

ACM/CS 114

Parallel algorithms for scientific applications

Michael A. G. Aïvázis

California Institute of Technology

Winter 2010

Unstructured grids

- ▶ solving differential equations using structured grids
 - ▶ finite differences
 - ▶ finite volumes
- ▶ finite elements take a different approach
 - ▶ subdivide the domain into elements
 - ▶ construct basis functions over each element
 - ▶
- ▶ a *simplicial mesh*
- ▶ what follows is an overview of very common but surprisingly hard problems that occur in simulations of physical systems

Finite elements

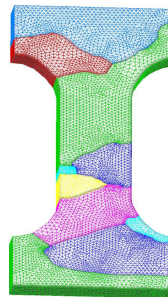


Meshing

- ▶ creating simplicial meshes is a surprisingly hard problem
- ▶ in two dimensions
- ▶ in three dimensions

Parallelization

- ▶ typically, the finest grain task is the element update that models the response of the material to the deformation induced by the motion of the vertices
 - ▶ that's where all the physics is
 - ▶ computationally intensive for non-trivial responses
 - ▶ usually involves solving a complicated variational problem, such as energy minimization
 - ▶ little or no interaction with neighboring elements, although non-local updates are becoming more common
 - ▶ e.g. thin shells, fracture based on deformation energy in a neighborhood
- ▶ coarse grain tasks are defined through *graph partitioning* that decomposes the mesh into smaller subgraphs
 - ▶ with uniform element and node distributions
 - ▶ well characterized boundaries
- ▶ performance characteristics
 - ▶ computational cost grows with the number of elements
 - ▶ communication cost grows with the number of nodes on partition boundaries

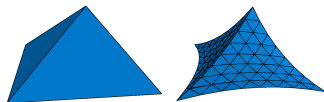
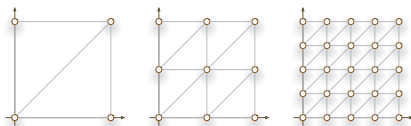


Graph partitioning



Uniform subdivision

- ▶ high fidelity simulations require high resolution
 - ▶ i.e. small elements, which implies large element counts
 - ▶ many interesting problems require tens or hundreds of millions of elements before they converge
 - ▶ creating input meshes of this size sequentially is impractical
- ▶ one possibility is *parallel uniform subdivision*
 - ▶ the input mesh is partitioned and distributed among processes
 - ▶ each process subdivides its simplices
 - ▶ edges get split in two, triangles in four, tetrahedra in eight
 - ▶ mesh quality is not affected significantly: only two new classes of simplices are introduced, regardless of the number of subdivision levels
 - ▶ the formation of new entities on partition boundaries implies the recalculation of the communications maps among processes



- ▶ can be done in linear time
 - ▶ scales well and can create enormous meshes (billions of elements)

Non-uniform subdivision and adaptive remeshing

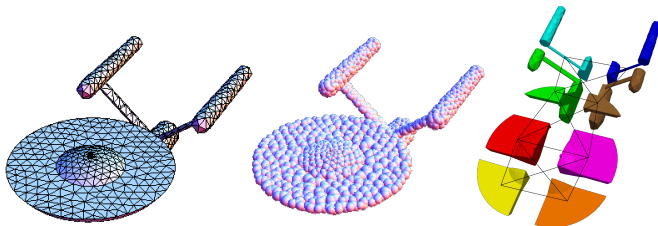
- ▶ there are many good non-uniform subdivision algorithms
 - ▶ e.g. longest edge subdivision: each tetrahedron that has an edge longer than some threshold gets subdivided by splitting that edge in two and connecting the new node to the two non-adjacent vertices
- ▶ they tend to be difficult to parallelize because they induce *one-sided* topological changes on the partition boundaries that must be communicated to the neighboring process
 - ▶ hence they are rarely used in large scale calculations
 - ▶ this is an *architectural* limitation, not an *algorithmic* one!
 - ▶ it requires restructuring the solver to support non-trivial communication with its neighbors
- ▶ similar considerations apply to adaptive remeshing
 - ▶ where a process discovers that it requires a higher element density to satisfy some correctness criterion
 - ▶ but difficult to implement when the refinement reaches the partition boundary

Contact detection

- ▶ modeling impact requires detecting when bodies come into contact and computing appropriate forces
 - ▶ both are difficult problems, especially for non-smooth, distributed meshes
 - ▶ the computations involve only elements on the free boundary
- ▶ the element updates and contact search have very different communication patterns
 - ▶ contact is *global*, since any two surface triangles can collide
 - ▶ and it is expensive, since it involves geometrical queries
 - ▶ it is best solved as an auxiliary parallel problem with its own partitioning
 - ▶ boundary partitioned using geometric criteria, such as axis-aligned cutting planes
 - ▶ requires frequent re-balancing as mesh geometry and topology change
- ▶ in *surface driven* contact detection
 - ▶ the boundary is partitioned and the centers of all the faces are inserted in some kind of spatial data structure
 - ▶ so that coarse calculations about contact candidates can be made efficiently
 - ▶ typically bucket or sparse-bucket arrays that support *orthogonal range queries* (ORQ)
 - ▶ candidates can be checked pairwise for actual contact using ray-triangle and edge-triangle intersection algorithms

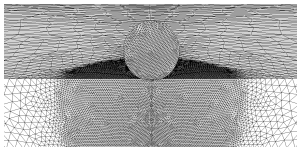
Volume based contact detection

- ▶ volume based approaches replace the elements on the boundary with equivalent spheres
- ▶ a variety of geometrical criteria can be used:
 - ▶ equivalent volume
 - ▶ diameter determined by the longest edge
- ▶ contact detection can take place very efficiently using ORQ algorithms
- ▶ simple spring models resolve the contact forces
 - ▶ based on simple potentials that take into account the sphere inter-penetration



Fracture and fragmentation

- ▶ when the material is under sufficient tension, modeling the physics correctly requires topological changes to the mesh
 - ▶ the material *fractures* when it is energetically favorable for an opening to form between two elements that are being pulled apart
- ▶ modeling this typically involves *cohesive elements*
 - ▶ prismatic elements that are inserted in the place of what used to be the common face of two tetrahedra and absorb excess energy
- ▶ when some threshold is exceeded, the cohesive element fractures and becomes free surface
- ▶ cohesive element insertion involves node, edge and face splitting
- ▶ difficulties arise as cracks run up to process boundaries and must be propagated across
- ▶ eventually, fragments form as cracks disconnect portions of the mesh

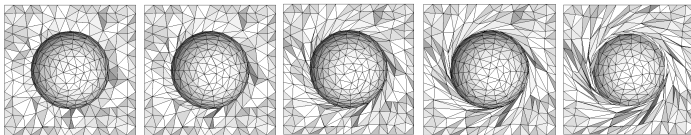


Mesh optimization

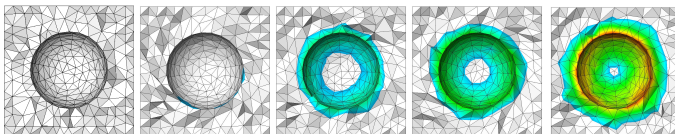
- ▶ the numerical accuracy of finite element calculations is determined in part by the quality of the mesh
 - ▶ the initial triangulation generated by the mesher
 - ▶ distortions induced by large deformations
- ▶ element quality is intuitively related to its deviation from an equilateral tetrahedron
 - ▶ there are a dozen or so pathological cases that involve either extremely small or extremely large internal angles
- ▶ optimizing mesh quality requires
 - ▶ an element quality metric that places a high penalty on the pathological cases
 - ▶ geometrical optimization: redistributing the mesh nodes so that it is possible to construct a defect-free triangulation
 - ▶ topological optimization: adjusting the way nodes are connected to form simplices
 - ▶ control over the element size, which is linked directly to both the numerical accuracy and the computational cost of the simulation
 - ▶ a sufficiently well characterized description of the boundary so that when new nodes are inserted near the boundary they do not induce distortions

Mesh optimization examples

- ▶ consider a mesh with an embedded sphere that is undergoing rigid rotation
- ▶ a quarter of a turn is sufficient to distort the mesh



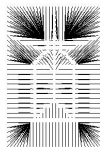
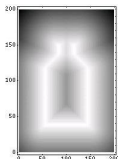
- ▶ continuous remeshing maintains good mesh quality; after three whole revolutions



- ▶ challenges:
 - ▶ keep numerical diffusion under control
 - ▶ implement in parallel

Implicit representations of surfaces

- ▶ simulating phenomena that involve the interaction of solids and fluids has its own challenges
 - ▶ fluid solvers are typically Eulerian, with structured grid that is fixed in space while the fluid flows through it
 - ▶ whereas solid solvers are typically Lagrangian, with an unstructured grid that moves along with the body it describes
- ▶ coupling requires some form of information exchange that enables
 - ▶ the solid solver to inform the fluid about the position and velocity of its evolving boundary
 - ▶ the fluid solver to exert a force on the solid through its pressure field on the boundary
- ▶ the solution involves constructing *level sets*, and computing the distance to the solid, the closest node and the closest face for each point on the grid



Summary of open problems

- ▶ many interesting physics problems require making topological changes to the simplicial meshes, and propagating them across process boundaries
 - ▶ good sequential algorithms exist, but they are difficult to parallelize
 - ▶ naïve parallelization is almost always prohibitively expensive
 - ▶ because maintaining coherence and consistency of the global information requires a lot of communication
- ▶ changing the solver architecture may be key to efficient parallel implementations
 - ▶ evolving mesh topology requires complicated interactions among processes
 - ▶ *event based* solutions enable such interactions
 - ▶ and improve simulation control by external agents
 - ▶ better monitoring through probes and sensors
 - ▶ *instrumentation* for virtual experiments