., 2007]

- Refined study: $40 \times 40 \times 32$ base grid, up to without AMR up to $\sim 209,7 \cdot 10^6$ cells, largest run $\sim 35,000 \ h \ CPU$

$\begin{array}{c|c|c|c|}
\textbf{Imax} & \textbf{Cp} & \textbf{\Delta Cp} & \textbf{Ct} & \textbf{\Delta Ct} \\
1 & 1.264 & 0.176 & -0.019 & 1.57 \\
2 & 1.423 & -0.019 & 0.052 & 0.157 \\
3 & 1.408 & -0.015 & 0.087 & 0.035 \\
\end{array}$

- Comparison with experimental results: 3 additional levels, $\sim 2000 \ h \ CPU$

<table>
<thead>
<tr>
<th>Experimental</th>
<th>Computational</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_D$</td>
<td>$C_D$</td>
</tr>
<tr>
<td>1.11 ± 0.08</td>
<td>1.01</td>
</tr>
<tr>
<td>0.29 ± 0.05</td>
<td>0.28</td>
</tr>
</tbody>
</table>

Dynamic motion, $M = 4$:
- Base grid $150 \times 125 \times 90$, two additional levels with $n_{1,2} = 2$
- 24,704 time steps, 36,808 h CPU on 256 cores IBM BG/P

Treated with experimental results: 3 additional levels, $\sim 2000 \ h \ CPU$

<table>
<thead>
<tr>
<th>Experimental</th>
<th>Computational</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_D$</td>
<td>$C_D$</td>
</tr>
<tr>
<td>$1.11 \pm 0.08$</td>
<td>$1.01$</td>
</tr>
<tr>
<td>$0.29 \pm 0.05$</td>
<td>$0.28$</td>
</tr>
</tbody>
</table>

Schlieren graphics on refinement regions

Treatment of thin structures

- Thin boundary structures or lower-dimensional shells require “thickening” to apply embedded boundary method
- Unsigned distance level set function $\varphi$
- Treat cells with $0 < \varphi < d$ as ghost fluid cells
- Leaving $\varphi$ unmodified ensures correctness of $\nabla \varphi$
- Use face normal in shell element to evaluate in $\Delta p = p^+ - p^-$
- Utilize finite difference solver using the beam equation
  $$\rho_s h \frac{\partial^2 w}{\partial t^2} + EI \frac{\partial^4 w}{\partial x^4} = p_F$$

FSI verification by elastic vibration

- Thin steel plate (thickness $h = 1 \ mm$, length $50 \ mm$), clamped at lower end
- $\rho_s = 7600 \ kg/m^3$, $E = 220 \ GPa$, $I = h^3/12$, $\nu = 0.3$
- Modeled with beam solver (101 points) and thin-shell FEM solver (325 triangles) by F. Cirak
- Left: Coupling verification with constant instananeous loading by $\Delta p = 100 \ kPa$
- Right: FSI verification with Mach 1.21 shockwave in air ($\gamma = 1.4$)
**Fluid-structure interaction**

Adaptive Lattice Boltzmann method with FSI

Thin elastic and deforming thin structures

---

**Shock-driven elastic panel motion**

Test case suggested by [Giordano et al., 2005]

- Forward facing step geometry, fixed walls everywhere except at inflow

<table>
<thead>
<tr>
<th>Time after passage of transducer 1 [ms]</th>
<th>Pressure MPa</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.56</td>
<td>1.6458 kg/m²</td>
</tr>
<tr>
<td>2.10</td>
<td>1.2 kg/m²</td>
</tr>
<tr>
<td>3.5</td>
<td>156.18 kPa</td>
</tr>
</tbody>
</table>

- SAMR base mesh 320 × 64(×2), \( n_x = 2 \)
- Intel 3.4GHz Xeon dual processors, GB Ethernet interconnect
  - Beam-FSI: 12.25 h CPU on 3 fluid CPU + 1 solid CPU
  - FEM-FSI: 322 h CPU on 14 fluid CPU + 2 solid CPU

**Detonation-driven plastic deformation**

Chapman-Jouguet detonation in a tube filled with a stoichiometric ethylene and oxygen \((C_2H_4 + 3 O_2, 295 K)\) mixture. Euler equations with single exothermic reaction \(A \rightarrow B\)

\[
\frac{\partial\rho}{\partial t} + \frac{\partial (\rho u_k)}{\partial x_k} = 0, \quad \frac{\partial\rho u_k}{\partial t} + \frac{\partial (\rho u_k^2 + p)}{\partial x_k} = 0, \quad k = 1, \ldots, d
\]

\[
\frac{\partial p}{\partial t} + \frac{\partial (pE + p)}{\partial x_k} = 0, \quad \frac{\partial (Yp)}{\partial t} + \frac{\partial (Yρu_k)}{\partial x_k} = ψ
\]

with

\[
p = (γ - 1)(pE - \frac{1}{2}ρu_k u_k - ρYq_0) \quad \text{and} \quad ψ = kYρ \exp \left( -\frac{E_{Aρ}}{p} \right)
\]

modeled with heuristic detonation model by [Mader, 1979]

\[
V' := 1 - \frac{(V_c - V)}{V_c} \quad Y' := \frac{y}{y - 1}
\]

If \(0 < Y' < 1\) and \(Y > 10^{-8}\) then

- If \(Y < Y'\) and \(Y' < 0.9\) then \(Y' := 0\)
- If \(Y' < 0.9\) then \(p' := (1 - Y')ρ_{CJ}\)
- else \(p' := p\)

\[
ρ_{A} := Y\rho
\]

\[
E := \frac{p'}{(p(γ - 1))} + Y'q_0 + \frac{1}{2}u_ku_k
\]

**Tube with flaps**

- Fluid: VanLeer FVS
  - Detonation model with \(γ = 1.24\), \(p_{CJ} = 3.3\) MPa, \(D_{CJ} = 2376\) m/s
  - AMR base level: 104 × 80 × 242, \(n_x = 2\), \(n_z = 4\)
  - \(≈ 4 \times 10^7\) cells instead of \(7.9 \times 10^8\) cells (uniform)
  - Tube and detonation fully refined
  - Thickening of 2D mesh: 0.81 mm on both sides (real 0.445 mm)

- Solid: thin-shell solver by F. Cirak
  - Aluminum, J2 plasticity with hardening, rate sensitivity, and thermal softening
  - Mesh: 8577 nodes, 17056 elements

- 16+2 nodes 2.2 GHz AMD Opteron quad processor, PCI-X 4x Infiniband network, \(≈ 4320\) h CPU to \(t_{end} = 450\) μs

---

**Tube with flaps: results**

- Fluid density and displacement in Y-direction in solid
- Schlieren plot of fluid density on refine-ment levels

[Cirak et al., 2007] vtf/fsi/amroc/TubeCJBurnFlaps - Fluid, Solid
**Coupled fracture simulation**

- Use HLLC approach because of robustness and positivity preservation

\[
q^{HLLC}(x_1, t) = \begin{cases} 
q_L, & x_1 < s_L t, \\
q_s^*, & s_L t \leq x_1 < s^* t, \\
q_R, & s^* t \leq x_1 \leq s_R t, \\
q_R^*, & x_1 > s_R t.
\end{cases}
\]

- Wave speed estimates [Davis, 1988]

\[
s_L = \min(u_{i,1} - c_L, u_{i,1,R} - c_R),
\]

\[
s_R = \max(u_{i,1} + c_L, u_{i,1,R} + c_R)
\]

- Unknown state [Toro et al., 1994]

\[
s^* = \frac{\rho_R - \rho_L + s_R u_{i,1,R}(s_L - u_{i,1,R}) - \rho_R u_{i,1,R}(s_L - u_{i,1,R})}{\rho_L(s_L - u_{i,1,L}) - \rho_R(s_L - u_{i,1,R})}
\]

- \[
q^*_{s^*} = \left[ \frac{(\rho E)^*}{\rho^*} + (s^* - u_{i,1,\tau}), \frac{\rho^*}{\rho_{s^* - u_{i,1,\tau}}} \left( s_{s^* - u_{i,1,\tau}} + \frac{\rho_{s^* - u_{i,1,\tau}}}{\rho_{s^* - u_{i,1,\tau}}} \right) \right], \frac{1}{\gamma_{s^* - u_{i,1,\tau}} - 1}, \frac{\gamma_{p_{s^* - u_{i,1,\tau}}} - 1}{\gamma_{s^* - u_{i,1,\tau}} - 1}^T
\]

\[
\eta = \rho^* s_{s^* - u_{i,1,\tau}} - s^*, \sigma = (L, R)
\]

- Evaluate waves as 

\[
\psi_1 = q_L^* - q_L, \psi_2 = q_R^* - q_L, \psi_3 = q_R^* - q_R^* \text{ and } \lambda_1 = s_L, 
\]

\[
\lambda_2 = s^*, \lambda_3 = s_R \text{ to compute the fluctuations } A^T \Delta = \sum_{\nu} A^T \nu \psi_\nu,
\]

- Overall scheme: Wave Propagation method [Shyue, 2006]

**Underwater explosion modeling**

- Volume fraction based two-component model with \( \sum_{i=1}^{m} \alpha_i = 1 \), that defines mixture quantities as

\[
\rho = \sum_{i=1}^{m} \alpha_i \rho^i, \quad \rho u = \sum_{i=1}^{m} \alpha_i \rho^i u, \quad \rho e = \sum_{i=1}^{m} \alpha_i \rho^i e^i
\]

Assuming total pressure \( p = (\gamma - 1) \rho e - \gamma p_\infty \) and speed of sound \( c = (\gamma (\rho + p_\infty)/\rho)^{1/2} \) yields

\[
\frac{\partial p}{\partial \gamma - 1} = \sum_{i=1}^{m} \alpha_i \rho^i, \quad \frac{\gamma p_\infty}{\gamma - 1} = \sum_{i=1}^{m} \alpha_i \rho^i
\]

and the overall set of equations [Shyue, 1998]

\[
\frac{\partial}{\partial t} \left( \frac{1}{\gamma - 1} \right) + u_\nu \frac{\partial}{\partial x_\nu} \left( \frac{1}{\gamma - 1} \right) = 0, \quad \frac{\partial}{\partial t} \left( \frac{\gamma p_\infty}{\gamma - 1} + u_\nu \frac{\partial}{\partial x_\nu} \left( \frac{\gamma p_\infty}{\gamma - 1} \right) = 0
\]

Oscillation free at contacts: [Abgrall and Karni, 2001][Shyue, 2006]

**Approximate Riemann solver**

**Underwater explosion FSI simulations**

- Air: \( \gamma^A = 1.4, \rho^A = 0, \rho^A = 1.29 \text{ kg/m}^3 \)
- Water: \( \gamma^W = 7.415, \rho^W = 296.2 \text{ MPa}, \rho^W = 1027 \text{ kg/m}^3 \)
- Cavitation modeling with pressure cut-off model at \( p = -1 \text{ MPa} \)
- 3D simulation of deformation of air backed aluminum plate with \( r = 85 \text{ mm}, h = 3 \text{ mm} \) from underwater explosion
  - Water basin [Ashani and Ghamsari, 2008] \( 2 \text{ m} \times 1.6 \text{ m} \times 2 \text{ m} \)
  - Explosion modeled as energy increase (\( mC_4 = 6.06 \text{ MJ/kg} \)) in sphere with \( r = 5 \text{ mm} \)
  - \( \rho_\alpha = 2719 \text{ kg/m}^3, E = 69 \text{ GPa}, \nu = 0.33, J2 plasticity model, yield stress } \sigma_\tau = 217.6 \text{ MPa} \)
- 3D simulation of copper plate \( r = 32 \text{ mm}, h = 0.25 \text{ mm} \) rupturing due to water hammer
  - Water-filled shocktube 1.3 m with driver piston [Deshpande et al., 2006]
  - Piston simulated with separate level set, see [Deiterding et al., 2009] for pressure wave
  - \( \rho_\alpha = 8920 \text{ kg/m}^3, E = 130 \text{ GPa}, \nu = 0.31, J2 plasticity model, } \sigma_\tau = 38.5 \text{ MPa}, \text{ cohesive interface model, max. tensile stress } \sigma_\tau = 525 \text{ MPa} \)
Underwater explosion simulation

- AMR base grid $50 \times 40 \times 50$, $n_{r,2,3} = 2$, $n_4 = 4$, $l_c = 3$, highest level restricted to initial explosion center, 3rd and 4th level to plate vicinity
- Triangular mesh with 8148 elements
- Computations of 1296 coupled time steps to $t_{end} = 1$ ms
- 10+2 nodes 3.4 GHz Intel Xeon dual processor, $\sim 130$ h CPU

<table>
<thead>
<tr>
<th>Maximal deflection [mm]</th>
<th>Exp.</th>
<th>Sim.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$20 \text{ g, } d = 25 \text{ cm}$</td>
<td>28.83</td>
<td>25.88</td>
</tr>
<tr>
<td>$30 \text{ g, } d = 30 \text{ cm}$</td>
<td>30.09</td>
<td>27.31</td>
</tr>
</tbody>
</table>

Plate in underwater shocktube

- AMR base mesh $374 \times 20 \times 20$, $n_{r,2} = 2$, $l_c = 2$, solid mesh: 8896 triangles
- $\sim 1250$ coupled time steps to $t_{end} = 1$ ms
- 6+6 nodes 3.4 GHz Intel Xeon dual processor, $\sim 800$ h CPU

Blast explosion in a multistory building

- $20 \text{ m} \times 40 \text{ m} \times 25 \text{ m}$ seven-story building similar to [Luccioni et al., 2004]
- Spherical energy deposition $\equiv 400$ kg TNT, $r = 0.5$ m in lobby of building
- SAMR: $80 \times 120 \times 90$ base level, three additional levels $n_{r,2} = 2$, $l_{lib} = 1$, $k = 1$
- Simulation with ground: 1, 070 coupled time steps, 830 h CPU ($\sim 25.9$ h wall time) on 31+1 cores
- $\sim 8,000,000$ cells instead of 55,296,000 (uniform)
- 69,709 hexahedral elements with material parameters. [Deiterding and Wood, 2013]

<table>
<thead>
<tr>
<th>Columns</th>
<th>$\rho_0$ [kg/m$^3$]</th>
<th>$\sigma_0$ [MPa]</th>
<th>$E_T$ [GPa]</th>
<th>$\beta$</th>
<th>$K$ [GPa]</th>
<th>$G$ [GPa]</th>
<th>$\bar{\epsilon}$</th>
<th>$\rho_0$ [MPa]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Walls</td>
<td>2010</td>
<td>50</td>
<td>11.2</td>
<td>1.0</td>
<td>21.72</td>
<td>4.67</td>
<td>0.02</td>
<td>-30</td>
</tr>
<tr>
<td></td>
<td>2010</td>
<td>25</td>
<td>11.2</td>
<td>1.0</td>
<td>6.22</td>
<td>4.67</td>
<td>0.01</td>
<td>-15</td>
</tr>
</tbody>
</table>

Blast explosion in a multistory building – II

- $p_0 = 64 \text{ MPa}$
- $p_0 = 173 \text{ MPa}$

\[ t = 0 \]
**Lattice Boltzmann method**

Boltzmann equation: \( \partial_t f + u \cdot \nabla f = \omega (f^{eq} - f) \)

Two-dimensional LBM for weakly compressible flows

Formulated on FV grids! (\( \rightarrow \) boundary conditions!)

\[
\rho(x, t) = \sum_{\alpha=0}^{8} f_{\alpha}(x, t), \quad \rho(x, t)u(x, t) = \sum_{\alpha=0}^{8} e_{\alpha}f_{\alpha}(x, t)
\]

1. Transport step \( T \): \( \tilde{f}_\alpha(x + e_\alpha \Delta t, t + \Delta t) = f_{\alpha}(x, t) \)

2. Collision step \( C \):

\[
f_{\alpha}(\cdot, t + \Delta t) = f_{\alpha}(\cdot, t + \Delta t) + \omega \Delta t \left( f^{eq}_{\alpha}(\cdot, t + \Delta t) - \tilde{f}_{\alpha}(\cdot, t + \Delta t) \right)
\]

with equilibrium function

\[
f^{eq}_{\alpha}(\rho, u) = \rho \tau_{\alpha} \left[ 1 + \frac{e_\alpha u}{c^2} + \frac{(e_\alpha u)^2}{2c^2} - \frac{u^2}{2c^2} \right]
\]

mit \( \tau_{\alpha} = \frac{1}{9} \left\{ 4, 1, 1, 1; \frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3} \} \)

Lattice speed of sound: \( c_s = \frac{1}{\sqrt{\gamma}} \Delta t \), pressure \( p = \sum_{\alpha} f^{eq}_{\alpha} c^2_s = p c^2_s = \rho R T \)

Collision frequency vs. kinematic viscosity: \( \omega = \frac{c^2_s}{\nu + \Delta t c^2_s/2} \) cf. [Hähnel, 2004]

**Adaptive LBM**

1. Complete update on coarse grid: \( f_{\alpha,n+1}^C := CT(f_{\alpha,n}^C) \)
2. Interpolate \( f_{\alpha,n}^C \) onto \( f_{\alpha,n}^F \) to fill fine halos. Set physical boundary conditions.
3. \( \tilde{f}_{\alpha,n}^F := T(f_{\alpha,n}^F) \) on whole fine mesh. \( f_{\alpha,n+1/2}^F := C(f_{\alpha,n}^F) \) in interior.
4. \( f_{\alpha,n+1/2}^F := T(f_{\alpha,n+1/2}^F) \) on whole fine mesh. \( f_{\alpha,n+1}^F := C(f_{\alpha,n+1/2}^F) \) in interior.

5. Average \( f_{\alpha,n+1/2}^F \) (inner halo layer), \( f_{\alpha,n+1/2}^C \) (outer halo layer) to obtain \( f_{\alpha,n}^C \)

6. Revert transport into halos: \( f_{\alpha,n+1/2}^C \rightarrow f_{\alpha,n}^C \)

7. Parallel synchronization of \( f_{\alpha,n+1/2}^C \)

8. Cell-wise update where correction is needed: \( f_{\alpha,n+1}^C := CT(f_{\alpha,n}^C \cdot f_{\alpha,n+1}^C) \)

Algorithm equivalent to [Chen et al., 2006].

**Verification - driven cavity**

- \( Re = 1500 \) in air, \( \nu = 1.5 \cdot 10^{-5} \) m²/s, \( u = 22.5 \) m/s.
- Domain size \( 1 \) mm \( \times \) \( 1 \) mm.
- Reference computation uses \( 800 \times 800 \) lattice.
- \( 588,989 \) time steps to \( t_e = 5 \cdot 10^{-3} \) s, \( \sim 35 \) h CPU.
- Statically adaptive computation uses \( 100 \times 100 \) lattice with \( r_{1.2} = 2 \).
- \( 294,452 \) time steps to \( t_e = 5 \cdot 10^{-3} \) s on finest level.

Isolines of density. Left: reference, right on refinement at \( t_e \).

**Driven cavity - dynamic refinement**

- Dynamic refinement based on heuristic error estimation of \( |u| \)
- Threshold intentionally chosen to show refinement evolution

- vffmpeg/annroc/lbm/applications/Navier-Stokes/2d/Cavity

Isolines of density on refinement (left), distribution to 4 processors (right).
Side-wind investigation for a train model

Complex boundary consideration with level set method
► Construction of macro-values in embedded cells by inter-/ extrapolation.
► Then use $f_{eq}^{\alpha}(\rho', u')$ to construct distributions in embedded ghost cells.
► 2nd order improvements possible, cf. [Peng and Luo, 2008].

Typical DLR problem
► 1:25 train model represented with 74,670 triangles (41,226 front body, 12,398 back body, 21,006 blade)
► Wind tunnel conditions: air at room temperature with 60.25 m/s ($M = 0.18$), $Re = 450,000$
► Systematic side wind investigation with $0 \geq \beta \geq 30^\circ$ to obtain lift, drag and roll moment coefficients
► vtf/amroc/lbm/applications/Navier-Stokes/3d/NGT2

Flow prediction, $Re = 450,000, \beta = 30^\circ$
► Domain 10 m $\times$ 2.4 m $\times$ 1.6 m
► Computation started in 3 steps. Full resolution after 5889 coarsest level steps or $t \geq 0.4$ s
► $\sim$ 1140 coarsest level steps in 24 h on 96 cores shown above. Overall cost $\sim$ 4600 h CPU.

Vorticity vector component perpendicular to middle axis.

Dynamic mesh adaptation
► Base mesh 500 $\times$ 120 $\times$ 80 cells, refinement factors 2,2,4.
► Refinement based on error estimation of $|u|$ up to second highest level.
► Highest level reserved to geometry refinement with $\Delta x = 1.25$ mm.

Simulation of a single turbine
► Geometry from realistic Vestas V27 turbine. Rotor diameter 27 m, tower height $\sim$ 35 m. Ground considered.
► Prescribed motion of rotor with 15 rpm. Inflow velocity 7 m/s.
► Simulation domain 200 m $\times$ 100 m $\times$ 100 m.
► Base mesh 400 $\times$ 200 $\times$ 200 cells with refinement factors 2,2,4. Resolution of rotor and tower $\Delta x = 3.125$ cm.
► 141,344 highest level iterations to $t_e = 30$ s computed.

Dynamically adapting mesh. View in wind direction.
Wake field behind turbine

Adaptive refinement

Preliminary simulation of the SWIFT array

Wakes in SWIFT array (preliminary)
References I


References II


References III


References IV


Lecture 6
The AMROC software system

Course Block-structured Adaptive Finite Volume Methods for Shock-Induced Combustion Simulation

Available SAMR software
- AMROC
- Massively parallel SAMR

References

Outline

Available SAMR software
- Simplified block-based AMR
- General patch-based SAMR

AMROC
- Overview
- Layered software structure

Massively parallel SAMR
- Performance data from AMROC

Simplified structured designs

Distributed memory parallelization fully supported if not otherwise stated.

- PARAMESH (Parallel Adaptive Mesh Refinement)
  - Library based on uniform refinement blocks [MacNeice et al., 2000]
  - Both multigrid and explicit algorithms considered
  - http://sourceforge.net/projects/paramesh
- Flash code (AMR code for astrophysical thermonuclear flashes)
  - Built on PARAMESH
  - Solves the magneto-hydrodynamic equations with self-gravitation
  - http://www.flash.uchicago.edu/site/flashcode
- Uintah (AMR code for simulation of accidental fires and explosions)
  - Only explicit algorithms considered
  - FSI coupling Material Point Method and ICE Method (Implicit, Continuous fluid, Eulerian)
  - http://www.uintah.utah.edu
- DAGH/Grace [Parashar and Browne, 1997]
  - Just C++ data structures but no methods
  - All grids are aligned to bases mesh coarsened by factor 2
  - http://userweb.cs.utexas.edu/users/dagh

Systems that support general SAMR

- SAMRAI - Structured Adaptive Mesh Refinement Application Infrastructure
  - Very mature SAMR system [Hornung et al., 2006]
  - Explicit algorithms directly supported, implicit methods through interface to Hypre package
  - Mapped geometry and some embedded boundary support
- BoxLib, AmrLib, MGLib, HGProj
  - Berkley-Lab-AMR collection of C++ classes by J. Bell et al., 50,000 LOC [Rendleman et al., 2000]
  - Both multigrid and explicit algorithms supported
- Chombo
  - Redesign and extension of BoxLib by P. Colella et al.
  - Both multigrid and explicit algorithms demonstrated
  - Some embedded boundary support
  - https://commons.lbl.gov/display/chombo
Further SAMR software

- Overture (Object-oriented tools for solving PDEs in complex geometries)
  - Overlapping meshes for complex geometries by W. Henshaw et al. [Brown et al., 1997]
  - Explicit and implicit algorithms supported
  - http://www.overtureframework.org
- AMRClaw within Clawpack [Berger and LeVeque, 1998]
  - Serial 2D Fortran 77 code for the explicit Wave Propagation method with own memory management
  - http://depts.washington.edu/clawpack
- Amrita by J. Quirk
  - Only 2D explicit finite volume methods supported
  - Embedded boundary algorithm
  - http://www.amrita-cfd.org
- Cell-based Cartesian AMR: RAGE
  - Embedded boundary method
  - Explicit and implicit algorithms
  - [Gittings et al., 2008]

AMROC

- “Adaptive Mesh Refinement in Object-oriented C++”
- ~ 46,000 LOC for C++ SAMR kernel, ~140,000 total C++, C, Fortran-77
- uses parallel hierarchical data structures that have evolved from DAGH
- Right: point explosion in box, 4 level, Euler computation, 7 compute nodes
- V1.0: http://amroc.sourceforge.net

The Virtual Test Facility

- Implements all described algorithms beside multigrid methods
- AMROC V2.0 plus solid mechanics solvers
- Implements explicit SAMR with different finite volume solvers
- Embedded boundary method, FSI coupling
- ~ 430,000 lines of code total in C++, C, Fortran-77, Fortran-90
- autoconf / automate environment with support for typical parallel high-performance system
- http://www.cacr.caltech.edu/asc
- [Deiterding et al., 2006][Deiterding et al., 2007]

The AMROC software system

<table>
<thead>
<tr>
<th>Available SAMR software</th>
<th>AMROC</th>
<th>Massively parallel SAMR</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Overview

The Virtual Test Facility

- Implements all described algorithms beside multigrid methods
- AMROC V2.0 plus solid mechanics solvers
- Implements explicit SAMR with different finite volume solvers
- Embedded boundary method, FSI coupling
- ~ 430,000 lines of code total in C++, C, Fortran-77, Fortran-90
- autoconf / automate environment with support for typical parallel high-performance system
- http://www.cacr.caltech.edu/asc
- [Deiterding et al., 2006][Deiterding et al., 2007]
Commonalities in software design

- Index coordinate system based on $\Delta x_{m,l} = \prod_{K=N+l+1}^{K_{max}} r_k$ to uniquely identify a cell within the hierarchy
- Box<dim>, BoxList<dim> class that define rectangular regions $G_{m,l}$ by lowerleft, upperright, stepsize and specify topological operations $\cap$, $\cup$
- Patch<dim,type> class that assigns data to a rectangular grid $G_{m,l}$
- A class, here GridFunction<dim,type>, that defines topological relations between lists of Patch objects to implement synchronization, restriction, prolongation, re-distribution
- Hierarchical parallel data structures are typically C++, routines on patches often Fortran

Embedded boundary method / FSI coupling

- Multiple independent EmbeddedBoundaryMethod objects possible
- Specialization of GFMs boundary conditions, level set description in scheme-specific F77 interface classes

Parallelized construction of space-filling curve

Computation of space filling curve

- Partition-Init
  1. Compute aggregated workload for new grid hierarchy and project result onto level 0
  2. Construct recursively SFC-units until work in each unit is homogeneous, GuCFactor defines minimal coarseness relative to level-0 grid
- Partition-Calc
  1. Compute entire workload and new work for each processor
  2. Go sequentially through SFC-ordered list of partitioning units and assign units to processors, refine partition if necessary and possible
- Ensure scalability of Partition-Init by creating SFC-units strictly local
- Currently still use of MPI_allgather() to create globally identical input for Partition-Calc (can be a bottleneck for weak scalability)

Partitioning example

- Cylinders of spheres in supersonic flow
- Predict force on secondary body
- Right: 200x160 base mesh, 3 Levels, factors 2,2,2, 8 CPUs

[Laurence et al., 2007]
## First performance assessment

- Test run on 2.2 GHz AMD Opteron quad-core cluster connected with Infiniband
- Cartesian test configuration
- Spherical blast wave, Euler equations, 3rd order WENO scheme, 3-step Runge-Kutta update
- AMR base grid $64^3$, $n_i = 2$, 89 time steps on coarsest level
- With embedded boundary method: 96 time steps on coarsest level
- Redistribute in parallel every 2nd base level step
- Uniform grid $256^3 = 16.8 \times 10^6$ cells

<table>
<thead>
<tr>
<th>Level</th>
<th>Grids</th>
<th>Cells</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>115</td>
<td>262,144</td>
</tr>
<tr>
<td>1</td>
<td>373</td>
<td>1,589,808</td>
</tr>
<tr>
<td>2</td>
<td>2282</td>
<td>5,907,064</td>
</tr>
</tbody>
</table>

Grids and cells used on 16 CPUs

The AMROC software system

### AMROC scalability tests

Basic test configuration
- Spherical blast wave, Euler equations, 3D wave propagation method
- AMR base grid $32^3$ with $n_i = 2$, 5 time steps on coarsest level
- Uniform grid $256^3 = 16.8 \times 10^6$ cells, 19 time steps
- Flux correction deactivated
- No volume I/O operations
- Tests run IBM BG/P (mode VN)

Weak scalability test
- Reproduction of configuration each 64 CPUs
- On 1024 CPUs: $128 \times 64 \times 64$ base grid, $>33,500$ Grids, $\sim 61 \cdot 10^6$ cells, uniform $1024 \times 512 \times 512 = 268 \cdot 10^6$ cells

<table>
<thead>
<tr>
<th>Level</th>
<th>Grids</th>
<th>Cells</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>606</td>
<td>32,768</td>
</tr>
<tr>
<td>1</td>
<td>975</td>
<td>135,312</td>
</tr>
<tr>
<td>2</td>
<td>910</td>
<td>3,639,040</td>
</tr>
</tbody>
</table>

Strong scalability test
- $64 \times 32 \times 32$ base grid, uniform $512 \times 256 \times 256 = 33.6 \cdot 10^6$ cells

<table>
<thead>
<tr>
<th>Level</th>
<th>Grids</th>
<th>Cells</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1709</td>
<td>65,536</td>
</tr>
<tr>
<td>1</td>
<td>1735</td>
<td>271,048</td>
</tr>
<tr>
<td>2</td>
<td>2210</td>
<td>7,190,208</td>
</tr>
</tbody>
</table>

## Cost of SAMR and ghost-fluid method

- Flux correction is negligible
- Clustering is negligible (already local approach). For the complexities of a scalable global clustering algorithm see [Gunney et al., 2007]
- Costs for GFM constant around $\sim 36$
- Main costs: Regrid(1) operation and ghost cell synchronization

<table>
<thead>
<tr>
<th>CPUs</th>
<th>Time per step (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>32.44</td>
</tr>
<tr>
<td>32</td>
<td>18.65</td>
</tr>
<tr>
<td>64</td>
<td>11.87</td>
</tr>
</tbody>
</table>

| Costs for Syncing basically constant
| Partitioning, Recompose, Misc (origin not clear) increase
| 1024 required usage of -DUAL option due to usage of global lists data structures in Partition-Calc and Recompose

## Weak scalability test

### Time per highest level step

<table>
<thead>
<tr>
<th>CPUs</th>
<th>Time per step (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>35.94</td>
</tr>
<tr>
<td>128</td>
<td>24.83</td>
</tr>
<tr>
<td>256</td>
<td>16.21</td>
</tr>
</tbody>
</table>

| Costs for GFM constant
| Syncing, Recompose, Misc (origin not clear) increase
| 1024 required usage of -DUAL option due to usage of global lists data structures in Partition-Calc and Recompose

### Breakdown of time per step with SAMR

- Integration, Syncing, Partition, Recompose, Misc
Available SAMR software AMROC Massively parallel SAMR References

Performace data from AMROC

Strong scalability test

- Uniform code has basically linear scalability (explicit method)
- SAMR visibly looses efficiency for > 512 CPU, or 15,000 finite volume cells per CPU

Strong scalability test - II

- Perfect scaling of Integration, reasonable scaling of Syncing
- Strong scalability of Partition needs to be addressed (eliminate global lists)

The AMROC software system

Available SAMR software AMROC Massively parallel SAMR References

Performance data from AMROC

Strong scalability test - Train side wind computation

- Computation is restarted from disk checkpoint at \( t = 0.526408 \) s.
- Time for initial re-partitioning removed from benchmark.
- 200 coarse level time steps computed.
- Refridding and re-partitioning every 2nd level-0 step.
- Computation starts with 51.8M cells (l3: 10.2M, l2: 15.3M, l1: 21.5M, l0= 4.8M) vs. 19.66 billion (uniform).
- Portions for parallel communication quite considerable (4 ghost cells still used).

The AMROC software system

Available SAMR software AMROC Massively parallel SAMR References

Performance data from AMROC

References


Supplementary material: Using the SAMR approach for elliptic problems

Course *Block-structured Adaptive Finite Volume Methods for Shock-Induced Combustion Simulation*

**Lecture 8**

**Supplementary material: Using the SAMR approach for elliptic problems**

**Outline**

- Adaptive geometric multigrid methods
  - Linear iterative methods for Poisson-type problems
  - Multi-level algorithms
  - Multigrid algorithms on SAMR data structures
  - Example

- Comments on parabolic problems

**The AMROC software system**

Adaptive geometric multigrid methods

Comments on parabolic problems

**References**


Poisson equation

\[ \Delta q(x) = \psi(x), \quad x \in \Omega \subset \mathbb{R}^d, \quad q \in C^2(\Omega), \quad \psi \in C^0(\Omega) \]

\[ q = \psi'(x), \quad x \in \partial \Omega \]

Discrete Poisson equation in 2D:

\[ \frac{Q_{i+1,k} - 2Q_{i,k} + Q_{i-1,k}}{\Delta x_1^2} + \frac{Q_{j,k+1} - 2Q_{j,k} + Q_{j,k-1}}{\Delta x_2^2} = \psi_{jk} \]

Operator

\[ A(Q_{\Delta x_1, \Delta x_2}) = \left[ \frac{1}{\Delta x_1^2} - \left( \frac{2}{\Delta x_1^2} \right) \frac{1}{\Delta x_2^2} \right] Q(x_{1,j}, x_{2,k}) = \psi_{jk} \]

\[ Q_{jk} = \frac{1}{\sigma} \left( (Q_{i+1,k} + Q_{i-1,k}) \Delta x_1^2 + (Q_{j,k+1} + Q_{j,k-1}) \Delta x_2^2 - \Delta x_1^2 \Delta x_2^2 \psi_{jk} \right) \]

with \( \sigma = \frac{2\Delta x_1^2 + 2\Delta x_2^2}{\Delta x_1^2 \Delta x_2^2} \)

Smoothing vs. solving

\( \nu \) iterations with iterative linear solver

\[ Q^{m+\nu} = S(Q^m, \psi, \nu) \]

Defect after \( m \) iterations

\[ d^m = \psi - A(Q^m) \]

Defect after \( m + \nu \) iterations

\[ d^{m+\nu} = \psi - A(Q^{m+\nu}) = \psi - A(Q^m + v_m^m) = d^m - A(v_m^m) \]

with correction

\[ \nu_m^m = S(\tilde{\psi}, d^m, \nu) \]

Neglecting the sub-iterations in the smoother we write

\[ Q^{m+1} = Q^m + \nu = Q^m + S(d^m) \]

Observation: Oscillations are damped faster on coarser grid.

Coarse grid correction:

\[ Q^{n+1} = Q^n + \nu = Q^n + \mathcal{P} \mathcal{S} \mathcal{R}(d^n) \]

where \( \mathcal{R} \) is suitable restriction operator and \( \mathcal{P} \) a suitable prolongation operator

Iterative methods

Jacobi iteration

\[ Q_{jk}^{m+1} = \frac{1}{\sigma} \left[ (Q_{j,k+1}^{m+1} + Q_{j,k-1}^{m+1}) \Delta x_1^2 + (Q_{j,k}^{m+1} + Q_{j,k}^{m+1}) \Delta x_2^2 - \Delta x_1^2 \Delta x_2^2 \psi_{jk} \right] \]

Lexicographical Gauss-Seidel iteration (use updated values when they become available)

\[ Q_{jk}^{m+1} = \frac{1}{\sigma} \left[ (Q_{j,k+1}^{m+1} + Q_{j,k-1}^{m+1}) \Delta x_1^2 + (Q_{j,k}^{m+1} + Q_{j,k}^{m+1}) \Delta x_2^2 - \Delta x_1^2 \Delta x_2^2 \psi_{jk} \right] \]

Efficient parallelization / patch-wise application not possible!

Checker-board or Red-Black Gauss Seidel iteration

1. \( Q_{jk}^{m+1} = \frac{1}{\sigma} \left[ (Q_{j,k+1}^{m+1} + Q_{j,k-1}^{m+1}) \Delta x_1^2 + (Q_{j,k}^{m+1} + Q_{j,k}^{m+1}) \Delta x_2^2 - \Delta x_1^2 \Delta x_2^2 \psi_{jk} \right] \)

for \( j + k \mod 2 = 0 \)

2. \( Q_{jk}^{m+1} = \frac{1}{\sigma} \left[ (Q_{j,k+1}^{m+1} + Q_{j,k-1}^{m+1}) \Delta x_1^2 + (Q_{j,k}^{m+1} + Q_{j,k}^{m+1}) \Delta x_2^2 - \Delta x_1^2 \Delta x_2^2 \psi_{jk} \right] \)

for \( j + k \mod 2 = 1 \)

Gauss-Seidel methods require \( \sim 1/2 \) of iterations than Jacobi method, however, iteration count still proportional to number of unknowns [Hackbusch, 1994]

Two-grid correction method

Relaxation on current grid:

\[ \tilde{Q} = S(Q^n, \psi, \nu) \]

\[ Q^{n+1} = \tilde{Q} + \mathcal{P} \mathcal{S} \mathcal{R}(\psi - A(\tilde{Q})) \]

Algorithm:

with pre- and post-iteration:

\[ \tilde{Q} := S(Q^n, \psi, \nu) \]

\[ d := \psi - A(Q) \]

\[ d_c := A(r) \]

\[ \tilde{Q} := S(\tilde{Q}, \psi, \nu) \]

\[ \psi := \nu + \mathcal{P} \mathcal{S} \mathcal{R}(\psi - A(\tilde{Q})) \]

\[ d := \psi - A(Q) \]

\[ d_c := A(r) \]

\[ \tilde{Q} := S(\tilde{Q}, \psi, \nu) \]

\[ \nu := \nu + \mathcal{P} \mathcal{S} \mathcal{R}(\psi - A(\tilde{Q})) \]

[Hackbusch, 1985]
Stencils and multi-level methods

V-cycle
\(\gamma = 1\)

2-grid

3-grid

4-grid

W-cycle
\(\gamma = 2\)

Stencils at coarse-fine boundaries in 1D

1D Example: Cell \(j, \psi - \nabla \cdot \nabla q = 0\)

\[d^j = \psi_j - \frac{1}{\Delta x_j} \left( \frac{1}{\Delta x_{j+1}} (Q^j_{j+1} - Q^j_j) - \frac{1}{\Delta x_{j-1}} (Q^j_j - Q^j_{j-1}) \right) = \psi_j - \frac{1}{\Delta x_j} \left( H^j_{j+\frac{1}{2}} - H^j_{j-\frac{1}{2}} \right)\]

\(H^j\) is approximation to derivative of \(Q^j\).

Consider 2-level situation with \(n+1 = 2\):

\[Q^{l+1}_{w-1}, Q^{l+1}_{w}, Q^{l+1}_{w+1}\]

Solution needs to be continuously differentiable across interface.

Easiest approach: \(H^{l+1}_{w+\frac{1}{2}} \equiv H^l_{j+\frac{1}{2}}\)

No specific modification necessary for 1D vertex-based stencils, cf. [Bastian, 1996]

Stencils at coarse-fine boundaries in 2D

Set \(H^{l+1}_{w+\frac{1}{2}} = H^l_j\). Inserting \(Q\) gives

\[Q^{l+1}_{w+1} - Q^l_{w+1} \over \Delta x_{l+1} = Q^j_j - Q^l_{w+1} \over 2 \Delta x_{l+1}\]

from which we readily derive

\[Q^{l+1}_{w+1} = 2 \over 3 Q^j_j + 1 \over 3 Q^l_{w+1}\]

for the boundary cell on \(l+1\). We use the flux correction procedure to enforce \(H^{l+1}_{w+\frac{1}{2}} \equiv H^l_{j+\frac{1}{2}}\) and thereby \(H^l_{j-\frac{1}{2}} \equiv H^l_j\).

Correction pass [Martin, 1998]

1. \(\delta H^l_{j+\frac{1}{2}} := -H^l_{j-\frac{1}{2}}\)
2. \(\delta H^l_{j-\frac{1}{2}} := \delta H^l_{j+\frac{1}{2}} + H^{l+1}_{w+\frac{1}{2}} = -H^l_{j-\frac{1}{2}} + (Q^j_j - Q^l_{w+1}) / 2 \Delta x_{l+1}\)
3. \(\delta^j := \frac{1}{\Delta x_j} \delta H^l_{j-\frac{1}{2}}\)

yields

\[\delta^j = \psi_j - \frac{1}{\Delta x_j} \left( \frac{1}{\Delta x_{j+1}} (Q^j_{j+1} - Q^j_j) - \frac{2}{3 \Delta x_{l+1}} (Q^j_j - Q^l_{w+1}) \right)\]

Stencils at coarse-fine boundaries in 2D

1. \(Q^{l+1}_{w+1} = 1 \over 3 Q^l_{w+1} + \frac{2}{3} \left( \frac{3}{4} Q^j_{j+1} + \frac{1}{4} Q^j_j \right)\)

In general:

\[Q^{l+1}_{w+1} = \left( 1 - \frac{2}{r_{l+1} + 1} \right) Q^l_{w+1} + \frac{2}{r_{l+1} + 1} (1 - f) Q^j_{j+1} + r f Q^j_j, k\]

with

\[f = \frac{x^j_{j+1} - x^j_j}{\Delta x_{j, f}}\]
Components of an SAMR multigrid method

- Stencil operators
  - Application of defect $d^l = \psi^l - A(Q^l)$ on each grid $G_{i,m}$ of level $l$
  - Computation of correction $\nu^l = S(0, d^l, \nu)$ on each grid of level $l$

- Boundary (ghost cell) operators
  - Synchronization of $Q^l$ and $\nu^l$ on $\bar{G}_j$
  - Specification of Dirichlet boundary conditions for a finite volume discretization for $Q^l \equiv w$ and $\nu^l \equiv w$
  - Specification of $\nu^l \equiv 0$ on $\bar{G}_j$

- Coarse-fine boundary flux accumulation and application of $\delta H^{l+1}$ on defect $d^l$
- Standard prolongation and restriction on grids between adjacent levels
- Adaptation criteria
  - E.g., standard restriction to project solution on 2x coarsened grid, then use local error estimation
- Looping instead of time steps and check of convergence

Additive geometric multigrid algorithm

**AdvanceLevelMG($l$) - Correction Scheme**

Set ghost cells of $Q^l$
Calculate defect $d^l$ from $Q^l, \psi^l$
[Change: $d^l := \psi^l - A(Q^l)$]

If ($l < l_{max}$)
- Calculate updated defect $r^{l+1}$ from $\nu^{l+1}, d^{l+1}$
- Restrict $r^{l+1}$ onto $d^l$

Do $\nu_2$ smoothing steps to get correction $\nu^l$

If ($l > l_{min}$)
- Do $\gamma > 1$ times
  - AdvanceLevelMG($l - 1$)
- Set ghost cells of $\nu^{l-1}$
- Prolongate and add $\nu^{l-1}$ to $\nu^l$
  - $\nu^l := \nu^l + \nu^{l-1}$
- If ($\nu_2 > 0$)
  - Set ghost cells of $\nu^l$
  - Update defect $d^l$ according to $\nu^l$
  - Do $\nu_2$ post-smoothing steps to get $r^l$
  - Add additional correction $r^l$ to $\nu^l$

Add correction $\nu^l$ to $Q^l$

**Example**

On $\Omega = [0,10] \times [0,10]$ use hat function

$$\psi = \begin{cases} -\alpha_0 \cos \left( \frac{r}{2 \alpha_0} \right), & r < \alpha_0 \\ 0, & \text{elsewhere} \end{cases}$$

with $r = \sqrt{(x_1 - x_0)^2 + (x_2 - y_0)^2}$

to define three sources with

<table>
<thead>
<tr>
<th>n</th>
<th>$\alpha_0$</th>
<th>$\alpha_1$</th>
<th>$\alpha_2$</th>
<th>$\gamma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.3</td>
<td>0.3</td>
<td>6.5</td>
<td>8.0</td>
</tr>
<tr>
<td>2</td>
<td>0.2</td>
<td>0.3</td>
<td>2.0</td>
<td>7.0</td>
</tr>
<tr>
<td>3</td>
<td>0.1</td>
<td>0.4</td>
<td>7.0</td>
<td>3.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Result</th>
<th>128 x 128</th>
<th>1024 x 1024</th>
<th>1024 x 1024</th>
</tr>
</thead>
<tbody>
<tr>
<td>$l_{max}$</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$l_{min}$</td>
<td>-4</td>
<td>-7</td>
<td>-4</td>
</tr>
<tr>
<td>$\nu_1$</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>$\nu_2$</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>V-Cycles</td>
<td>15</td>
<td>16</td>
<td>341</td>
</tr>
<tr>
<td>Time [sec]</td>
<td>9.4</td>
<td>27.7</td>
<td>563</td>
</tr>
</tbody>
</table>

Stop at $\|d^l\|_{l_{max}} < 10^{-7}$ for $l \geq 0$, $\gamma = 1$, $\nu_2 = 2$
Some comments on parabolic problems

- Consequences of time step refinement
- Level-wise elliptic solves vs. global solve
- If time step refinement is used an elliptic flux correction is unavoidable.

References


References II

