CallaghanInnovation

Fracture Effects Update

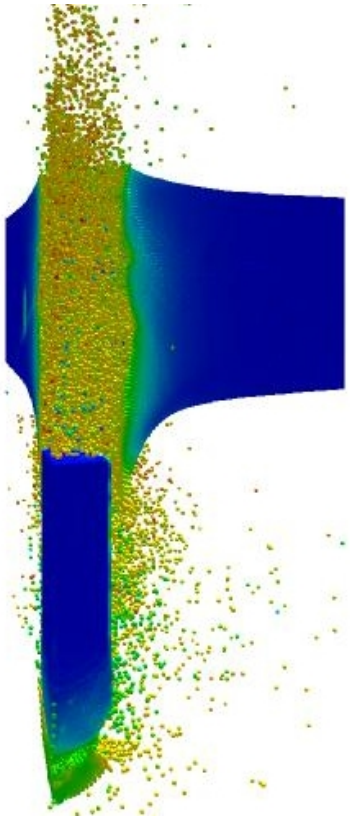
Biswajit Banerjee

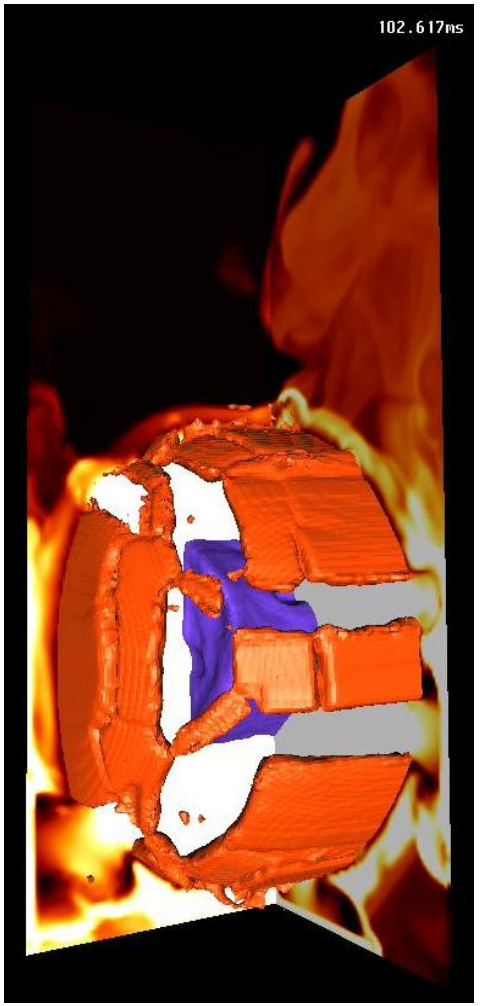
15 April, 2013

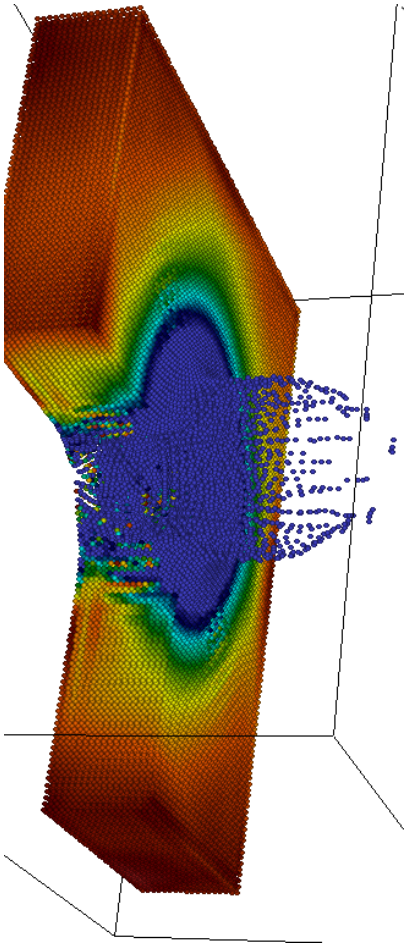
Biswajit.Banerjee@callaghaninnovation.govt.nz

Fracture simulation

MPM simulations with Uintah

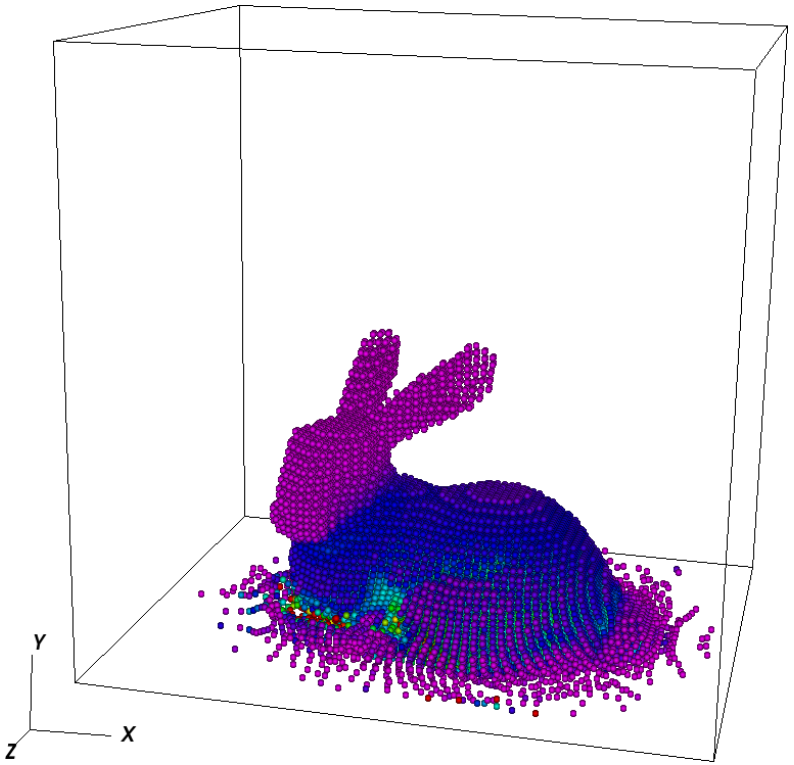


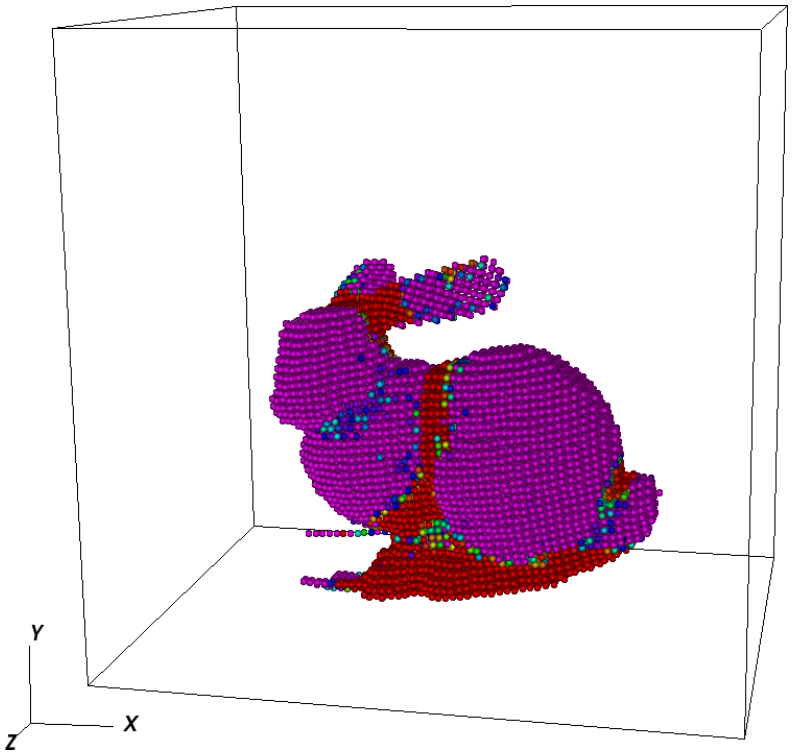




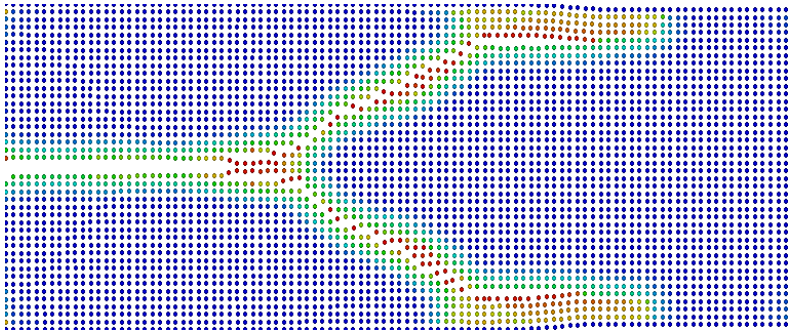
MPM simulations with Vaango

Notice that large regions remain relatively rigid.

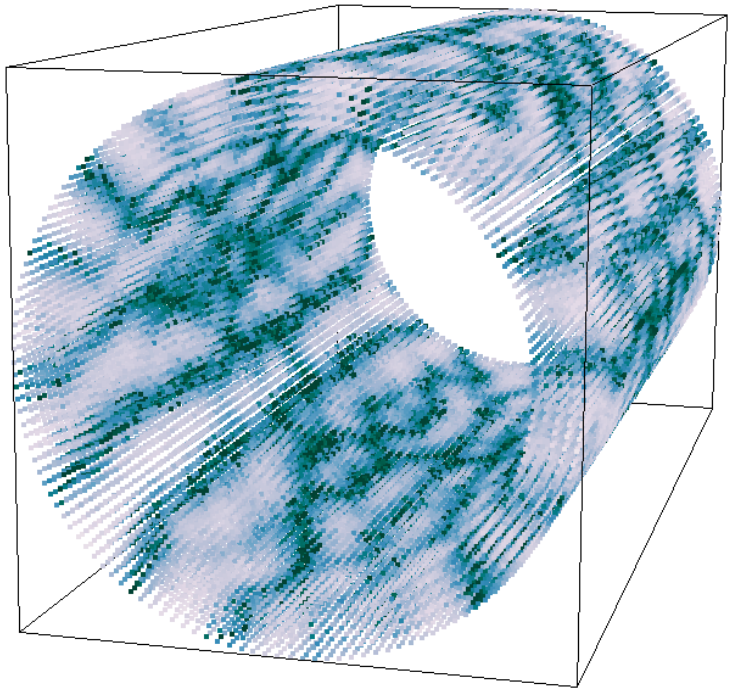




Peridynamics simulations

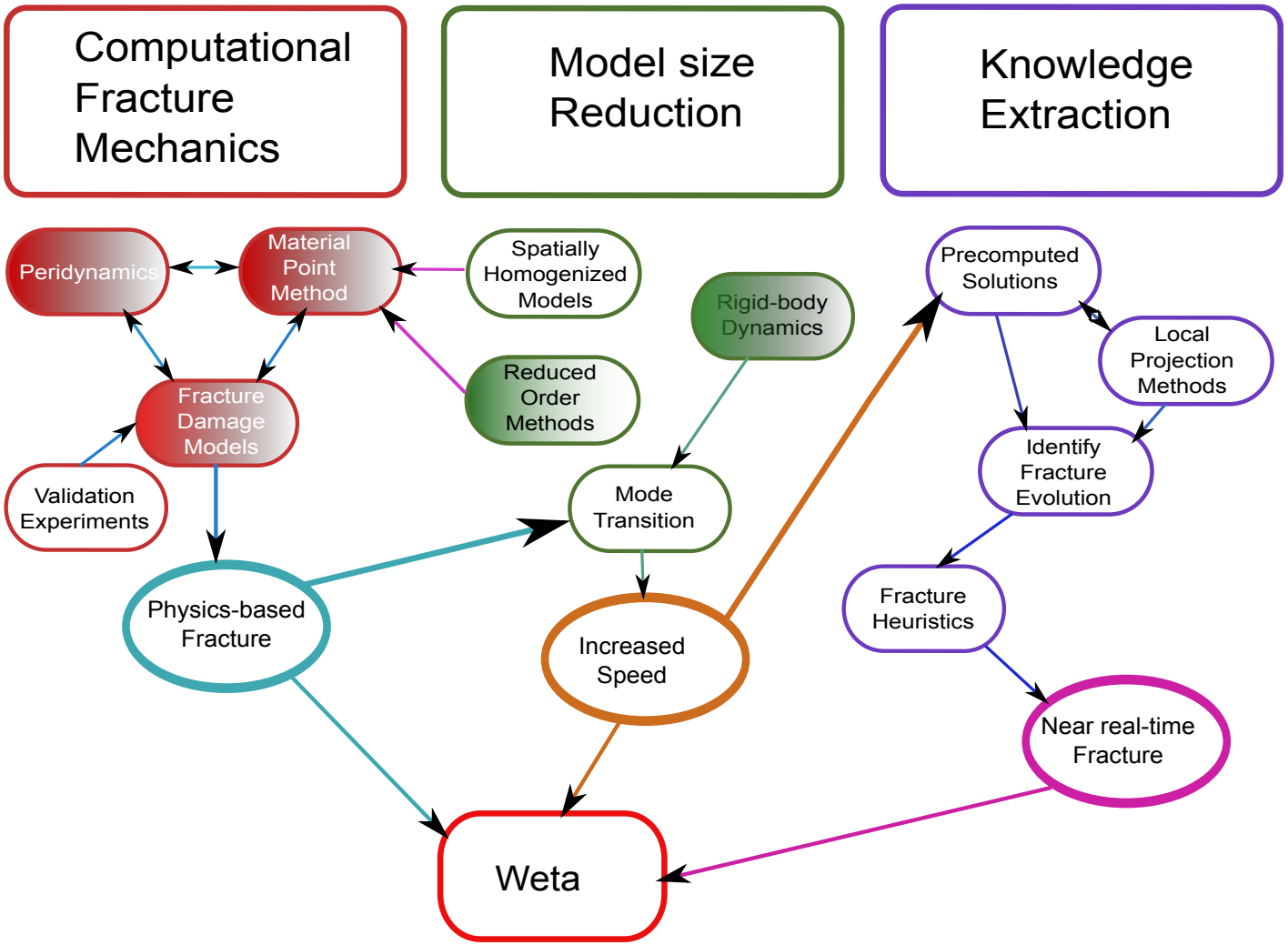


EMUNE



Peridigm

Project plan



Short-term Tasks

- Biswajit: Serial and Parallel implementations of Peridynamics.
- Bryan: Rigid-body dynamics with MPM.
- Kumar: Contact algorithms and anisotropic material models.
- Florin: Anisotropic peridynamic fracture for wood/plasterboard.
- Andreas: Extraction of fracture surfaces from particle simulations for rendering.
- Rojan: Model-order reduction approaches for fracture.
- Hooman: Hybrid MPM-Peridynamics approaches.

Recent progress

- Serial multibody peridynamics code being developed
- Several approaches tried (e.g., Matiti/Matiti2D/EMU2DC)
- Currently developing EMU2DC
- The serial version will be used by Hooman for his work

The input file

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!-- <!DOCTYPE Vaango SYSTEM "input.dtd"> -->
<!-- @version: -->
<Vaango>
  <Meta>
    <title> Test input file for peridynamics </title>
  </Meta>
  <Time>
    <max_time> 1.0 </max_time>
    <max_iterations> 500 </max_iterations>
    <delt> 0.00000002 </delt>
  </Time>
  <Output>
    <output_file> test_output.dat </output_file>
    <output_iteration_interval> 25 </output_iteration_interval>
  </Output>
  <Peridynamics>
    <simulation_type> dynamic  </simulation_type>
    <modulus_type>    constant </modulus_type>
    <horizon_factor>  4.01     </horizon_factor>
  </Peridynamics>
  ..............................
</Vaango>
```

Domain

```
<Domain>
  <min> [0.0, −2.0, 0.0] </min>
  <max> [4.0, 2.0, 0.0] </max>
  <num_cells> [10, 10, 1] </num_cells>
  <BoundaryConditions>
    <VelocityBC>
      <velocity> [0.0, 1.4e7, 0.0] </velocity>
      <Area>
        <point> [0.0, −0.2, 0.0] </point>
        <point> [1.0, −0.2, 0.0] </point>
      </Area>
    </VelocityBC>
    <VelocityBC>
      <velocity> [0.0, −1.4e7, 0.0] </velocity>
      <Area>
        <point> [0.0, 0.2, 0.0] </point>
        <point> [1.0, 0.2, 0.0] </point>
      </Area>
    </VelocityBC>
  </BoundaryConditions>
</Domain>
```

Material

```
<Material name="material␣1">
  <young_modulus> 72.0e9 </young_modulus>
  <density> 2440.0 </density>
  <fracture_energy> 135.0 </fracture_energy>
  <DamageModel>
    <damage_viscosity> [0.0, 0.05, 0.0] </
        damage_viscosity>
    <damage_index> 0.35 </damage_index>
    <damage_stretch> [0.0, 0.0, 1.0] </
        damage_stretch>
  </DamageModel>
</Material>

<Material name="material␣2">
  <young_modulus> 72.0e8 </young_modulus>
  <density> 244.0 </density>
  <fracture_energy> 13.5 </fracture_energy>
  <DamageModel>
    <damage_viscosity> [0.0, 0.005, 0.0] </
        damage_viscosity>
    <damage_index> 0.035 </damage_index>
    <damage_stretch> [0.0, 0.0, 0.1] </
        damage_stretch>
  </DamageModel>
</Material>
```

```
MaterialSPArray mat_list;
int count = 0;
for (Uintah::ProblemSpecP mat_ps = ps->findBlock(
       "Material"); mat_ps != 0;
     mat_ps = mat_ps->findNextBlock("Material"))
           {
  MaterialSP mat = std::make_shared<Material>();
  mat->initialize(mat_ps);
  mat->id(count);
  mat_list.emplace_back(mat);
  ++count;
  std::cout << *mat << std::endl;
}
```

Body

```
<Body name="body␣1">
  <material name="material␣1"/>
  <Geometry>
    <input_node_file>    nodes_test_103by42.txt   </input_node_file>
    <input_element_file> element_test_103by42.txt </input_element_file>
  </Geometry>
  <InitialConditions>
    <velocity> [0.0, 0.0, 0.0] </velocity>
    <Crack>
      <LineString>
        <point> [−0.05, 0.0, 0.0] </point>
        <point> [−0.04, 0.0, 0.0] </point>
        <point> [−0.03, 0.0, 0.0] </point>
      </LineString>
    </Crack>
    <Crack>
      <LineString>
        <point> [0.04, 0.0, 0.0] </point>
        <point> [0.05, 0.0, 0.0] </point>
      </LineString>
    </Crack>
  </InitialConditions>
  <BoundaryConditions>
    <ExtForce>
      <force> [0.0, 1.4e7, 0.0] </force>
      <min> [0.0, −0.2, 0.0] </min>
      <max> [1.0, −0.2, 0.0] </max>
    </ExtForce>
    .......
  </BoundaryConditions>
</Body>
```

FamilyComputer

```
namespace Emu2DC {

  class FamilyComputer {
  public:
    /**
     * Create an empty BondfamilyComputer object
     */
    FamilyComputer();
    ~FamilyComputer();
    /**
     *  Find which cells the nodes sit in and
           create a unordered map that maps
           nodes to cells
     *
     * @param domain Reference to the domain
           object
     * @param nodeList Reference to the vector of
           NodeP objects inside the domain
     */
    void createCellNodeMap(const Domain& domain,
                           const NodePArray&
                                 nodeList);

    ...
  };
}
```

```
typedef std::tr1::unordered_multimap<long64,
        NodeP, Hash64> CellNodePMap;
typedef CellNodePMap::iterator
        CellNodePMapIterator;
typedef std::pair<long64, NodeP> CellNodePPair;
```

```
// using stdlib shared_ptr instead of SCIRun::
       Handle
class Node;
typedef std::shared_ptr<Node> NodeP;
```

```
// function object class for Hashing with
       lookup3
struct Hash64 {
  std::size_t operator() (const long64& cellID)
          const {
    const u8* key = (const u8*) &cellID;
    u32 len = sizeof(cellID);
    u32 seed = 13;

    return lookup3((const u8*) &key, sizeof(key
           ), 13 );
  }
```

Lessons so far

- Learning recent developments in C++
- Keeping code simple but general is not easy
- ....

Questions?