Peridynamic Modeling of Structural Damage and Failure

Stewart Silling
Sandia National Laboratories
sasilli@sandia.gov

Simon Kahan
Cray Inc.
skahan@cray.com

April 21, 2004

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Outline

- Why another method?
- Theory
- Examples
- Parallelization and performance
- EMU on the Cray MTA-2

Contributors

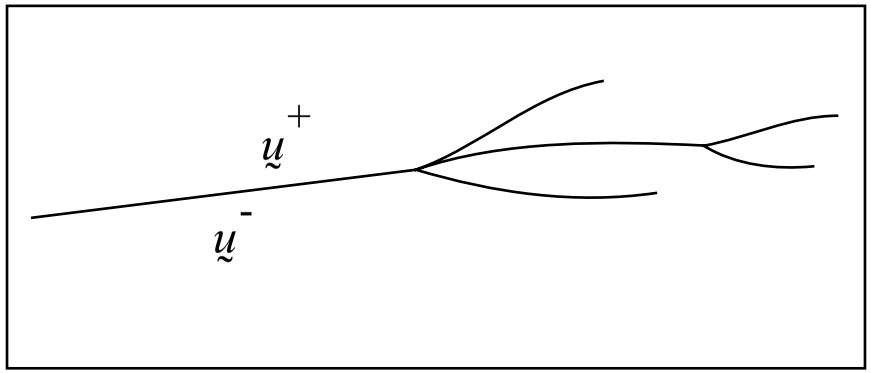
- Abe Askari, Boeing Phantom Works
- Tony Crispino, Century Dynamics, Inc.
- Paul Demmie, Sandia
- Bob Cole, Sandia
- Prof. Florin Bobaru, University of Nebraska

Sponsors

- Department of Energy
- Joint DOE/DoD Munitions Technology Program
- The Boeing Company
- US Nuclear Regulatory Commission

Need for a new theory: Why is fracture a hard problem?

- Classical continuum mechanics uses partial differential equations.
  - But the partial derivatives do not exist along cracks and other discontinuities.



  - Special techniques of fracture mechanics are cumbersome.

Goal

Develop a model in which exactly the same equations hold everywhere, regardless of any discontinuities.

- To do this, get rid of spatial derivatives.

Basic idea of the peridynamic theory

- Equation of motion:

$$\rho \ddot{\underset{\sim}{u}} = \underset{\sim}{L}_u + \underset{\sim}{b}$$

where $\underset{\sim}{L}_u$ is a functional.

- A useful special case:

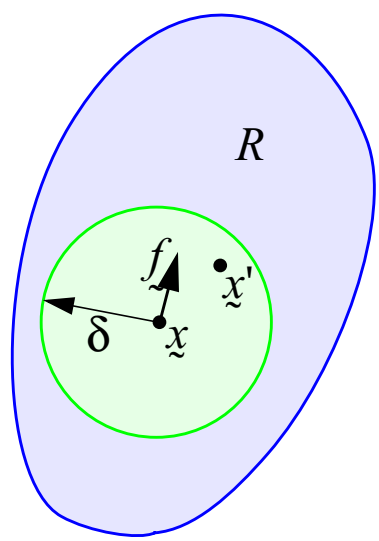
$$\underset{\sim}{L}_u(\underset{\sim}{x}, t) = \int_R \underset{\sim}{f}(\underset{\sim}{u}(\underset{\sim}{x}', t) - \underset{\sim}{u}(x, t), \underset{\sim}{x}' - \underset{\sim}{x}) dV_{\underset{\sim}{x}'}$$

where $\underset{\sim}{x}$ is any point in the reference configuration, and $\underset{\sim}{f}$ is a vector-valued function.

More concisely:

$$\underset{\sim}{L} = \int_R \underset{\sim}{f}(\underset{\sim}{u}' - \underset{\sim}{u}, \underset{\sim}{x}' - \underset{\sim}{x}) dV'.$$

- $\underset{\sim}{f}$ is the <u>pairwise force function</u>. It contains all constitutive information.
- It is convenient to assume that $\underset{\sim}{f}$ vanishes outside some <u>horizon</u> δ.

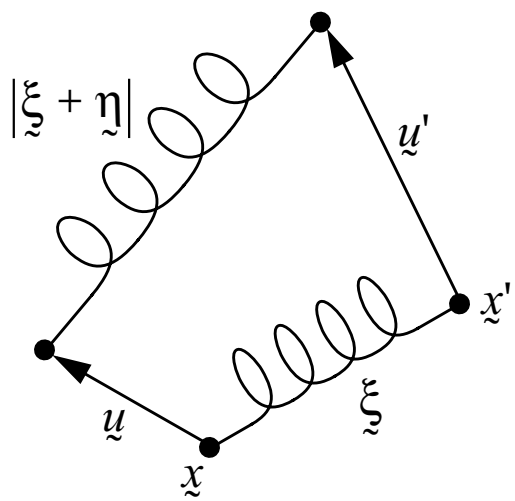


Microelastic materials

- Simplest class of constitutive models: <u>microelastic:</u>
  - There exists a scalar-valued function w, called the <u>micropotential</u>, such that

$$\underset{\sim}{f}(\underset{\sim}{\eta}, \underset{\sim}{\xi}) = \frac{\partial w}{\partial \underset{\sim}{\eta}}(\underset{\sim}{\eta}, \underset{\sim}{\xi})$$

  - Interaction between particles is equivalent to an elastic spring.
  - The spring properties can depend on the reference separation vector.

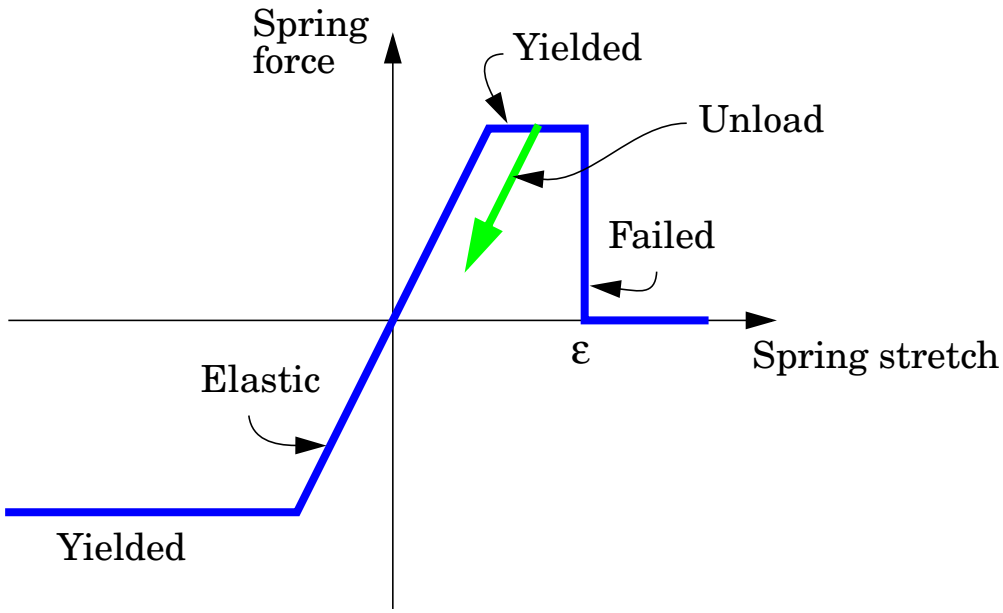


  - Work done by external forces is stored in a recoverable form

Some useful material models

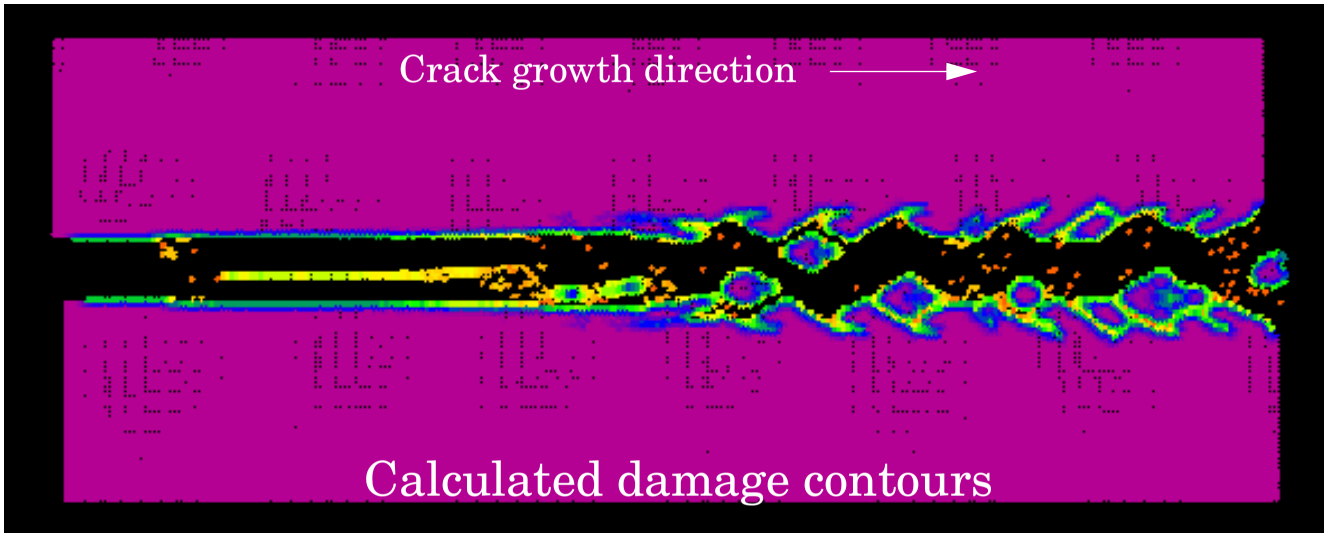
- Microelastic
  - Each pair of particles is connected by a spring.
- Linear microelastic
  - The springs are all linear.
- Microviscoelastic
  - Springs + dashpots
- Microelastic-plastic
  - Springs have a yield point
- Ideal brittle microelastic: springs break at some critical stretch ε.

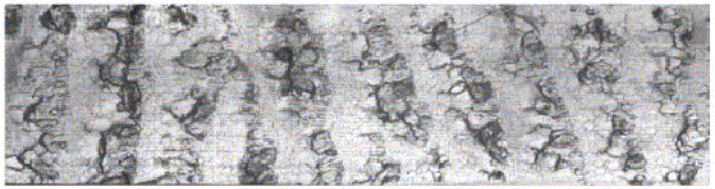
need to store bond data!



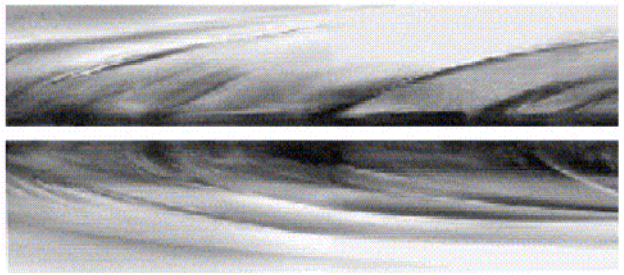
Dynamic fracture in a PMMA plate

- Plate is stretched vertically.
- Code predicts stable-unstable transition.



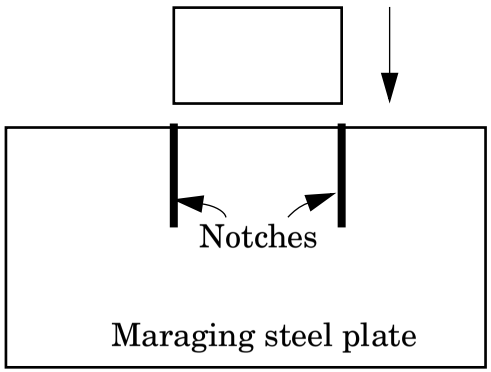


Experiment*

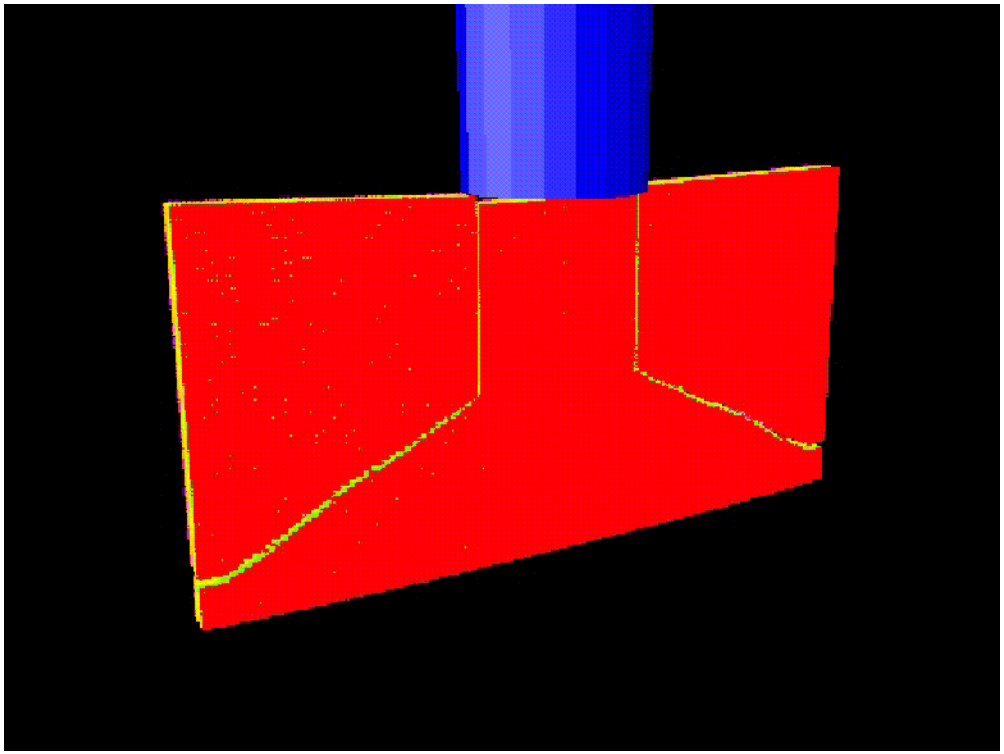


*J. Fineberg & M. Marder, *Physics Reports* **313** (1999) 1-108

Dynamic fracture in a tough steel: mode transition



- Code predicts correct crack angles*.
- Crack velocity ~ 900 m/s.



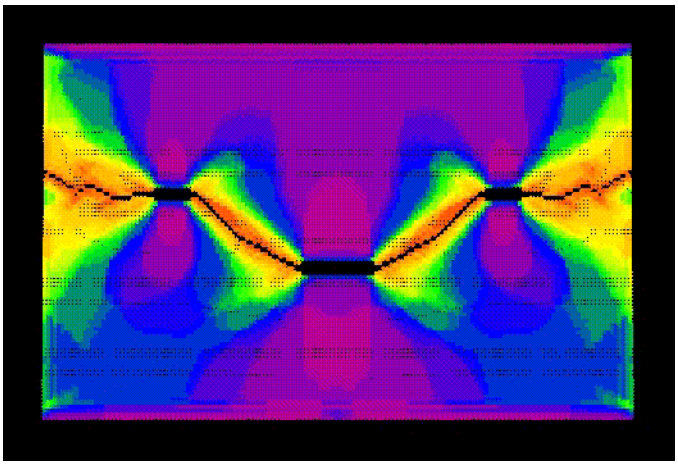
*J. F. Kalthoff & S. Winkler, in *Impact Loading and Dynamic Behavior of Materials*, C. Y. Chiem, ed. (1988)

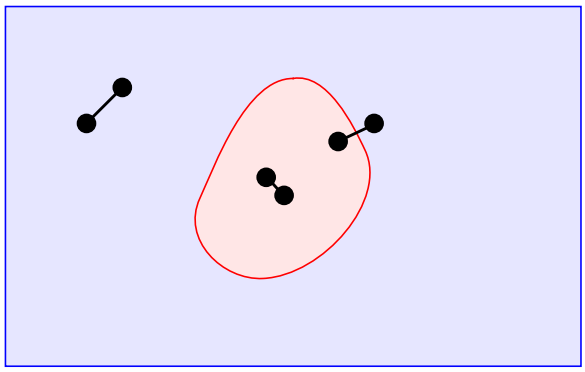
Peridynamic fracture model is "autonomous"

Sandia National Laboratories

Computational Physics Department

- Cracks grow when and where it is energetically favorable for them to do so.
- Path, growth rate, arrest, branching, mutual interaction are predicted by the constitutive model and equation of motion (alone).
  - No need for any externally supplied relation controlling these things.
- Any number of cracks can occur and interact.

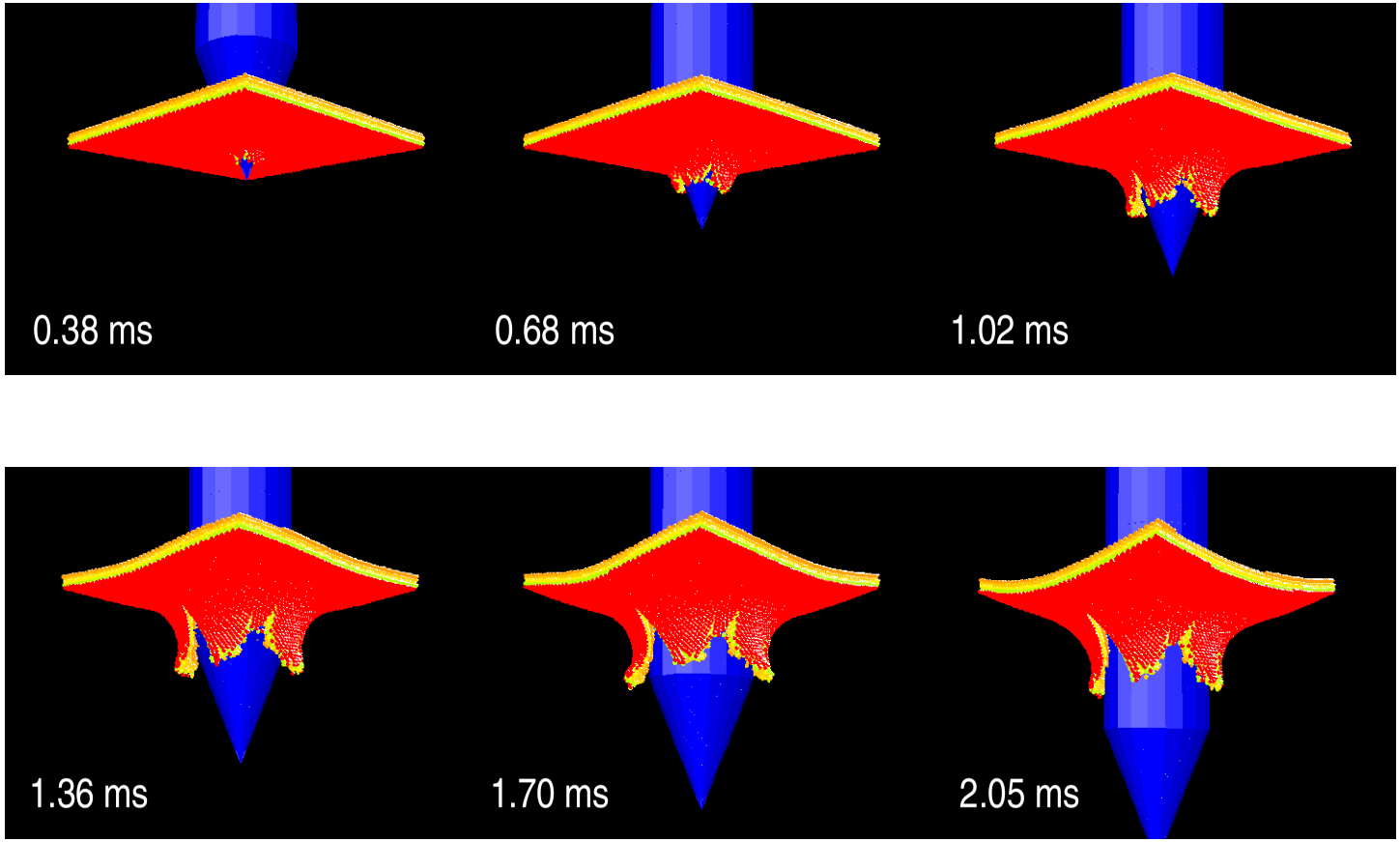




- Interfaces between materials have their own bond properties.

Example:
Perforation of thin ductile targets

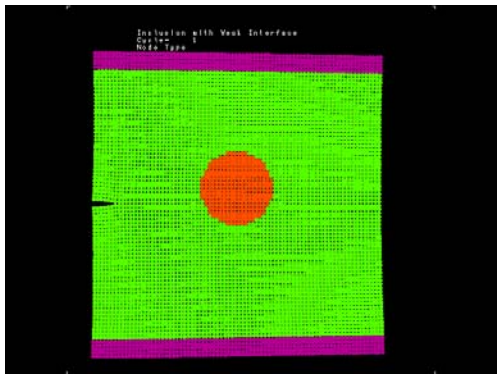
- Peak force occurs at about 0.4ms (end of drilling phase):[1]



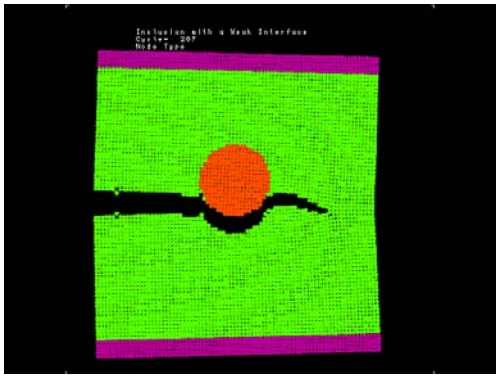
1. Not all of the target is shown.

Example:
Composite material fracture

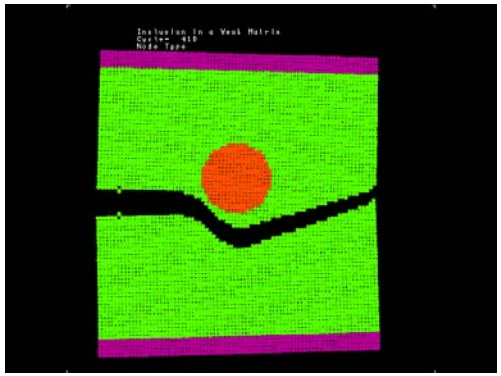
- Crack path, growth, and stability depend only on material properties.
- No need for separate laws governing crack growth.



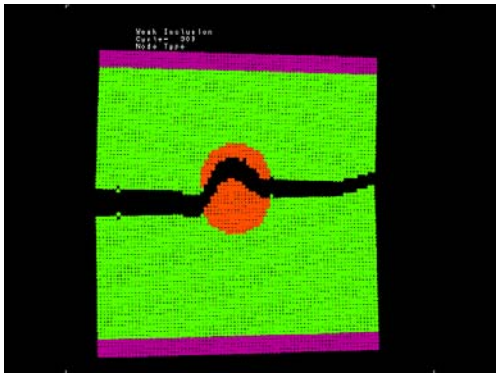
Initial condition



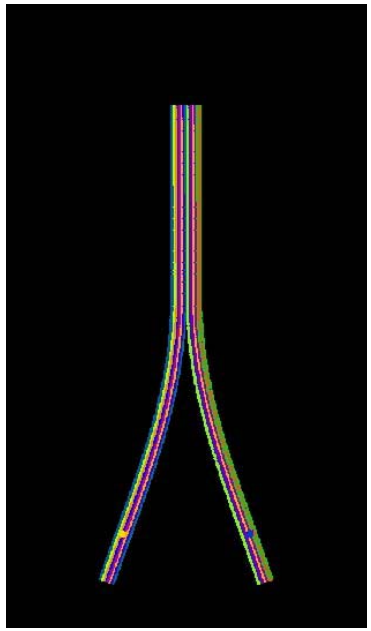
Weak interface



Weak matrix



Weak fiber

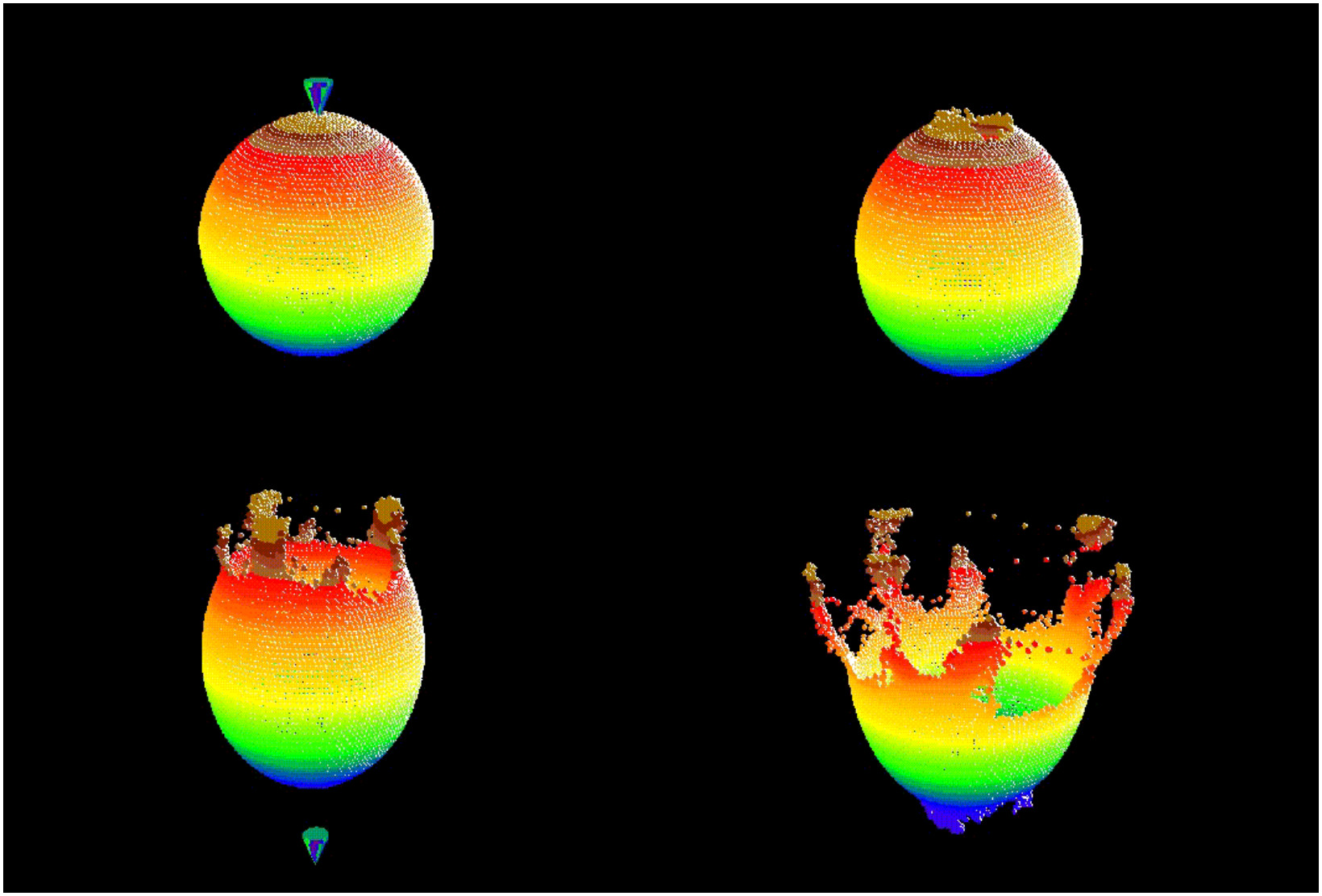


DCB test on
a composite

Example:
Dynamic fracture in a balloon

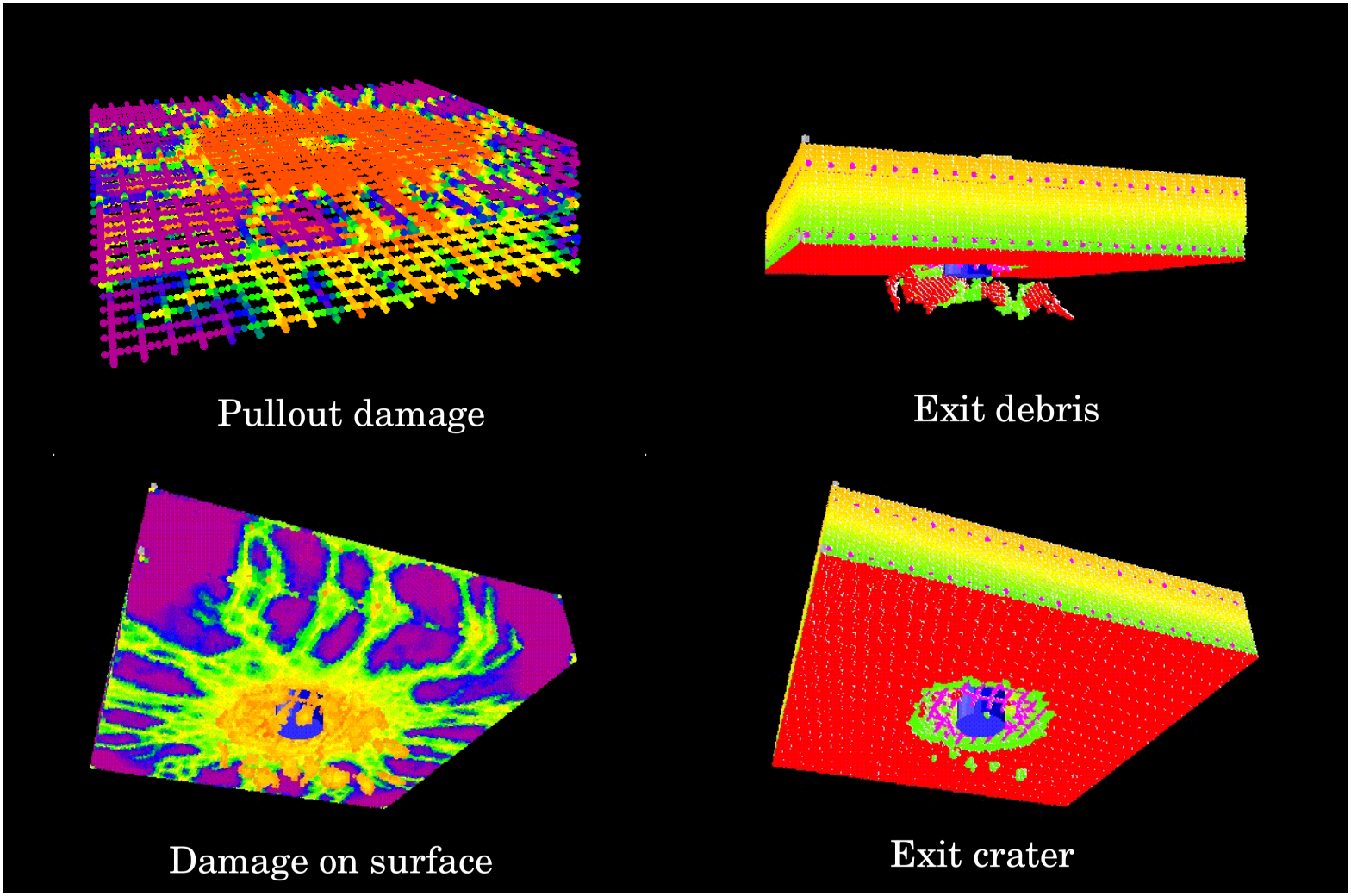
Sandia National Laboratories

Computational Physics Department



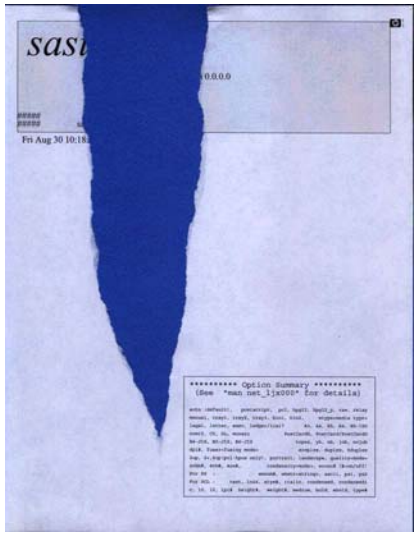
CRAY

Example:
Penetration into reinforced concrete

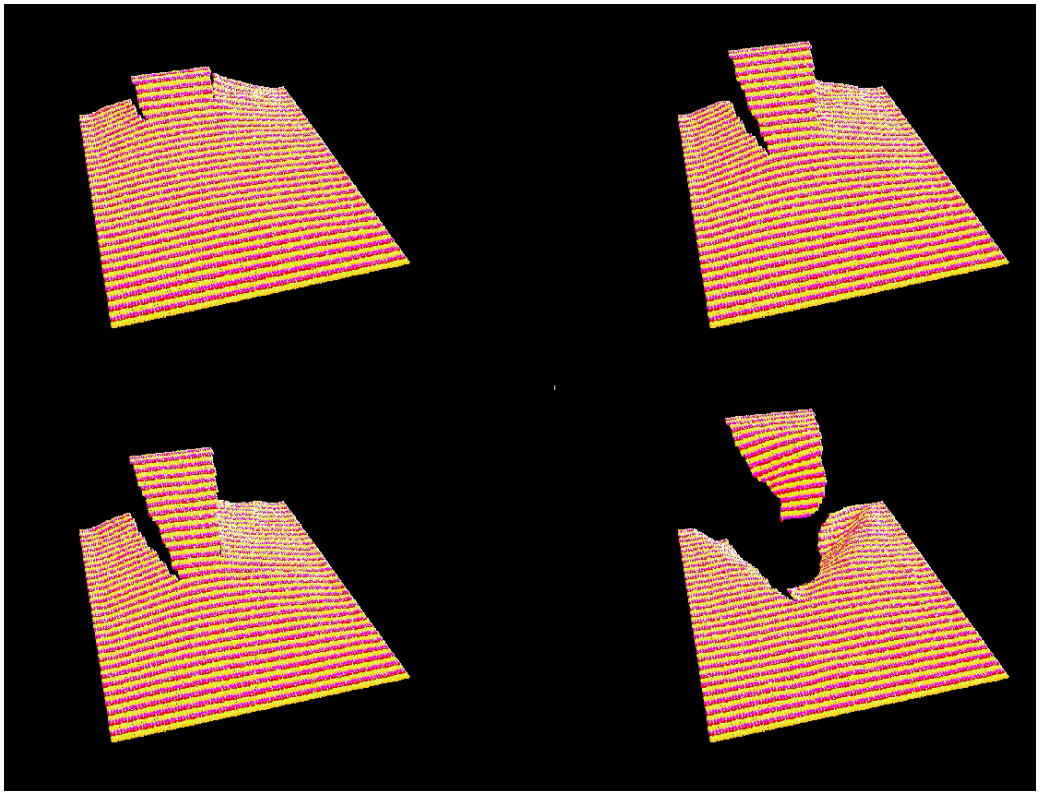


Example: Membrane fracture

- Elastic sheet is held fixed on 3 sides. Part of the 4th side is pulled upward.



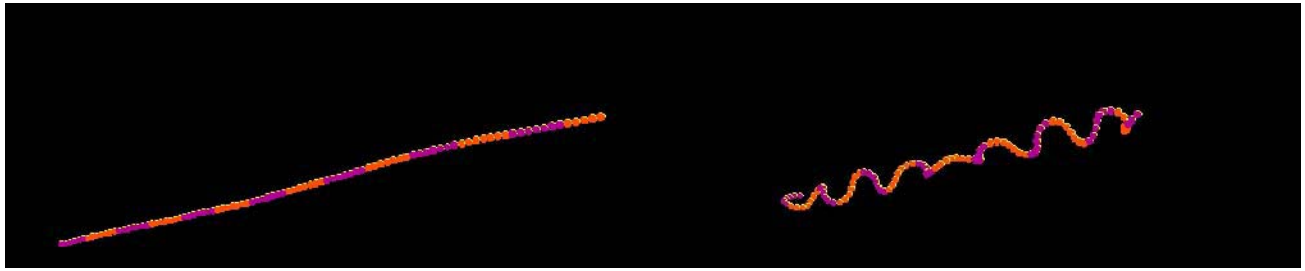
"Experiment"



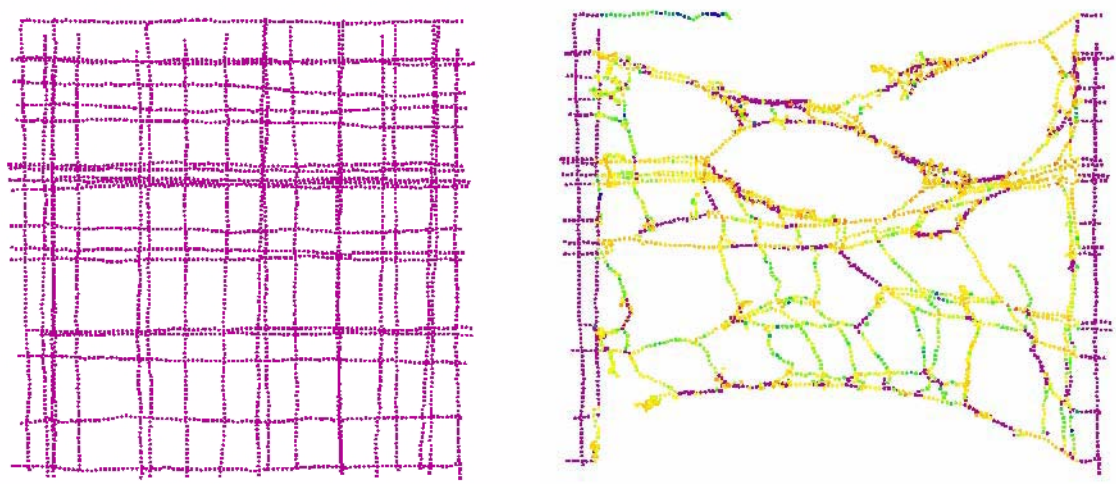
- Cracks interact with each other and eventually join up.

Examples: Mechanics of fibers

- Fibers can interact through long-range (e.g. van der Waals) or contact. f



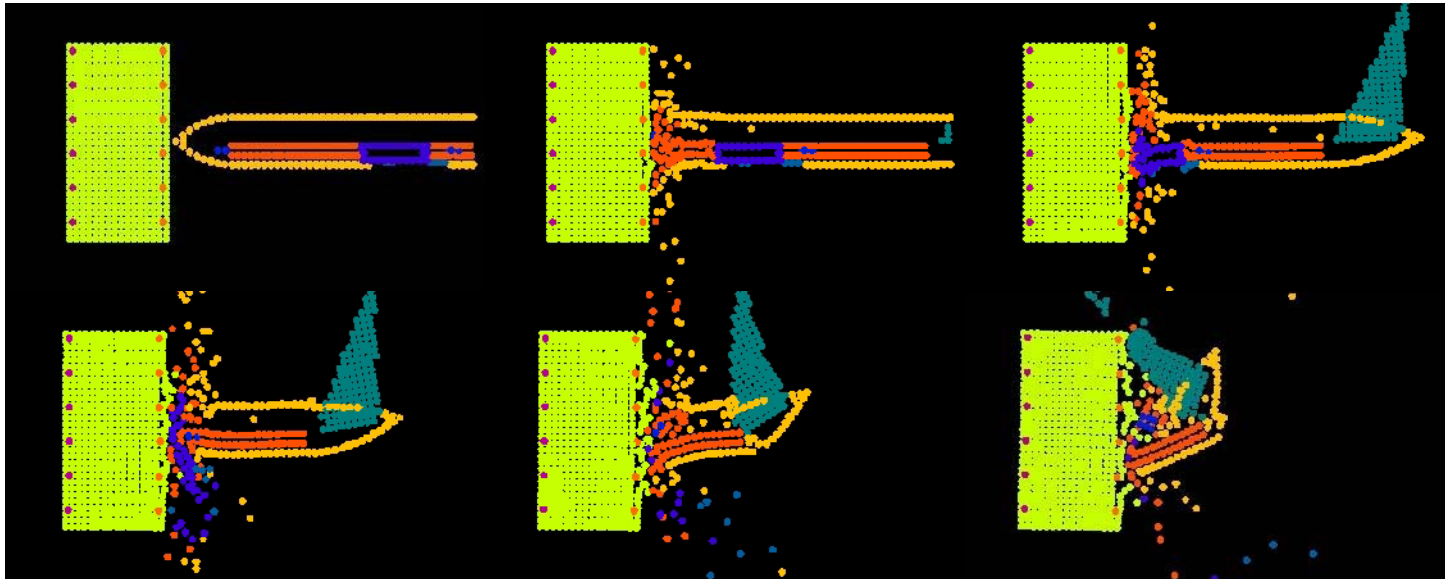
Self-shaping of a fiber due to interactions between different parts



Stretching of a network of fibers
(courtesy of Prof. F. Bobaru, University of Nebraska)

Aircraft impact onto structures

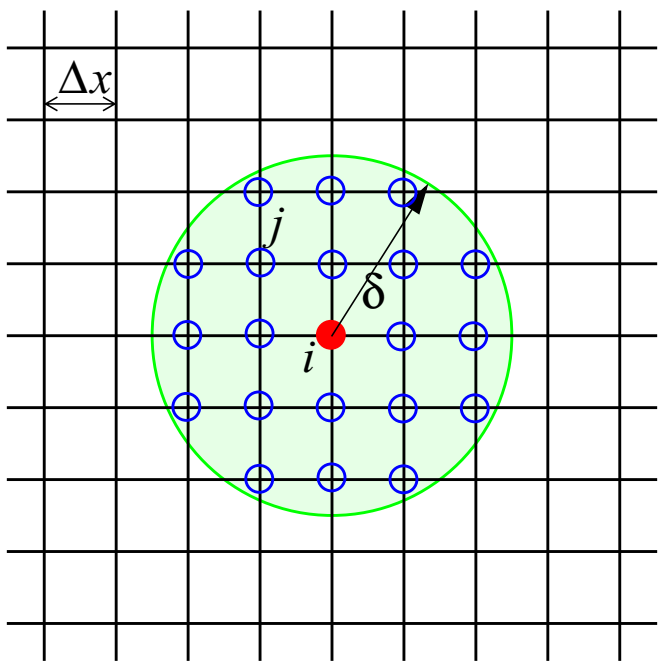
- F4 into a 3.6m thick concrete block*



*simulation of full-scale experimental data in open literature (Sugano et al., *Nuclear Engineering and Design* **140** 373-385 (1993).

Numerical solution method for dynamic problems

- Theory lends itself to mesh-free numerical methods.
  - No elements.
  - Changing connectivity.
- Brute-force integration in space.

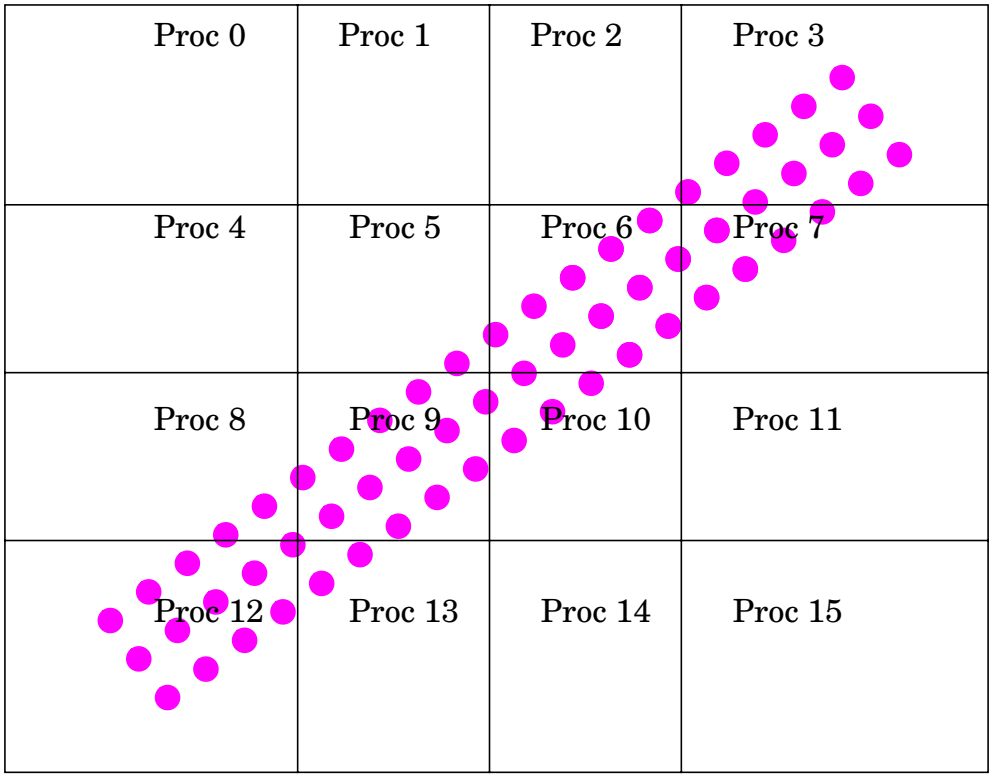


$$\rho \ddot{\underset{\sim}{u}}^i = \sum_{\left|\underset{\sim}{x}^j - \underset{\sim}{x}^i\right| < \delta} \underset{\sim}{f}(\underset{\sim}{u}^j - \underset{\sim}{u}^i, \underset{\sim}{x}^j - \underset{\sim}{x}^i)(\Delta x)^3$$

- Typical (macroscale) model: $\delta \approx 3\Delta x$
  - If long-range forces are important, could need much larger δ.

Parallelization

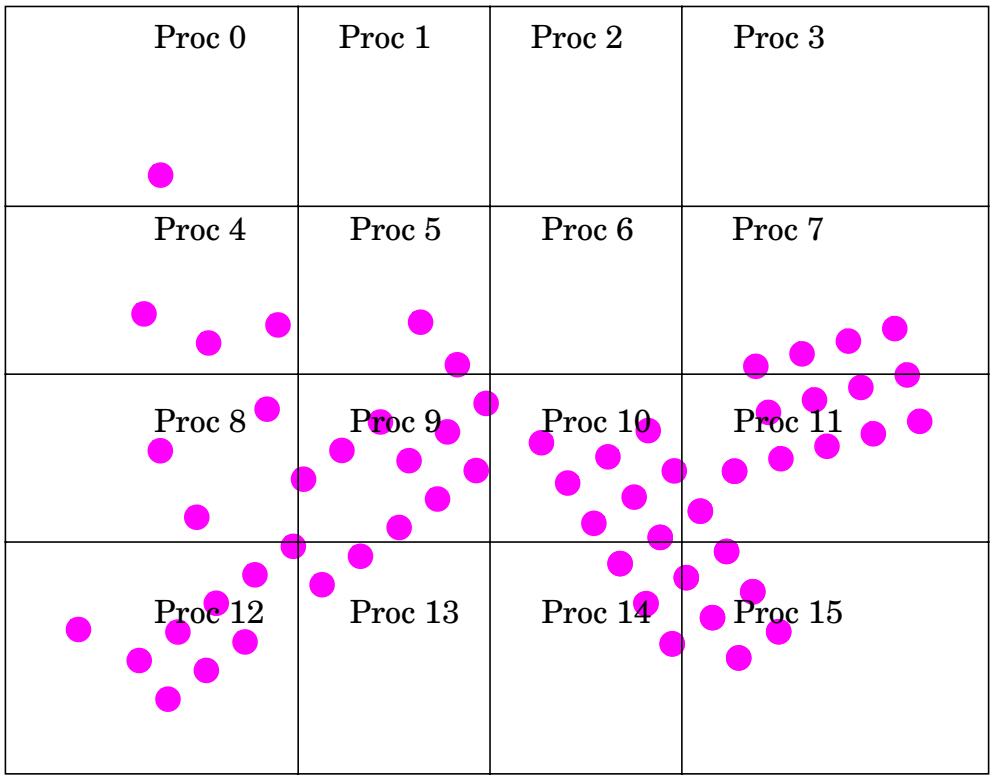
- Each processor is assigned a fixed rectangular region of space.
  - Regions are assigned so that each slice (in each direction (x,y,z) contains an equal number of nodes.



  - Easy to implement.
  - OK if the grid is more or less rectangular.
  - Static.
  - Some processors may do nothing!

Parallelization, ctd.

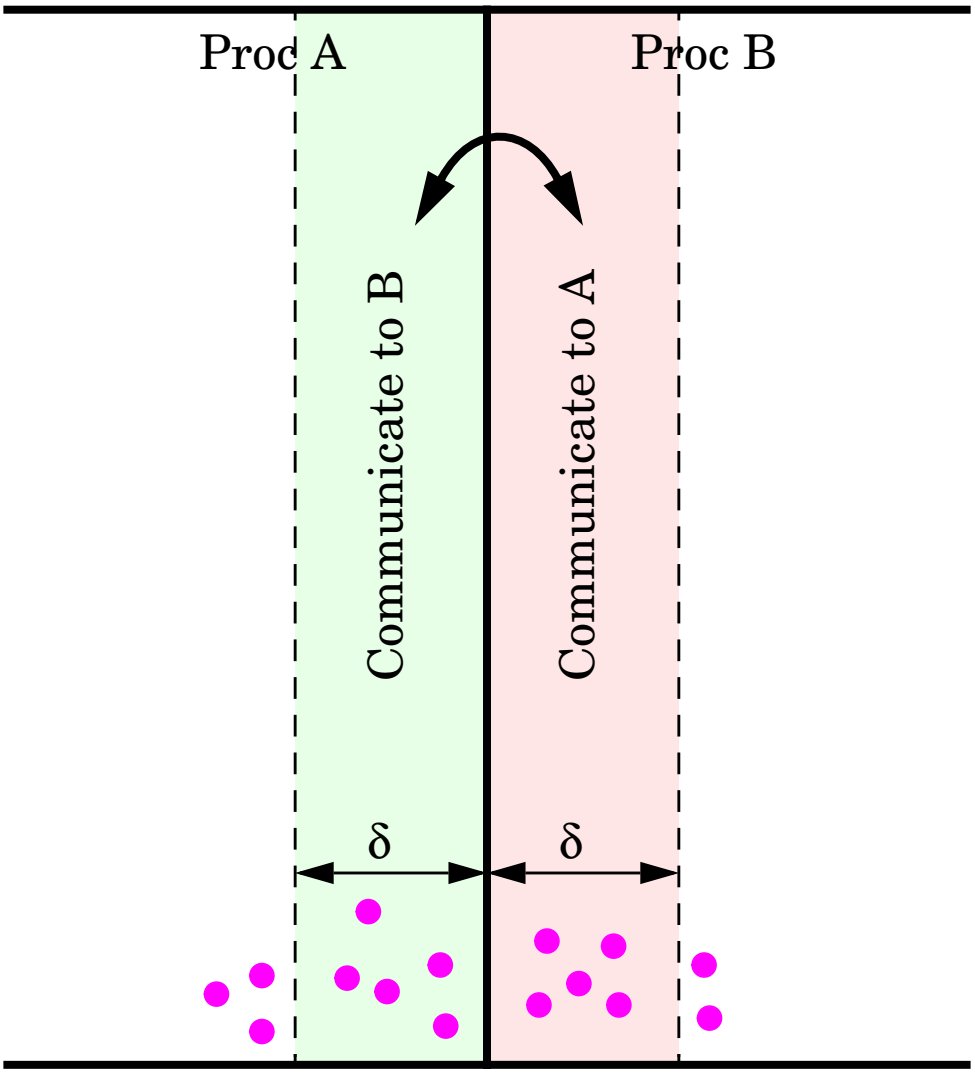
- Material nodes can migrate between processors.



- Could improve load balancing by changing the region owned by each processor as the calculation progresses.

Parallelization, ctd.
Communication requirements

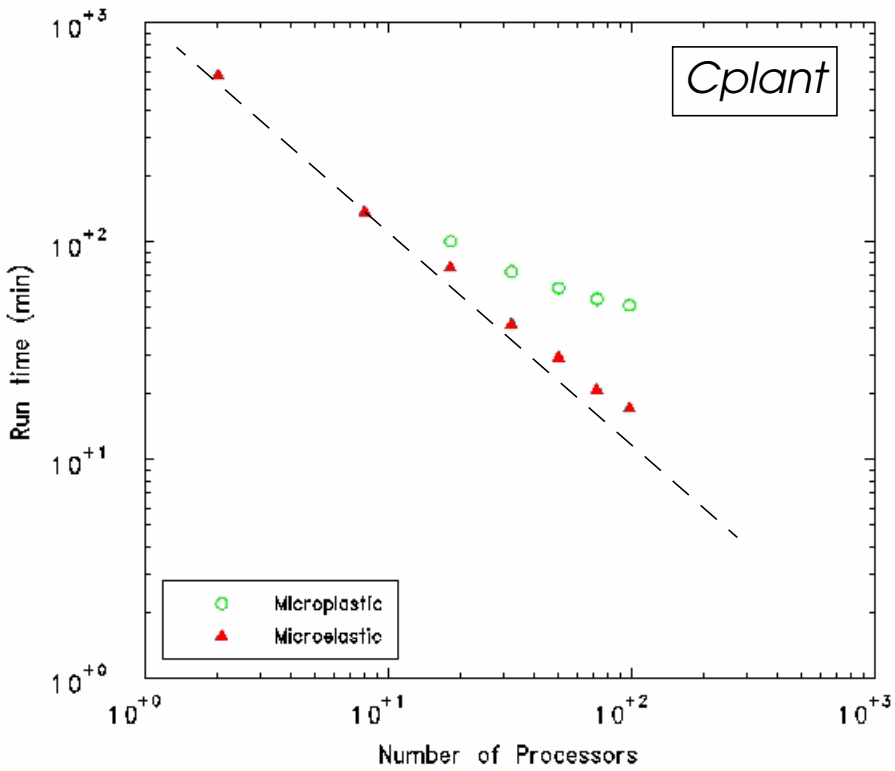
- Exchange of data must take place for nodes within δ of any other processor's region.



- The cost of this depends strongly on δ!

Parallelization, ctd. Timings

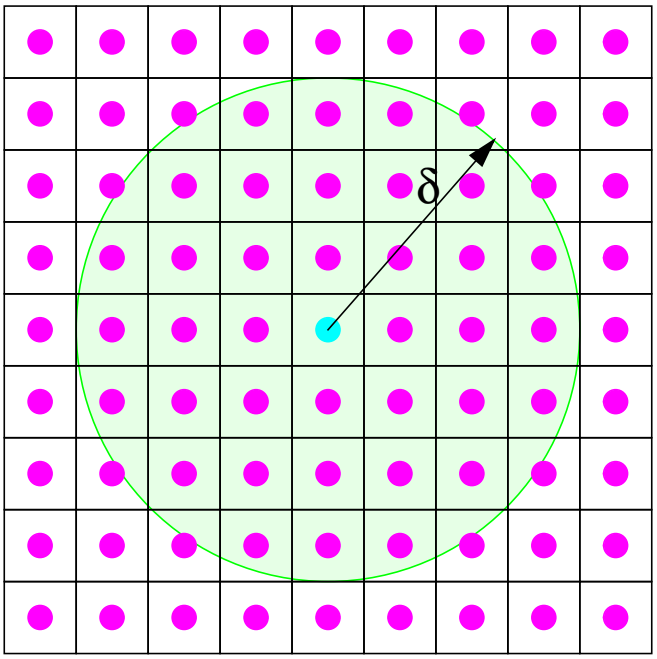
- Performance depends on material model used.



- Microelastic model requires only node data.
- Microplastic model requires bond data.
  - For each interacting pair of nodes.
  - Each node interacts with ~200 neighbors.
  - Results in much heavier communication requirements.

Parallelization, ctd.
Issue with current approach

- In the limit of one node per processor, each processor must communicate with a large number of others:



- Results in different limiting speedup properties than for a typical hydrocode.

Parallelization, ctd.
Issue with current approach

- In nanoscale modeling, $\frac{\delta}{\Delta x}$ may be large because of long-range forces.
  - Greatly increases communication requirements.

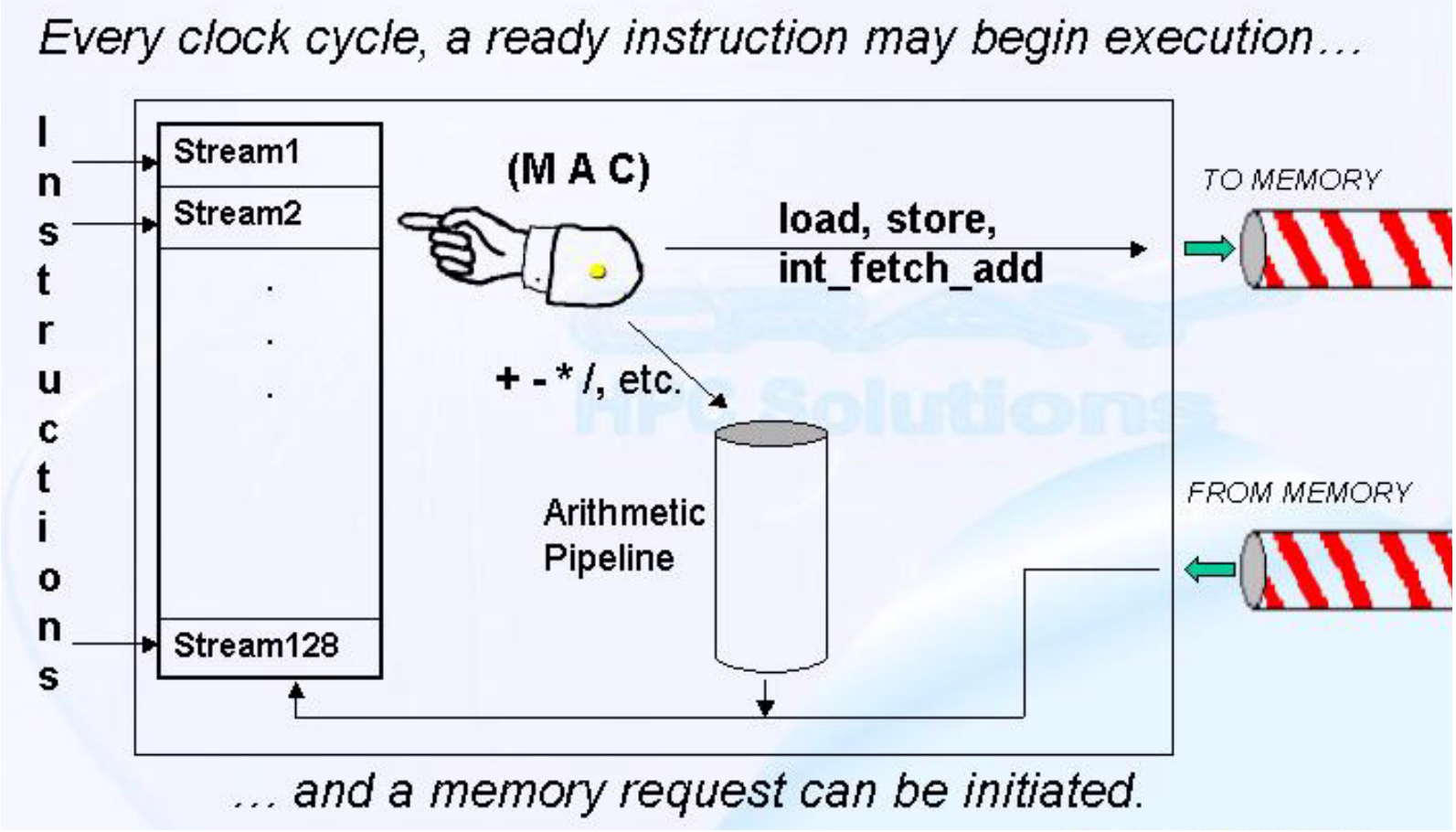
Parallelization, ctd. Discussion

- What would an ideal architecture for this algorithm look like?
  - Shared memory seems near ideal.
    - Avoids communication issues.
    - Avoids need to assign fixed regions of space to each processor.
  - Multi-threaded architecture (MTA) may have big advantages.
    - Any processor can do any node without regard to its location.
    - No need to write MPI calls.
    - No need for load balancing.

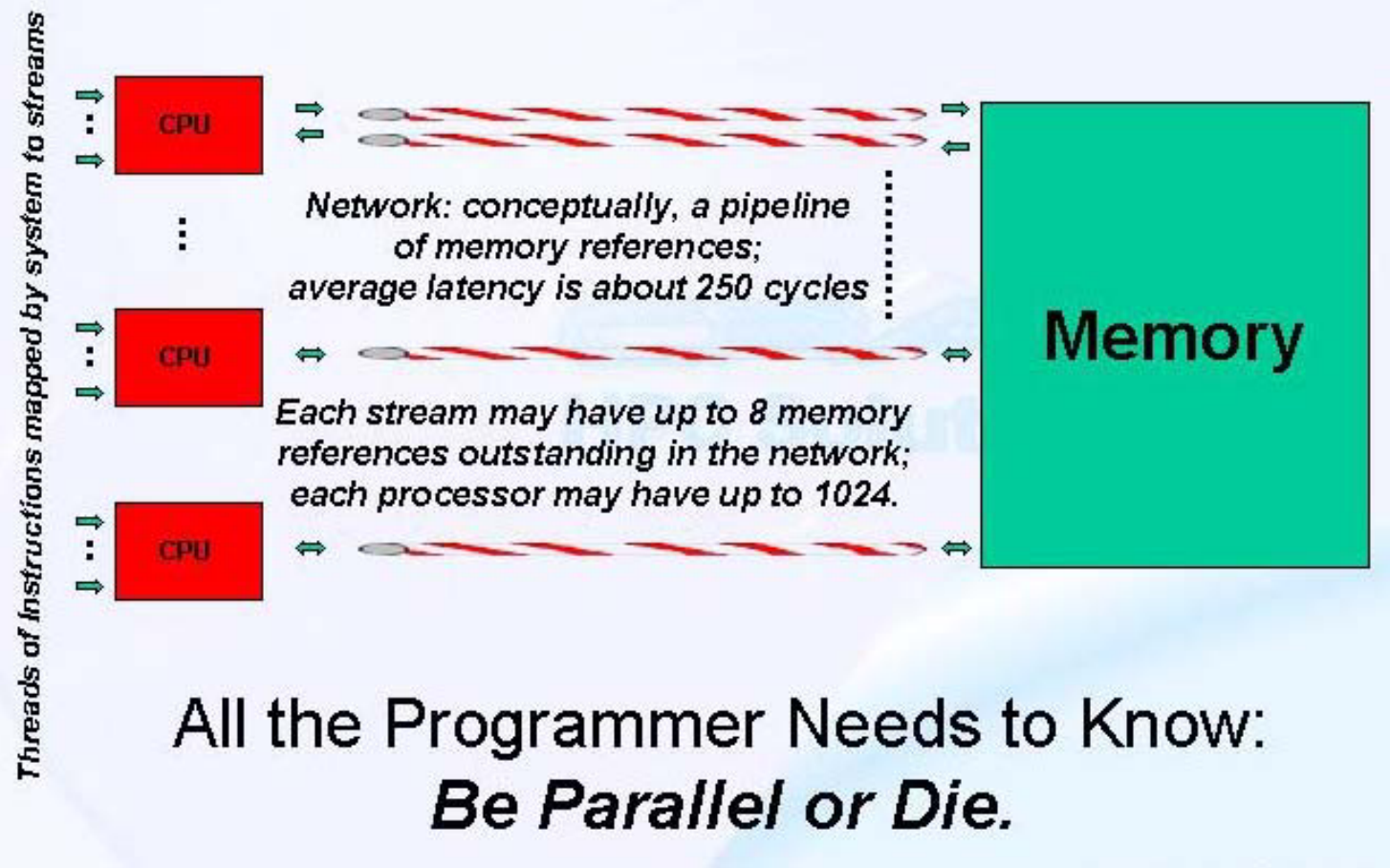
What is the Cray MTA-2?

- Large, Flat shared-memory supercomputer
  - Cacheless; no local memory; no memory hierarchy
  - 4GB/Processor spread across system
  - Largest system (Navy Research Lab) is 40p/160GB
  - Next generation (2005) scales to over 4000p/32TB
- Multi-threaded 200MHz Microprocessors
  - Up to 128 active threads each with own register set share each CPU.
  - Tolerates memory latency: while any thread has an instruction whose operands are loaded, the CPU issues instructions.
- Simple Programming Model
  - Sufficient thread parallelism => High processor utilization => scalable performance

MTA-2 Processor



Conceptual Cray MTA-2 System



EMU on the MTA-2

- EMU's computation consists of a repeating series of loops over nodes.
  - Using EMU's MPI model, these nodes are divided up amongst different processors:

```
do I = 1, MY_NODES
end do
communicate_interface_data
```

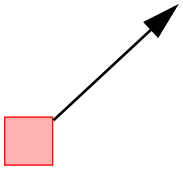
  - On the MTA, no per processor assignment is made:

```
do I = 1, ALL_NODES
end do
```

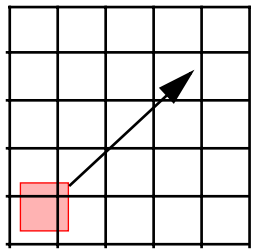
- With the help of the MTA's parallelizing compiler, each loop over nodes can (AND MUST) be parallelized.
- Scaling then depends only upon the number of nodes and the work distribution amongst them:
  - Large number of nodes =>
    large number of active threads =>
    scalable performance

EMU on the MTA-2: An example

- We ran EMU on a block of nodes moving through space.
  - On platforms that divy up the nodes amongst processors based on memory layout, the work distribution is dependent on that distribution: as the block moves, the load balance changes.
  - The shape of the block and where it is in space does not matter at all to the MTA.
  - All that matters is that there are enough nodes over which to parallelize.
  - *No changes were made to EMU to accommodate the evolution of nodes as the block moves.*



Constant velocity motion



Load balancing as described previously does not work well!

EMU on the MTA-2: Timing results from the example

EMU Performance: Rigid Block Translation
(preliminary)

System/ #Processors	Seconds for 1000 nodes:		Seconds for 1,000,000 nodes:
	total	Per iter	Per iteration
Sun*/1	316	1.19	374
SGI/1	437	--	--not measured--
SGI/6	372	--	--
MTA-2°/1	1039	3.92	1122
MTA-2/5	262	.979	228
MTA-2/10	157	.572	123

*900MHz UltraSPARC III with 32GB of memory running serial (no MPI)
°200MHz MTA-2 with 40GB of memory

Comments on MTA performance

- The performance of the 200MHz MTA-2 per processor is slower than faster clocked processors.
- 1000 nodes does not offer enough parallelism to scale to more than 6 on an MTA-2 with 10 procs.
- The larger 1,000,000 node problem provides enough parallelism to scale near perfectly on the MTA-2.
- Absolute performance of MTA-2 relative to cache-dependent processors increases with problem size.

Conclusions

- EMU, because it is based on integral equations, performs differently from traditional codes.
  - Each node interacts with many neighbors.
  - This influences communication requirements on distributed memory systems.
- Experience with EMU on the MTA-2:
  - What is required by the programmer...
    - Ensure loops are parallel -- period.
    - Larger problems should scale to larger MTA.
    - Shared scalars accessed by too many threads may lead to “hot spots” that require mitigation.
  - Evolution of EMU, e.g., to adaptive grid, is likely to be straightforward on the MTA.