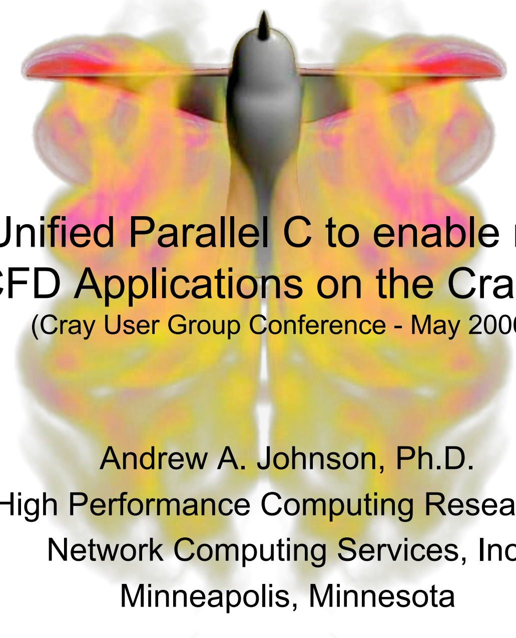


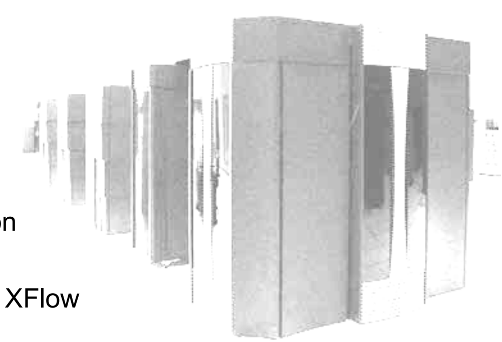
Using Unified Parallel C to enable new types of CFD Applications on the Cray X1/E

(Cray User Group Conference - May 2006)

Andrew A. Johnson, Ph.D.
Army High Performance Computing Research Center
Network Computing Services, Inc.
Minneapolis, Minnesota

Outline

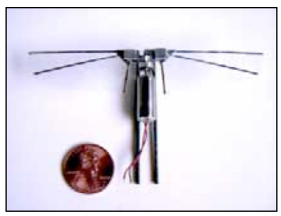
- Hummingbird Application
  - Problem Set-up
  - Results
- Technical Details
  - CFD Overview
  - Modeling and Mesh Generation
  - HPC Overview
  - New Dynamic-Mesh CFD and XFlow
  - Implementation Details
- Micro-UAVs
- Other Applications
- Future Plans



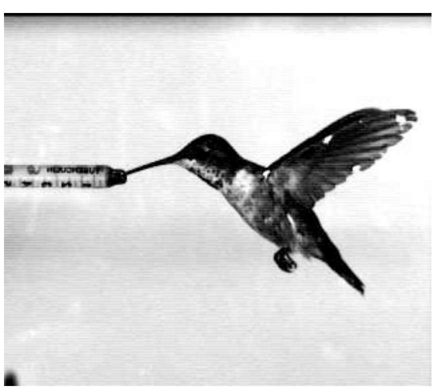
This presentation was developed in connection with contract DAAD19-03-D-0001 with the U.S. Army Research Laboratory. The views and conclusions contained in this presentation are those of the authors and should not be interpreted as presenting the official policies or positions, either expressed or implied, of the U.S. Army Research Laboratory or the U.S. Government unless so designated by other authorized documents. Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

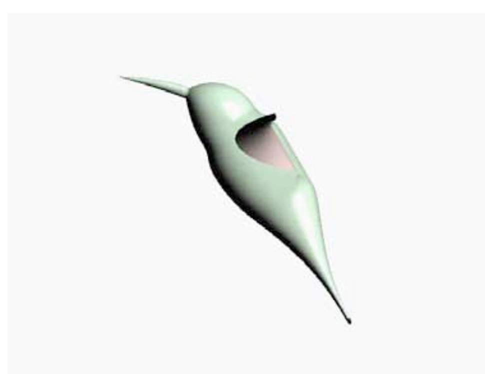
Hummingbird Application

- Early application of new numerical method and CFD code
  - Dynamic-Mesh CFD (new XFlow simulation code)
  - Challenging fluid-structure interaction applications
- Interests in small flapping-wing UAVs
- Learn from nature how animals fly, behave, perform maneuvers, and control
  - Big push in this area for many applications
  - Hummingbird is an ideal application to learn from



Hummingbird Application (geometry and wing motion)

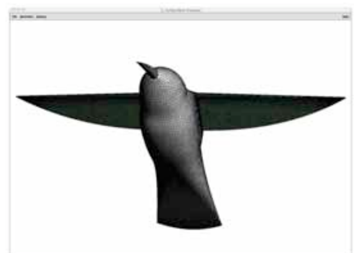




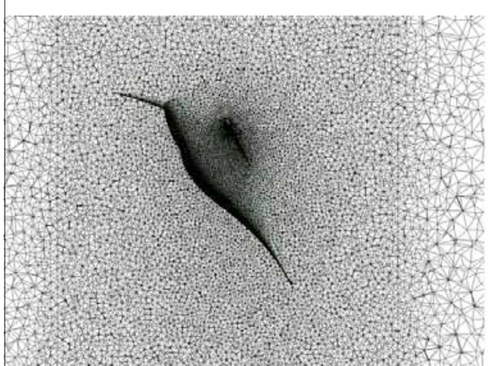
Hummingbird Application (simulation)

- CFD simulation performed on the Cray X1E
  - 20 Multi-Streaming Processors (MSPs)
  - Uses new parallel programming model Unified Parallel C (UPC)
  - Took about a day to run (3,000 time steps)
    - 400 GBytes for the simulation
    - Every other time-step saved
  - Started with 5 million tetrahedral elements, ended with around 8 million elements
  - Realistic conditions (Re ~ 2,000)
- Using new CFD code **XFlow** which implements the **Dynamic-Mesh CFD** method
  - Time-Accurate Incompressible Navier-Stokes equations
  - Stabilized Finite Element Method
  - Coupled iterative equation solver
    - GMRES-based
  - More on all of this later

AHPCRC's 256 MSP Cray X1E (NCSI Data Center, Minneapolis)



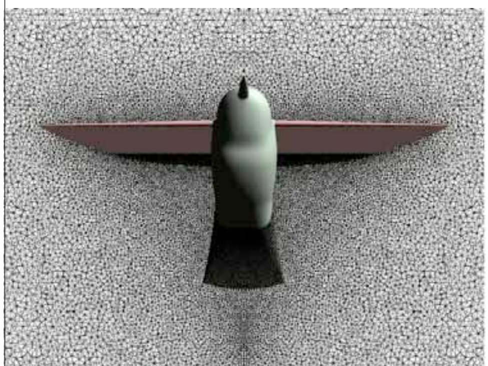
Hummingbird Application (results)



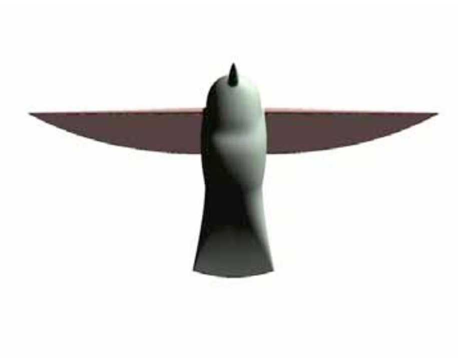
Mesh at Cross-Section

Velocity Vectors at Cross-Section

Hummingbird Application (results)

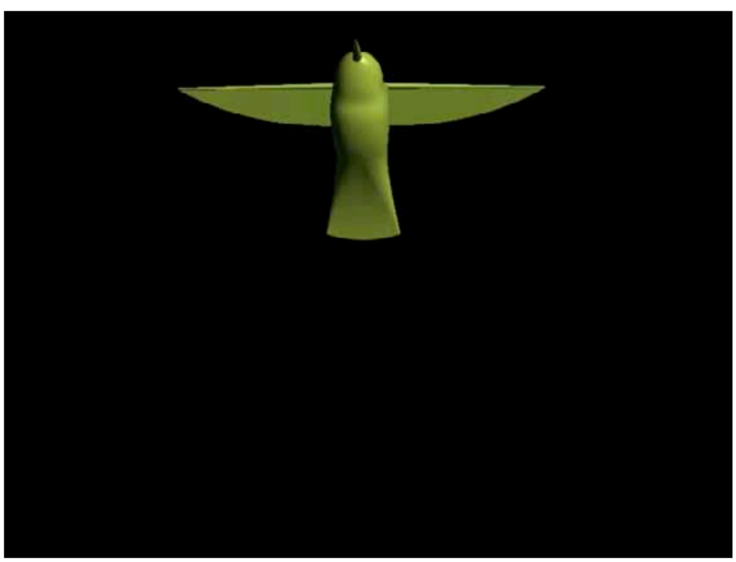


Mesh at Cross-Section



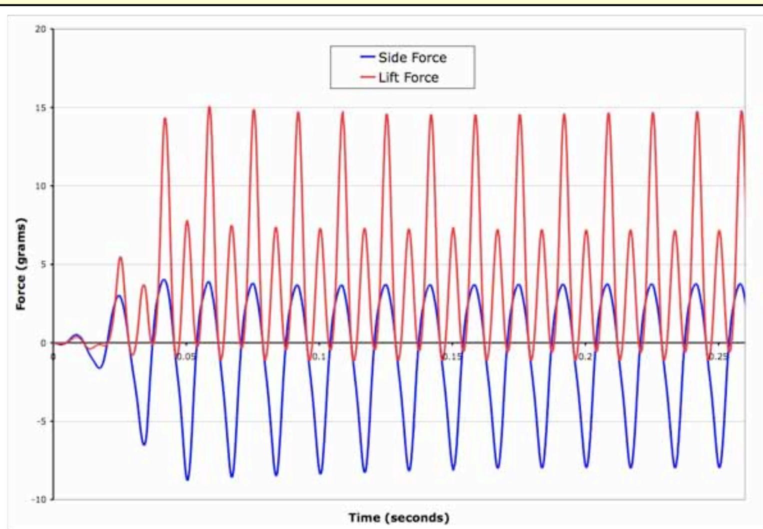
Velocity Vectors at Cross-Section

Hummingbird Application (results)



Hummingbird Application (results)

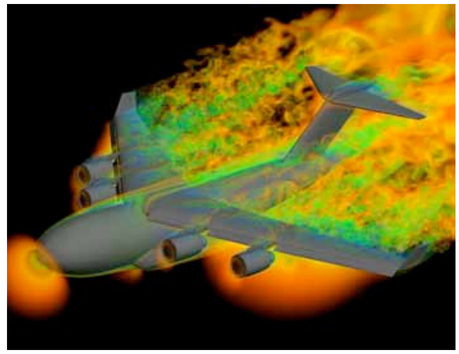
Hummingbird Application (Lift and Thrust)



About 4.5 grams lift (average) - Small forward thrust (average)

Technology Involved

- "Traditional" CFD Technology
  - Governing equations
  - Numerical formulations
  - ALE moving mesh methods
  - Solver details and types
- Automatic Mesh Generation
- Parallel Implementations
  - Programming languages
  - Mesh partitioning techniques
  - Inter-processor communication
- Dynamic-Mesh CFD Method
  - New type of method and code
  - Unique combination of existing techniques and methods



CFD Technology (Formulation)

- Time-Accurate Incompressible Navier-Stokes Equations
  - Have compressible formulations also
  - LES-based turbulence models (when needed)
    - Experimenting with others (RANS, DES)
  - Velocity and pressure are "prime" variables
- Stabilized Finite-Element Formulation
  - Unstructured meshes (usually tetrahedral elements)
  - Linear basis functions for all variables
    - Variables are stored at the mesh nodes
  - SUPG and PSPG stabilization
- ALE Methods
  - Modify fluid advection velocity with mesh velocity

CFD Technology (Formulation)

Conservation of Momentum and Mass

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \bullet \nabla \mathbf{u} + \mathbf{f}\right) + \nabla p - \mu \nabla^2 \mathbf{u} = 0$$

$$\nabla \bullet \mathbf{u} = 0$$

Variational (FEM) Formulation

$$\int_{\Omega_n} \mathbf{w}^h \bullet \rho\left(\dot{\mathbf{u}}^h + (\mathbf{u}^h - \mathbf{v}^h) \bullet \nabla \mathbf{u}^h + \mathbf{f}\right) d\Omega + \int_{\Omega_n} \varepsilon(\mathbf{w}^h) : \sigma(p^h, \mathbf{u}^h) d\Omega - \int_{(\Gamma_n)_\mathbf{h}} \mathbf{w}^h \bullet \mathbf{h}^h d\Gamma$$

$$+ \sum_{e=1}^{(n_{el})_n} \int_{\Omega_n^e} \tau\left[(\mathbf{u}^h - \mathbf{v}^h) \bullet \nabla \mathbf{w}^h - \nabla \bullet \sigma(q^h, \mathbf{w}^h)\right] \bullet \left[\rho\left(\dot{\mathbf{u}}^h + (\mathbf{u}^h - \mathbf{v}^h) \bullet \nabla \mathbf{u}^h + \mathbf{f}\right) - \nabla \bullet \sigma(p^h, \mathbf{u}^h)\right] d\Omega$$

$$+ \int_{\Omega_n} q^h \nabla \bullet \mathbf{u}^h d\Omega + \sum_{e=1}^{(n_{el})_n} \int_{\Omega_n^e} \delta \nabla \bullet \mathbf{w}^h \rho \nabla \bullet \mathbf{u}^h d\Omega = \mathbf{0}$$

CFD Technology (Formulation)

- Based on (known) boundary displacements, need to determine the motion of the interior of the mesh
- Model the domain as a linear-elastic solid and solve that equation system whenever the mesh moves
  - Every time-step for known, forced motions
  - Every non-linear iteration for unknown (coupled) motions

$$\nabla \cdot \sigma^* = 0$$

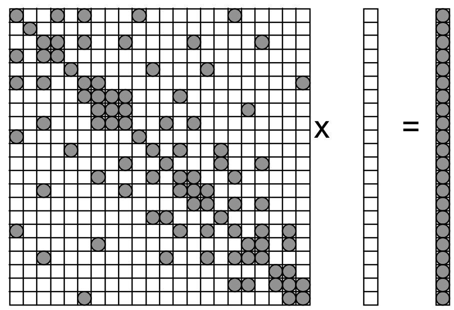
$$\sigma^* = \lambda\left(\mathrm{tr}\varepsilon^*\right)\mathbf{I} + 2\mu\varepsilon^*$$

$$\varepsilon^* = \frac{1}{2}\left(\nabla \mathbf{v} + (\nabla \mathbf{v})^\mathrm{T}\right)$$

$$\int_\Omega \varepsilon^*(\mathbf{w}^h) : \sigma^*(\mathbf{v}^h) d\Omega = \int_{\Gamma_h} \mathbf{w}^h \cdot \mathbf{h}^* d\Gamma$$

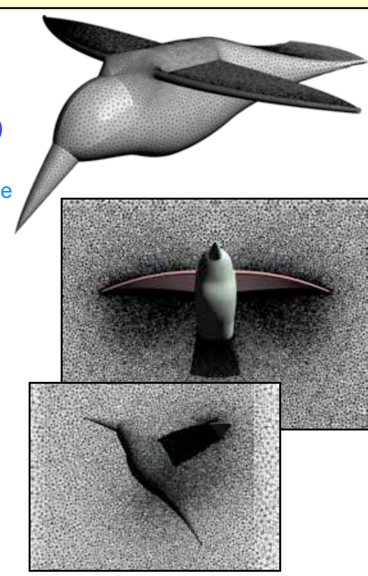
CFD Technology (Solver)

- Solve coupled (velocity-pressure) equation system at each non-linear iteration of each time-step
- Large sparse-matrix equation system
  - Use matrix-free methods in many (most) cases
- Use GMRES-based iterative equation solver
  - Diagonal Preconditioner
    - Experimenting with others
  - CG sometimes



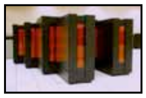
Automatic Mesh Generation

- Automatically create a large unstructured mesh for arbitrary geometry based on a CAD/surface definition
- Code developed and used since 1992 (DMG)
  - Actually developed a system of tools
    - ModelGui - CAD, DSG - Surface, DMG - Volume
- Delaunay Method
  - Tetrahedral elements
  - Face-swapping technique
  - Fast heap-based searches (among others)
- Complex coding and methods
  - Very sensitive to numerical round-off errors
- Control of refinement is critical
- Meshes anywhere between 1 and 8 million tetrahedrons generated on a workstation
  - Roughly 1 million elements per minute



Parallel Computing Systems

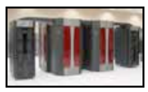
Cray 2 : Vectorization/Autotasking ~ 1985



TMC CM-5 : CMF/CMSSL ~ 1991

Cray T3D : PVM ~ 1994

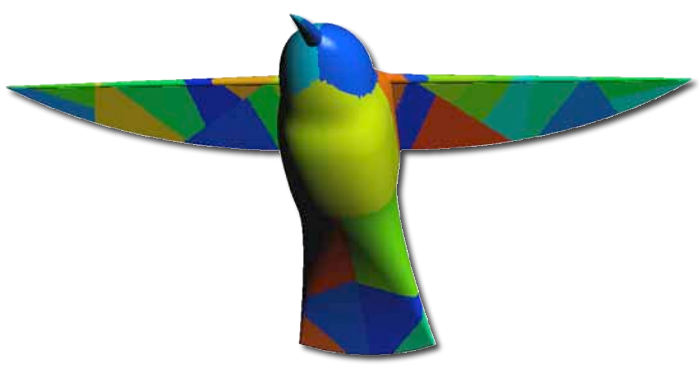
Cray T3E : MPI ~ 1996



Cray X1/E : UPC/CAF/Vector ~ 2003

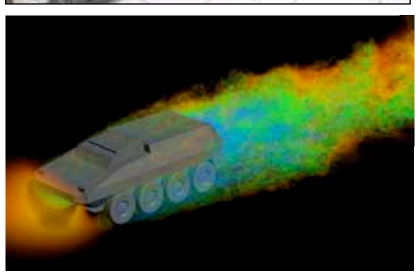
Parallel Implementation (Mesh Partitioning)

- Need to determine how to distribute (partition) the mesh and assign a piece of the mesh to each processor
- Built our own Recursive-Center-Bisection partitioner
  - Use to use “ParMetis” in the past



Parallel Implementation (dynamic re-partitioning)

AHPCRC's Cray X1E



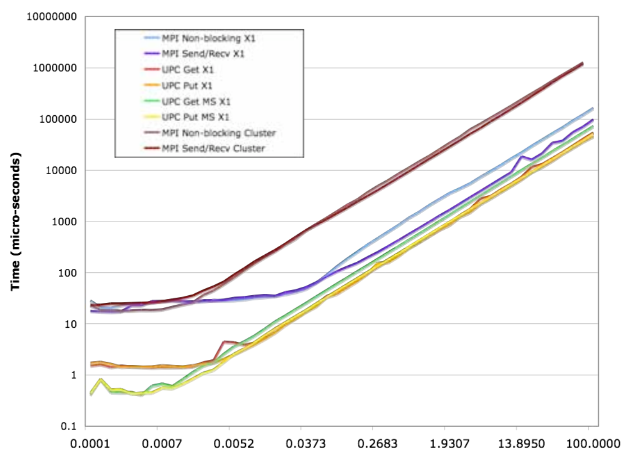
- Installed early 2003
  - Parallel computer (256 MSPs)
    - Distributed memory
  - Special Vector processor
    - 19.2 GFlops peak per MSP
  - Fast processor interconnect
  - Liquid cooled
- Highlights
  - CFD codes vectorize and perform well
    - 5+ GFlops overall per MSP
  - Enables large-scale simulations
    - 5 to 10 million elements are common
    - 50 to 100 million elements in several cases
    - 1.1 billion element application is the largest
- Global Address-space concepts at the hardware level

Parallel Implementation (Languages)

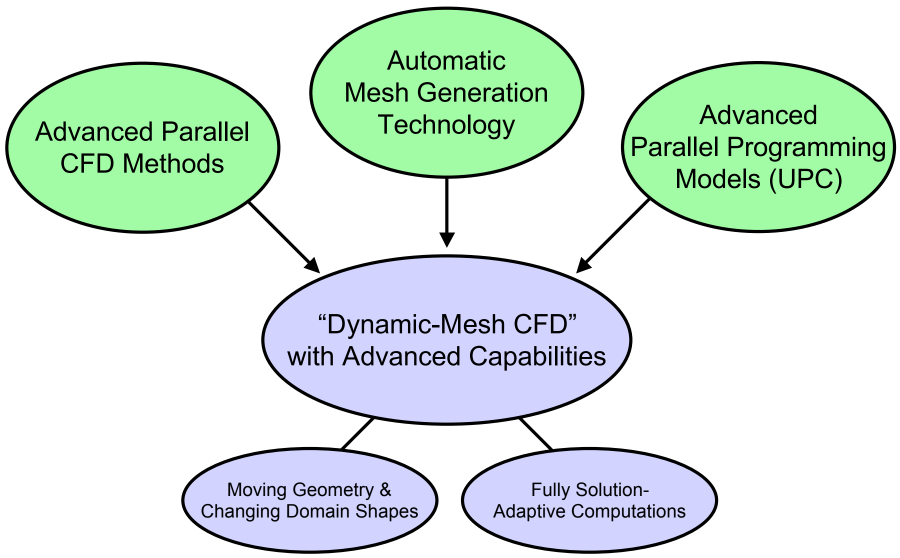
- MPI is the most common parallel "language" used
  - Our "traditional" CFD simulation codes are based on MPI
    - BenchC and Aeolus
    - 100% portable to most architectures
  - Well supported and optimized on "cluster" architectures
- New Dynamic-Mesh CFD method "requires" a newer language with advanced features
  - Parallel Global Address-Space concepts
  - Unified Parallel C
    - Developed and promoted by the "intelligence" community
      - Supporting development of UPC compilers for many platforms
    - Runs well on the Cray X1/E (supported at the hardware level)
    - Co-Array Fortran is similar
  - Easier to develop parallel codes and algorithms with UPC
  - Can implement more complex algorithms and methods

Parallel Implementation (UPC on the Cray X1E)

- PGAS concepts supported at the hardware level
- High bandwidth - Low latency network



New Directions in CFD…



Dynamic-Mesh CFD (details)

- Fully integrate automatic mesh generation within the parallel flow solver
  - Mesh generation never stops and runs in-conjunction with the flow solver
    - Element connectivity changes as required to maintain a "Delaunay" mesh
    - New nodes added as required to match user-specified refinement values
    - Existing nodes deleted when not needed
  - Mesh continuously changes due to changes in geometry and/or the solution
    - Mesh size can grow or shrink at each time step
- Very complicated method
  - Parallelism (UPC), vectorization, dynamic data structures, solvers (mesh moving and fluid flow), general CFD accuracy, scalability, CAD links, etc.
  - Will take time to fully evaluate

Dynamic-Mesh CFD (details)

At each time step…

1. Test if re-partitioning is required
2. Set-up inter processor communication
   - Required at each time step, if the mesh has changed
3. Block (color) elements into vectorizable groups
   - Due to the node-scatter operation in the Finite Element routines
4. Calculate the refinement value at each mesh node by solving the Laplace equation with known values on the boundaries
   - May be replaced by relating mesh refinement values to an error measure
5. Move the mesh by solving a modified form of the Linear Elasticity equations
6. Solve the coupled fluid-flow equation system
7. Apply the "Dynamic-Mesh Update" routines
   1. Swap element faces in order to obtain a "Delaunay" mesh
   2. Add new nodes (based on element measures) at locations where there are not enough
   3. Delete existing nodes from locations where there are too many
   4. Swap element faces to improve mesh quality

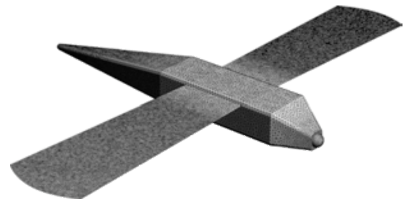
Dynamic-Mesh CFD (example)

```
/* ELEMENT BACKUP */

elemA = elementsSH[epA][eA];
elemB = elementsSH[epB][eB];

/* PRELIMINARYS */

ifc  = elementsSH[epA][eA].f [iA];
ifcP = elementsSH[epA][eA].fp[iA];
if (facesSH[ifcP][ifc].fix) return NOT_OK;

nA  = elementsSH[epA][eA].n [iA];
nB  = elementsSH[epB][eB].n [iB];
npA = elementsSH[epA][eA].np[iA];
npB = elementsSH[epB][eB].np[iB];
for (i = 0; i < NFC; i++){
   fB[i]  = elementsSH[epB][eB].f [i];
   fpB[i] = elementsSH[epB][eB].fp[i];
}
if (fqc == AR){
   q1 = elementsSH[epA][eA].qu;
   if (elementsSH[epB][eB].qu > q1)
      q1 = elementsSH[epB][eB].qu;
}

nmin = elementsSH[epA][eA].n [iminA];
pmin = elementsSH[epA][eA].np[iminA];
nmid = elementsSH[epA][eA].n [imidA];
pmid = elementsSH[epA][eA].np[imidA];
nmax = elementsSH[epA][eA].n [imaxA];
pmax = elementsSH[epA][eA].np[imaxA];
```

```
/* CONSTRUCT NEW ELEMENTS */

neplus = 0;
ne1 = eA;                    nep1 = epA;
ne2 = eB;                    nep2 = epB;
ne3 = new_elem(&neplus);     nep3 = MYTHREAD;

hn[0] = elementsSH[nep2][ne2].hn;
hn[1] = elementsSH[nep3][ne3].hn;
elementsSH[nep2][ne2] = elementsSH[epA][eA];
elementsSH[nep3][ne3] = elementsSH[epA][eA];
elementsSH[nep2][ne2].hn = hn[0];
elementsSH[nep3][ne3].hn = hn[1];

elementsSH[nep1][ne1].n [iminA] = nB;
elementsSH[nep1][ne1].np[iminA] = npB;
elementsSH[nep2][ne2].n [imidA] = nB;
elementsSH[nep2][ne2].np[imidA] = npB;
elementsSH[nep3][ne3].n [imaxA] = nB;
elementsSH[nep3][ne3].np[imaxA] = npB;

/* CHECK NEW ELEMENT STATUS */

ier1 = elem_state(ne1, nep1);
ier2 = elem_state(ne2, nep2);
ier3 = elem_state(ne3, nep3);
if (ier1 != OK || ier2 != OK || ier3 != OK)
   return NOT_OK;
```

Application - Micro-UAVs

- Always had an interest in UAVs
- Important technology for the DoD
- Lots of areas to look into
  - Optimization techniques for shape and motion
  - Optimization techniques for control algorithms
  - Fluid-structure interactions

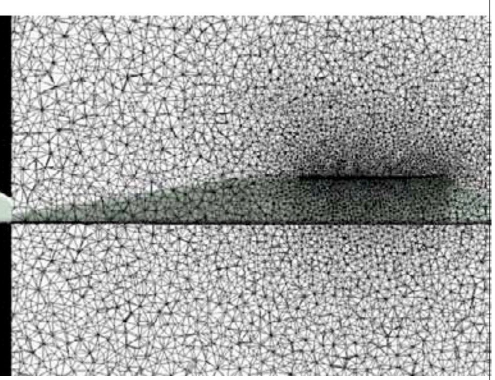


Clips from "Artificial Muscles EAP"
Yoseph Bar-Cohen, Ph.D.
Jet Propulsion Laboratory
http://ndeaa.jpl.nasa.gov/nasa-nde/lommas/eap/EAP-web.htm

Micro-UAVs (results)

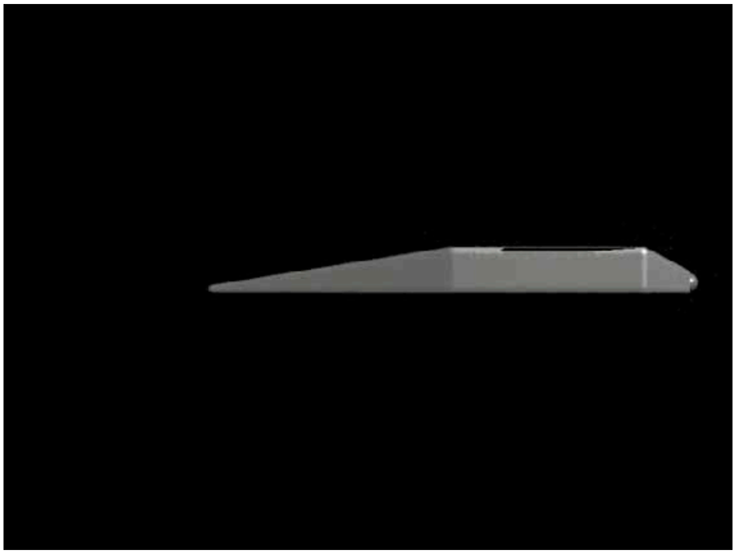
Reynolds number ~ 400

Velocity Vectors at a Cross-Section



Mesh at a Cross-Section

Micro-UAVs (results)

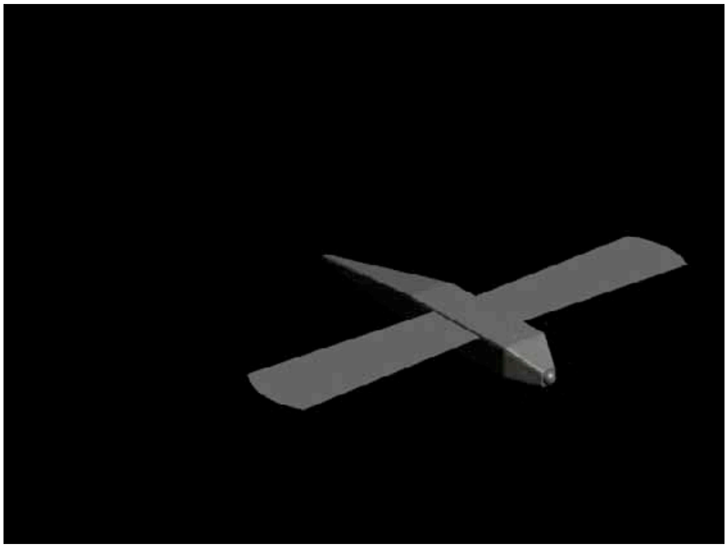


AHPCRC

29

Network Computing Services, Inc.
© 2006

Micro-UAVs (results)

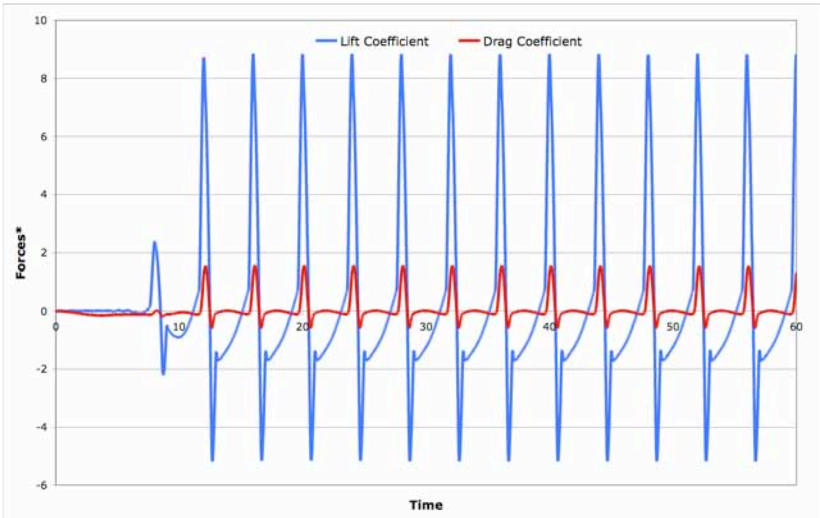


AHPCRC

30

Network Computing Services, Inc.
© 2006

Micro-UAVs (forces)



AHPCRC

31

NETWORK COMPUTING SERVICES, INC.
© 2006

Application - Airdrop Systems

- Very challenging and important application area for the Army
  - Expensive and dangerous to perform live tests
- Broad, coupled application area with lots of areas to look into
- Initial runs of the opening of a parachute

AHPCRC

32

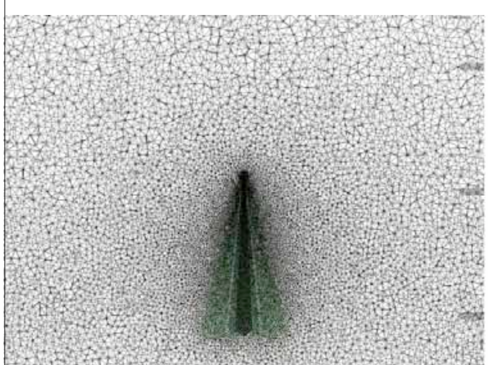
NETWORK COMPUTING SERVICES, INC.
© 2006

Airdrop Systems (results)

- Realistic flow conditions and time-scales
  - Reynolds Number ~ 4.5 million
    - LES-based turbulence model used
  - Parachute is dropped (velocity/position dynamically coupled to drag forces)
- Forced “algebraic” parachute opening motion
  - No coupled fabric model yet

AHPCRC 33 Network Computing Services, Inc. © 2006

Airdrop Systems (results)

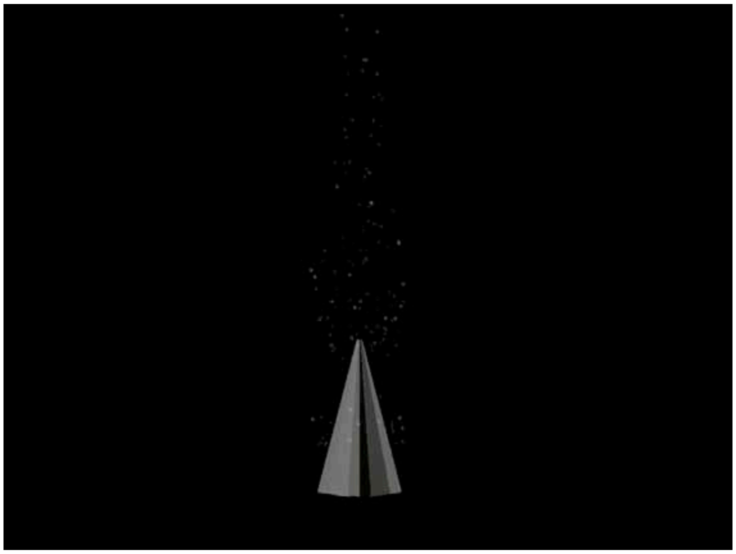


Mesh at a Cross-Section

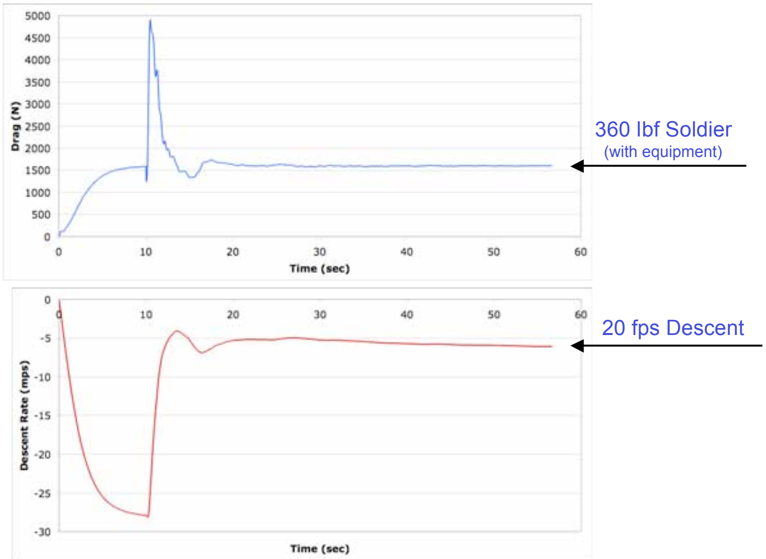
Velocity Vectors at a Cross-Section

AHPCRC 34 Network Computing Services, Inc. © 2006

Airdrop Systems (results)

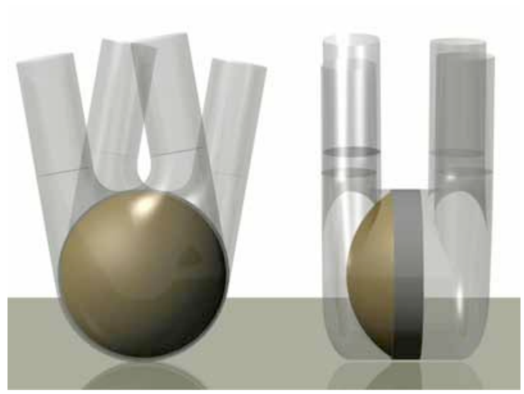


Airdrop Systems (forces and velocities)



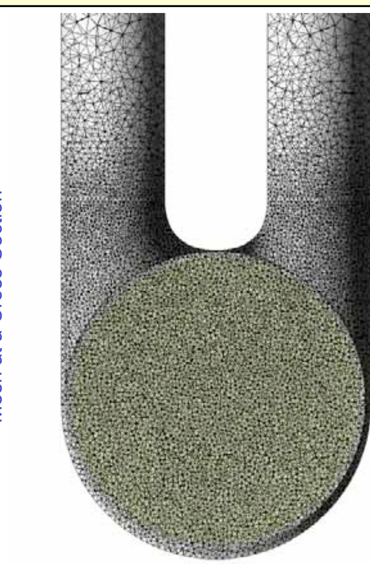
Application - Bio-Medical

- Simulation of a pumping mechanism
  - Could represent a Total Artificial Heart
- Moving (pulsating) membrane driving the flow
- Special boundary condition to model the valves
  - Restrict flow in one direction



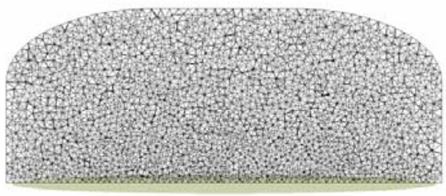
Artificial Heart (results)

Mesh at a Cross-Section



Velocity Vectors at a Cross-Section

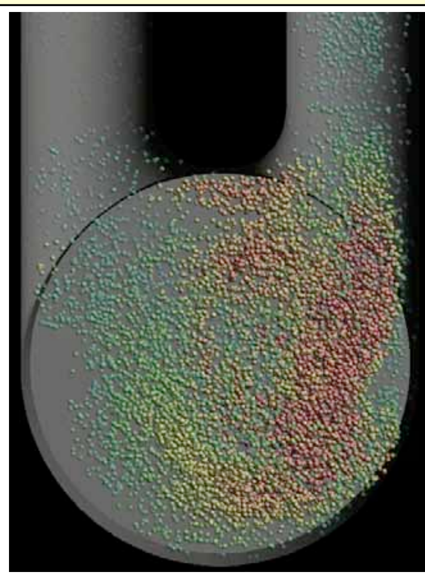
Artificial Heart (results)



Mesh at a Cross-Section

Velocity Vectors at a Cross-Section

Artificial Heart (results)



Future Plans

- Many methods and details yet to be worked out
  - Performance and scalability
  - Larger meshes, shorter run-times
- Behavior and performance on cluster systems (Berkley-UPC)
  - Interested in UPC on the Cray XT3
- Better fluid-structure coupling
  - Development (inclusion) of structures models
    - Membranes and shells
- Dynamics
  - Some exist (particles and parachute)
  - Full 6 degrees-of-freedom (3 with symmetry)
    - Basic control algorithms may be required
- Applications