Numerical simulation of reacting flows using the unstructured adaptive mesh refinement code HAMISH

Stewart Cant, Jian Fang, Umair Ahmed, Girish Nivarti, Charles Moulinec, David Emerson, Nilanjan Chakraborty
Outline

• Background
• Introduction to the HAMISH Code
• Code Tests
  – 1-D planar flame
  – 1-D Head-on quench
  – 2-D flame propagation
  – 2-D and 3-D channel flow
  – 2-D thermal conduction
  – 2-D and 3-D Rayleigh-Taylor instability
  – 3-D Taylor-Green vortex
  – 3-D isotropic decaying turbulence
• Scalability and Code Profiling
• Summary and Perspectives
Background

• Adaptive Mesh Refinement
  – Dynamic adaption of the mesh based on the solution
  – Local in space and time

• Advantages of AMR
  – Higher accuracy and lower cost compared with a static mesh
  – CPU time and memory savings
  – Full control of the local mesh resolution
  – More detailed physics for the same number of cells

• Main Applications
  – Problems with large dynamic range of scales
  – Flames, two-phase flow, boundary layers, shock waves
AMR in practice

Problems with interfaces
- Flames (Boxlib)
- R-T instability (ENZO Code)
- (FLASHCode)

Problems with discontinuities
- Moving Shock-Wave (PARAMESH)
- Supersonic Vehicle

Problems with great variety of scales
- Computing Cosmic Cataclysms
- Turbulence (FLASH Code)

Problems with complex geometries
- Engine Combustion
- Drone
HAMISH

Detailed chemical description

Unstructured AMR using Octree-based Morton ordering

Level set approach for two-phase flow

MPI based parallelisation with dynamic load balancing

High-order adaptive RK time integration

High-order reconstruction method

Detailed chemical description

Unstructured AMR using Octree-based Morton ordering

Level set approach for two-phase flow

MPI based parallelisation with dynamic load balancing

High-order adaptive RK time integration

High-order reconstruction method
AMR in HAMISH

- Cartesian unit-cell mesh - unstructured

Morton ordering

AMR cell heirarchy
Refinement criterion based on the Euclidean norm of the local Laplacian
Tree balancing ensures that (at most) $h-2h$ transitions exist

Partition Interval Table stores the highest local Morton code on each processor

Conservation of fluxes

- Flux calculation (linear scheme: 2\textsuperscript{nd} order)

\[ f_{\text{Interface}} = \frac{1}{2} \left( f_1 + f_{\text{INEXT}} \right) \]

\[ \frac{\partial f}{\partial x}_{\text{Interface}} = \frac{(f_1 - f_{\text{INEXT}})}{0.5(L_1 + L_{\text{INEXT}})} \]

\[ f_{\text{Interface}} = \frac{1}{2} \left( f_1 + 0.5(f_{\text{INEXT}} + f_{\text{INEXT2}}) \right) \]

\[ \frac{\partial f}{\partial x}_{\text{Interface}} = \frac{(f_1 - 0.5(f_{\text{INEXT}} + f_{\text{INEXT2}}))}{0.5(L_1 + L_{\text{INEXT}})} \]

Conservation is ensured
RENO scheme

Arbitrarily high-order reconstruction scheme

Solution is reconstructed within each cell using polynomial basis functions $\phi$

$$u(x, y, z) = \bar{u}_0 + \sum_{k=1}^{K} a_k^{(u)} \phi_k(x, y, z)$$

$$\phi_k = \psi_k - \int_{-1/2}^{1/2} \int_{-1/2}^{1/2} \int_{-1/2}^{1/2} \psi_k dxdydz$$

Fourth order sweet-spot: monomials $\psi$ are:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
<td>$x$</td>
<td>2</td>
<td>$x^2$</td>
<td>$x^3$</td>
<td>$y$</td>
<td>$yx$</td>
<td>$y^2$</td>
<td>$yx^2$</td>
<td>$y^2x$</td>
<td>$y^3$</td>
<td>$z$</td>
<td>$zx$</td>
<td>$zy$</td>
<td>$z^2$</td>
<td>$zx^2$</td>
<td>$zxy$</td>
<td>$z^2x$</td>
<td>$zy^2$</td>
<td>$z^2y$</td>
</tr>
</tbody>
</table>

Integrate over a cell:

$$\bar{u}_j = \bar{u}_0 + \sum_{k=1}^{K} a_k^{(u)} A_{jk}$$

$$A_{jk} = \frac{1}{\hat{h}_j} \int_{-\hat{h}_j/2}^{\hat{h}_j/2} \int_{-\hat{h}_j/2}^{\hat{h}_j/2} \int_{-\hat{h}_j/2}^{\hat{h}_j/2} \phi_k(x - \hat{x}_j, y - \hat{y}_j, z - \hat{z}_j) dxdydz$$

Solve the linear system:

$$A_{jk} a_k^{(u)} = b_j^{(u)}$$

using Singular Value Decomposition, producing the Moore-Penrose Pseudoinverse $A_{kj}^*$

Note that $A_{jk}$ (and $A_{kj}^*$) depend only on the local geometric configuration of the stencil.

Fluxes obtained from the polynomials evaluated at Gauss integration points on each cell face

Fluxes calculated for the same face in adjacent cells reconciled using a Riemann solver
Code Tests

- 1-D planar flame results
Code Tests

- 1-D planar flame results with AMR
Code Tests

- **1-D planar flame results with AMR**

- Fixed grid simulation with 2048 cells
- AMR simulation finished with 157 cells
Code Tests

- 1-D HOQ results with AMR

- AMR simulation started with 400 cells, finished with 160 cells
Code Tests

2D laminar flame propagation

- NSCBC gradient outflow
- Single-step chemistry
- AMR + parallel
- Base mesh 128x128
- Circular laminar flame
- Inward propagation
Code Tests

- 2-D Periodic Channel flow: results with AMR

Non-reacting viscous flow
Code Tests

laminar and turbulent channel flow

2D comparisons

3D simulations with AMR
Code Tests

- **2-D thermal conduction problem**
  - Pure thermal conduction case
  - Periodic boundary condition for all sides
  - No chemical reaction
  - Initial condition
    \[ T = 300 + 100 \exp \left( -\frac{(x - x_0)^2 + (y - y_0)^2}{4\delta^2} \right) \]
Code Tests

Rayleigh-Taylor instability

2D

3D
Code Tests

- 3-D Taylor-Green vortex

\[ u = U_0 \sin(x/L) \cos(y/L) \cos(z/L) \]
\[ v = -U_0 \cos(x/L) \sin(y/L) \cos(z/L) \]
\[ w = 0 \]
\[ p = p_0 + \frac{\rho_0 U_0^2}{16} \left[ \cos(2x/L) + \cos(2y/L) \right] \left[ \cos(2z/L) + 2 \right] \]
\[ \rho = \rho_0 \]
\[ T = \frac{p}{\rho R} \]

\[ 2\pi \times 2\pi \times 2\pi, \text{Re}=1600, \text{Ma}=0.1 \]

Code Tests

- 3-D Taylor-Green vortex

![KE temporal evolution](image1)

![Enstrophy temporal evolution](image2)
Code Tests

- 3-D Isotropic decaying grid turbulence

Q-Criterion

Vorticity magnitude

Velocity magnitude

- Fixed grid 128x128x128
- Best AMR criterion remains uncertain
- Criterion based on enstrophy currently being tested
Scalability so far

- Scalability of HAMISH without AMR (128\(^3\) cells)

<table>
<thead>
<tr>
<th>Cores</th>
<th>Runtime / s</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>1632.8</td>
<td>1.00</td>
</tr>
<tr>
<td>16</td>
<td>910.6</td>
<td>1.79</td>
</tr>
<tr>
<td>24</td>
<td>670.5</td>
<td>2.44</td>
</tr>
<tr>
<td>32</td>
<td>529.1</td>
<td>3.09</td>
</tr>
<tr>
<td>48</td>
<td>345.1</td>
<td>4.73</td>
</tr>
<tr>
<td>64</td>
<td>263.2</td>
<td>6.20</td>
</tr>
<tr>
<td>96</td>
<td>184.0</td>
<td>8.87</td>
</tr>
<tr>
<td>128</td>
<td>138.0</td>
<td>11.83</td>
</tr>
<tr>
<td>192</td>
<td>102.9</td>
<td>15.87</td>
</tr>
<tr>
<td>256</td>
<td>84.82</td>
<td>19.26</td>
</tr>
<tr>
<td>392</td>
<td>62.3</td>
<td>26.20</td>
</tr>
<tr>
<td>512</td>
<td>46.3</td>
<td>35.26</td>
</tr>
</tbody>
</table>
Pie chart shows relative CPU costs when AMR is active at every solver step. Cost of AMR is about 60% of the total - i.e. a single AMR step costs about 1.5 times as much as a single solver step.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCCOMP</td>
<td>Compares two Morton codes in their entirety</td>
</tr>
<tr>
<td>MCXYZC</td>
<td>Converts x-y-z coordinates into a Morton code at the specified level</td>
</tr>
<tr>
<td>MCCI2O</td>
<td>Converts an encoded integer array to an octal string</td>
</tr>
<tr>
<td>OCFIND</td>
<td>Searches the local octree using a given Morton code</td>
</tr>
<tr>
<td>STEPPR</td>
<td>Time stepping of the solution, including calculating RHS</td>
</tr>
<tr>
<td>ADAPTM</td>
<td>Adapts the spatial mesh</td>
</tr>
</tbody>
</table>
Summary and Perspectives

- HAMISH code is being tested and accuracy has been assessed
- Good performance and scalability are observed
- Adaptive Mesh Refinement is working and offers good local resolution
  - 1D+2D flames, HOQ, 2D+3D channels, R-T instability, TGV, 3D HIT
- Significant savings in total mesh requirement
- AMR step costs about 50% more than a solver step
  - but AMR step required only every 10 solver steps or fewer
- Current HAMISH code demonstrates its capability in capturing small-scale structures and interfaces in turbulent reacting flows.

Next:
- Further code optimisation
- Improved level set formulation for two-phase flow
- Post-processing tools for turbulence simulation
- OpenMP support
Acknowledgements

The project is supported by the UKCTRF under EPSRC grants EP/K025163/1 and EP/R029369/1, and by EPSRC Software Flagship Project EP/P022286/1