Object-oriented Design of an AMR-algorithm for Distributed Memory Computers

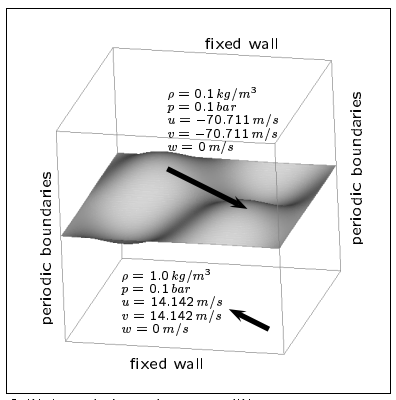
Ralf Deiterding

Chair of Numerical Mathematics & Scientific Computing
Institute of Mathematics
Technical University Cottbus
Germany
deiterding@math.tu-cottbus.de
http://www.math.tu-cottbus.de/~deiter

The blockstructured mesh refinement algorithm (AMR) by M. Berger and J. Oliger is widely used for the adaptive calculation of hyperbolic conservation laws. While serial AMR-implementations can easily be parallelized for shared memory supercomputers, the currently emerging generation of distributed memory architectures requires an appropriate design from the base level of data structures on.

We describe the object-oriented design of a framework that capsulates parallelization details inside hierarchical grid functions which are automatically distributed following the floor plan of a global grid hierarchy. Dynamic redistribution placing higher level data on the same computing node as the coarsest level data is combined with synchronization via overlapping ghost cell regions.

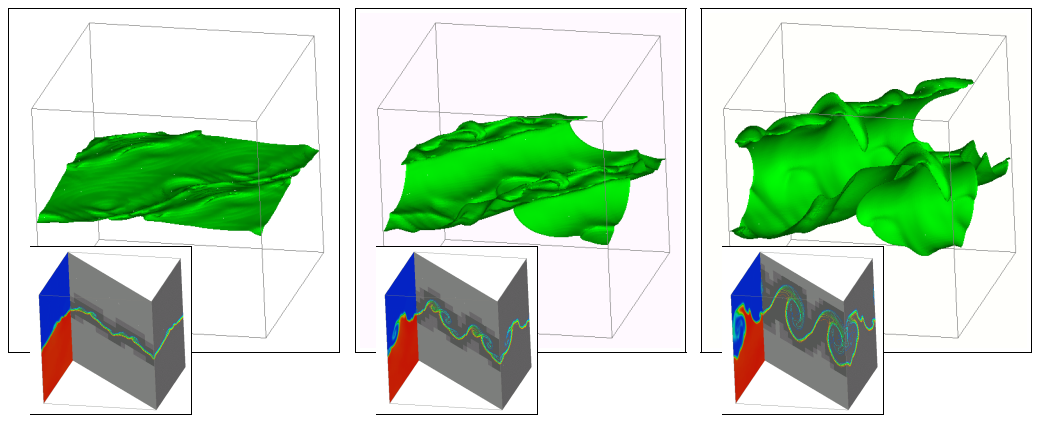
The strategy is tailored for the AMR-algorithm and benchmark calculations in two and three dimensions confirm the efficiency of the approach.
Our actual implementation employs Fortran-functions for the computational expensive single grid operations, while the framework itself highly benefits from the usage of C++. Parallel communication relies on the MPI standard.



Initial and boundary conditions on computational domain $\Omega = [0m, 2m] \times [0m, 2m] \times [0m, 2m]$.

A Kelvin-Helmholtz like instability in 3D

- Euler-equations for an ideal gas in 3D (Air with $\gamma = 1.4$)
- Wave propagation scheme with Minmod wave-limiter and transverse wave propagation
- Computation of 792 time steps with CFL-No. ≈ 0.9 to $t_{end} = 0.04s$
- Coarse grid $40 \times 40 \times 40$ cells. 2 levels with refinement factor 2. Finest level corresponds to $160 \times 160 \times 160$ grid (4 M cells).
- Number of cells increases continuously from 1.1 M up to 3 M.
- 71h real time on 7 nodes Pentium III-450 MHz (nonadative $\approx$ 190h)



Rollup of the contact line. Isosurface of density at $0.5\,kg/m^3$ at $t = 0.01s, 0.02s, 0.033s$. A diagonal cut shows the refinement grids.

Specific Application

Single grid Fortran-functions for:
- Time-explicit finite-volume scheme
- Initial condition
- Physical boundary conditions

Interface-objects to framework

AMR

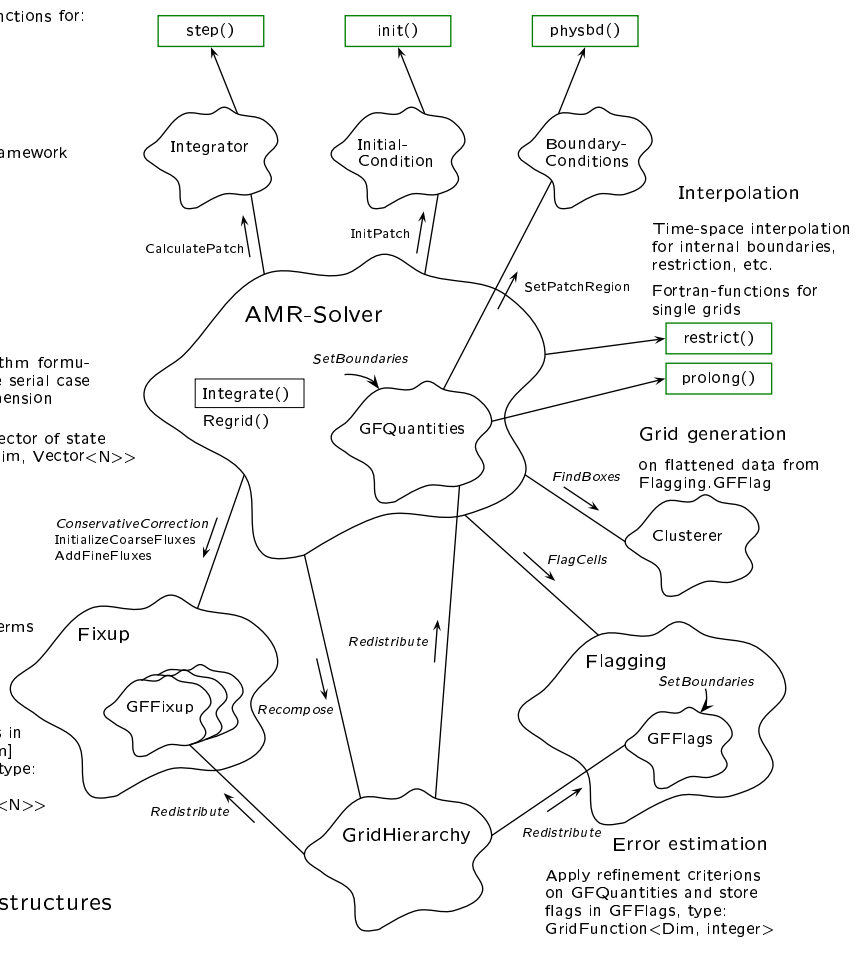
Recursive AMR-algorithm formulated nearly like in the serial case and independet of dimension

GFQuantities stores vector of state type: GridFunction<Dim, Vector<N>>

Flux correction

Calculate correction terms to replace coarse grid fluxes by find grid fluxes at hanging nodes

Store correction terms in array GFFixup[1:2 Dim] at sides of fine grids; type: GridFunction<Dim-1, Vector<N>>



Interpolation

Time-space interpolation for internal boundaries, restriction, etc.

Fortran-functions for single grids

Grid generation

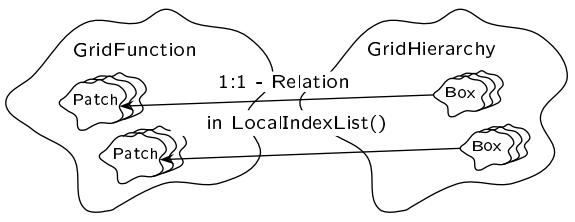
on flattened data from Flagging.GFFlag

Error estimation

Apply refinement criterions on GFQuantities and store flags in GFFlags, type: GridFunction<Dim, integer>

Hierarchical data structures

- Data is stored in distributed GridFunction<Dim,DataType>
- GridFunctions allocate Fortran-format data in Patch<Dim, DataType>-objects according to LocalBoxLists() of GridHierarchy



- Patch<Dim,DataType> adds ghost cells (colored regions)
- Data of all levels resides on same node $->$ Most operations are strictly local
- AMR initiates parallel communication (procedures in italics) by calling SetBoundaries(), Recompose(), Restribute()

Main program

lev = 0
Integrate(*lev*)

Methods of AMR-Solver

Recursive Procedure Integrate(l)
  Repeat r_l times
    GFQuantities.SetBoundaries(l, t)
    If (time to regrid ?) Then
      Regrid(l)
    CalculatePatches($l, \Delta t_l$)
    If (level $l+1$ exists ?) Then
      GFQuantities.SetBoundaries($l, t + \Delta t_l$)
      Integrate($l+1$)
      Fixup.ConservativeCorrection($l, l+1$)
      Restrict($l, l+1$)
    $t := t + \Delta t_l$

Procedure Regrid(l_{fixed})
  For $l = l_{finest}$ DownTo l_{fixed} Do
    Flagging.FlagCells(l)
    Clusterer.FindBoxes(Flagging.GFFlags(l))
      gives NewLocalBoxList($l+1$)
    EnsureNesting(LocalBoxList(l),
      NewLocalBoxList($l+1$))
  GridHierarchy.Recompose()
  For GF in list of GridFunctions Do
    GF.Redistribute()

For $l = 0$ To l_{fixed} Do
  CreatePatches(LocalBoxList(l))
  Copy data of old patches and delete them
  Synchronize ghost cells
For $l = l_{fixed}+1$ To l_{finest} Do
  CreatePatches(LocalBoxList(l))
  Initialize by time-space interpolation from level $l-1$ using prolong()
  Copy data of old patches and delete them
  SetBoundaries(l)

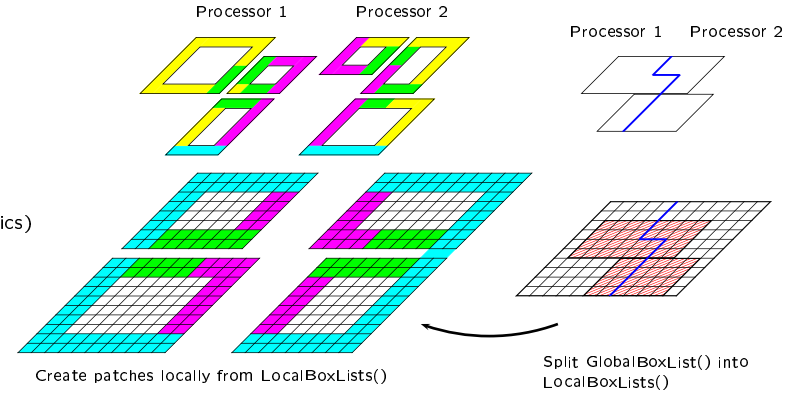
For i in LocalIndexList(l) Do
  For all boundaries of GFQuantities(l, t, i) Do
    1. If (level $l > 0$?) Then time-space interpolate from GFQuantities($l-1$), using prolong()
    2. Synchronize with neighboring patches of level l: Local and *parallel* copy
    3. Apply physbd() if necessary

For i in LocalIndexList(l) Do
  CalculatePatch(GFQuantities(l, t, i), Δt_l) gives GFQuantities($l, t + \Delta t_l, i$); store numerical fluxes $F_{l,t,i}$ temporarily
  If (level $l+1$ exists ?) Then
    Fixup.InitializeCoarseFluxes($l+1, F_{l,t,i}$)
  If (level $l > 0$?) Then
    Fixup.AddFineFluxes($l, i, F_{l,t,i}$)

For i in LocalIndexList($l+1$) Do
  project GFQuantities($l+1, t+\Delta t_l, i$) into GFQuantities($l, t+\Delta t_l$), using restrict()

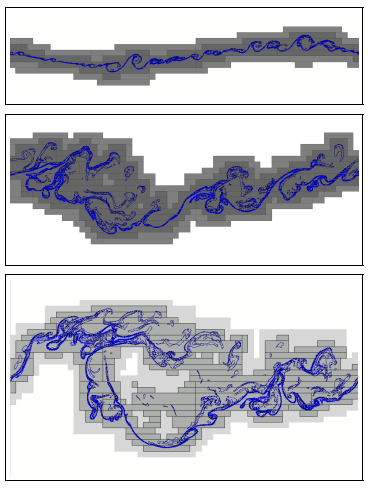
For i in LocalIndexList(l) Do
  EstimateError(GFQuantities(l, t, i), Δt_l),
  ApproximateGradients(GFQuantities(l, t, i))
  and set GFFlags(l, i) accordingly
GFFlags.SetBoundaries(l)

$l_{finest} := l_{finest} + 1$
For $l = l_{fixed}$ To l_{finest} Do
  Merge all NewLocalBoxLists(l) *into GlobalBoxList*(l)
Repartition(GlobalBoxList(0),...,
  GlobalBoxList(l_{finest})) creates LocalBoxList(l), LocalIndexList(l) for $l = 0, ..., l_{finest}$.



Create patches locally from LocalBoxLists()

Split GlobalBoxList() into LocalBoxLists()



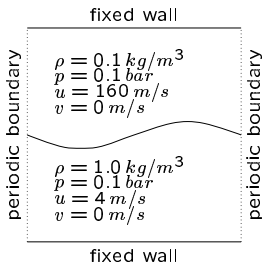
Growing pertubations of the contact discontinuity. Isolines of density at $t = 0.00375s$, $0.01375s$, $0.025s$ on refinement grids of 1st (light grey) and 2nd (dark grey) level.

A Kelvin-Helmholtz instability in 2D

- Euler-equations for an ideal gas in 2D (Air with $\gamma = 1.4$)
- Wave propagation scheme with Minmod wave-limiter and transverse wave propagation
- Computation of 1066 time steps with CFL-No. ≈ 0.95 to $t_{end} = 0.025s$
- Coarse grid 120×120 cells. 2 levels with refinement factor 4. Finest level corresponds to 1920×1920 grid (3.7 M cells).
- Number of cells increases continuously from 167 k up to 1.1 M.
- 44.8h real time on 6 nodes SP2 (nonadaptive $\approx$ 1250h)

Task	P=1 Time		P=2 Time		P=4 Time		P=8 Time		P=16 Time	
Patch integration	42462	77.5	20932	72.9	10040	63.7	5032	54.1	2492	42.4
Recomposition *Redistribution*	3207	5.8	1485 963	5.1 3.5	1061 841	6.7 5.3	745 924	8.1 9.9	590 786	10.3 13.4
Boundary setting *Synchronization*	353	0.7	452 690	1.6 2.4	510 1163	3.2 7.5	484 997	5.2 10.7	479 863	8.1 14.6
Interpolation	3138	5.7	1570	5.5	776	4.9	390	4.2	207	3.5
Conservative Fixup *Synchronization*	1580	2.9	807 25	2.8 0.1	436 107	2.8 0.7	236 92	2.5 1.0	155 59	2.6 1.0
Clustering	2161	3.9	797	2.8	334	2.1	131	1.4	78	1.3
Not measured	1871	3.5	940	3.3	487	3.1	273	2.9	166	2.8
Total / Parallel Eff.	54772	100	28661	95.5	15755	86.9	9304	73.6	5875	58.3

Breakdown of computational time on SP2 for 216 time steps $t_{end} = 0.0025s$. Times are in seconds. Parallel efficiency $= T_1 / (P \cdot T_P)$



Initial and boundary conditions on computational domain $\Omega = [0m, 2m] \times [0m, 2m]$.

Task	P=1 Time		P=2 Time		P=4 Time		P=6 Time	
Patch integration	80513	93.3	40008	88.0	21216	84.4	13518	77.5
Recomposition *Redistribution*	476	0.6	170 126	0.4 0.3	117 110	0.5 0.4	76 76	0.4 0.4
Boundary setting *Synchronization*	398	0.4	212 2583	0.5 5.7	178 2168	0.7 8.6	160 2768	0.9 15.9
Interpolation	2100	2.4	1073	2.4	600	2.4	354	2.0
Conservative Fixup *Synchronization*	1124	1.3	548 64	1.2 0.1	319 62	1.3 0.3	181 77	1.0 0.5
Clustering	450	0.5	224	0.5	110	0.4	68	0.4
Not measured	1260	1.5	430	0.9	240	1.0	164	1.0
Total / Parallel Eff.	86321	100	45438	95.0	25120	85.9	17442	82.5

Breakdown of computational time of above 3D calculation. 102 time steps to $t_{end} = 0.005s$ on PC-Cluster with Pentium III-450 MHz and Myrinet network. Times are in seconds.