Introduction to the SBLI code

N. De Tullio and N.D. Sandham
Faculty of Engineering and the Environment
University of Southampton
Outline

• Overview of the numerical algorithms

• Review of recent code re-engineering

• Examples of current numerical investigations

• Code design and structure

• Introduction to HiPSTAR – new code building partly on SBLI structure
Numerics overview

- Compressible Navier-Stokes solver (Fortran 95+)
- 4th order central differences (5-point stencil)
- 3rd order explicit Runge-Kutta (RK3 and RK4) time advance
- Stability improved via an entropy splitting approach
- Characteristic BCs to avoid wave reflections
- Shock capturing with TVD
- Implicit 6th order filter (7-point stencil)
- Multi-block capabilities
- Inter- and intra-block parallelism using MPI (in-house library)
Previous re-engineering project[1]

- Unifications of different code versions: Multi-block, LES, Airfoil simulations (C-grid), fully 3D code version (3D curvilinear grids)
- Update to Fortran 95+ standard
- Development of a validation suite including:
  - Shock boundary-layer/interactions

Previous re-engineering project[1]

- Unifications of different code versions: Multi-block, LES, Airfoil simulations (C-grid), fully 3D code version (3D curvilinear grids)
- Update to Fortran 95+ standard
- Development of a validation suite including:
  - Shock boundary-layer/interactions
  - Mack mode instability

Previous re-engineering project[1]

- Unifications of different code versions: Multi-block, LES, Airfoil simulations (C-grid), fully 3D code version (3D curvilinear grids)
- Update to Fortran 95+ standard
- Development of a validation suite including:
  - Shock boundary-layer/interactions
  - Mack mode instability
  - 3D curvilinear capability

Previous re-engineering project\textsuperscript{[1]}

- Unifications of different code versions: Multi-block, LES, Airfoil simulations (C-grid), fully 3D code version (3D curvilinear grids)
- Update to Fortran 95+ standard
- Development of a validation suite including:
  - Shock boundary-layer/interactions
  - Mack mode instability
  - 3D curvilinear capability
- Scalability tests
  - up to 1024 processors in 2009
  - up to 200,000 by Mike Ashworth (Daresbury Lab.)

\textsuperscript{[1]} Yao et al., Re-engineering a DNS code for high-performance computation of turbulent flows. AIAA Paper 2009-566.
Current research examples

a) Boundary-layer instability over a porous surface

- 64 blocks
- Approximately 150,000 points per block
Current research examples

b) Roughness-induced transition to turbulence

- 6 blocks
- 156 million points in block 6, 2 million in block 2
b) Roughness-induced transition to turbulence

- 6 blocks
- 156 million points in block 6, 2 million in block 2
- Special treatment of density at edges
Code design

1. Number and size of blocks: 1-64 with 100,000 to 200 M points
2. Block connectivity: matching nodes
3. FD stencil: 5 points, 2 halos (central diff), 7 points, 3 halos (filter), 6-point stencil for one-sided scheme
4. Time-marching: explicit
5. Language: FORTRAN 95 - 2000
6. Parallelism: MPI
7. Inter-block communications: own “swap” routines (more later)
8. Mesh refinement: No
9. Data per grid-point: 70 numbers (work arrays and metrics)
10. Parallel I/O: MPI I/O (seems problematic for large arrays)
11. Viz. during parallel exec: No, but might be worth looking at
12. Other notes: grid and multi-block interfaces loaded by one proc and broadcast to the rest
Code structure

INIT (allocate arrays, read grids and block interf, set communicators, etc…)

do t = tin, tend
  do i = 1, RKsteps
    call SWAP
    call RHS  ! most of the effort due to work array multiplication
    call BC
  end do
  if(filter) call SWAP, call FILTER
  if(TVD) call SWAP, call TVD
  if(writeflag) call WRITE_Q
end do
end program
Swap routines for inter-block comm.

\[ l = 0 \]

\[
\text{do intf = 1, intf_num} \\
\quad \text{intf_start(intf) = l+1} \\
\quad \text{do n = intf_start, intf_end} \\
\quad \quad l = l+1 \\
\quad \quad qout(l) = q(n) \quad \text{! pack halo data for all interfaces into contiguous array} \\
\quad \text{end do} \\
\text{end do}
\]

\! Communicate

\[
\text{do intf = 1, intf_num} \\
\quad \text{intfproc = intf_proc(intf) \quad ! interface processor for this interface (set in init stages)} \\
\quad l = \text{intf_start(intf)} \\
\quad \text{call MPI_isend(qo(l), count, type, intfproc, intf_intercomm(intf), request, ierr)} \\
\quad \text{call MPI_irecv(qin(l), count, type, intfproc, intf_intercomm(intf), request, ierr)} \\
\text{end do}
\]
Typical RHS structure

! Viscosity calculation

do k = 1-zhalo:nz+zhalo
    do j = 1-yhalo:ny+yhalo
        do i = 1-xhalo:nx+xhalo
            wx(i,j,k,1-4) = q(i,j,k,1-4) ! rho, rho*u, rho*v, rho*w
            wx(i,j,k,7) = q(i,j,k,5) ! rho*E
            q(i,j,k,2-4) = q(i,j,k,2-4)/q(i,j,k,1) ! u, v and w
            wx(i,j,k,5) = ct*wx(i,j,k,7)/wx(i,j,k,1)-0.5*(q(i,j,k,2)*q(i,j,k,2)
                  +q(i,j,k,3)*q(i,j,k,3)
                  +q(i,j,k,4)*q(i,j,k,4)) ! Temperature
            wx(i,j,k,17) = reinv*(wx(i,j,k,5)*sqrt(wx(i,j,k,5))*(1+T1/T2)/(wx(i,j,k,5)+T1/T2) ! Viscosity
        end do
    end do
end do