Representing Arbitrary Bounding Surfaces in the Material Point Method

Carter M. Mast

6th MPM Workshop
Albuquerque, New Mexico
August 9-10, 2010

Participants:

Peter Mackenzie-Helnwein, Pedro Arduino, and Greg Miller

Department of Civil and Environmental Engineering – University of Washington – Seattle, WA

Outline

- Motivation and Overview
- Approach
- Implementation
- Outlook/Future Research

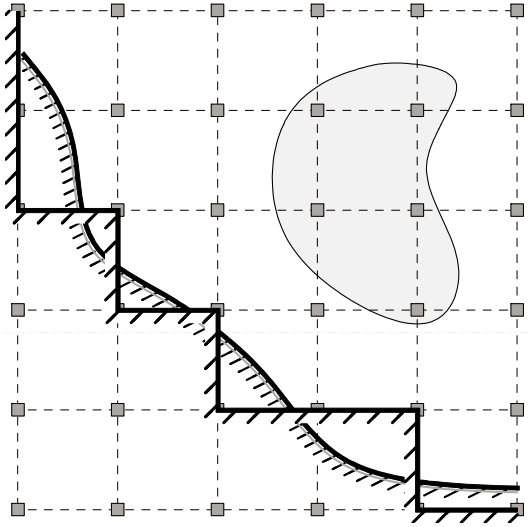
Motivation

- Loading on structures due to landslide/debris flows
  - Landslide/Debris flow
    - Appropriate numerical method (MPM)
      - Material models
      - Phase transition
      - Etcetera
  - Domain
    - Topological --- e.g. hillside
    - Structural
    - Both require general surfaces

Motivation

- Loading on structures due to landslide/debris flow
  - Landslide/Debris flow
    - Appropriate numerical method (MPM)
      - Material models
      - Phase transition
      - Etcetera
  - Domain
    - Topological --- e.g. hillside
    - Structural
    - Both require general surfaces

Overview



- Disadvantages of this surface representation:
  - Surface is dependent on the computational nodes
  - Unrealistic

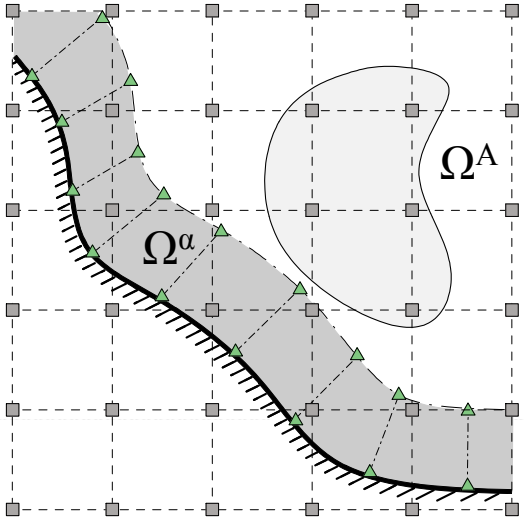
Overview

- Potential solutions/fix to the surface representation problem:
  - Refine the mesh
  - Represent the surface as a rigid body
  - Irregular mesh over entire domain
    - Computational expensive
      - Increase total number of particles
      - Search algorithm
      - Meshing algorithm

Overview

- Potential solutions to the surface representation problem:
  - Refine the mesh
  - Represent the surface as a rigid body
  - Irregular mesh over entire domain
    - Computational expensive
      - Increase total number of particles
      - Search algorithm
      - Meshing algorithm
  - Introduce a second grid
    - “Dual-Grid Approach”

Approach



- Dual-Grid methodology
  - Introduce a separate (additional) grid that follows the geometry of the bounding surface
  - Two grids, one body
  - Effectively communicate dynamic information between the two grids

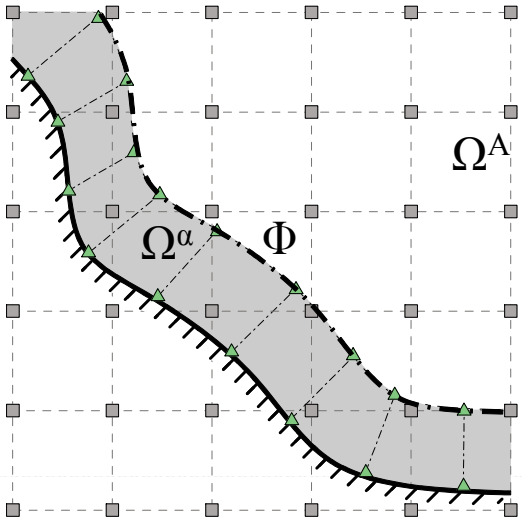
Approach

- Dual-Grid methodology
  - *The Blending Approach*
    - Each grid is used to create independent fields for velocity and acceleration
    - Piecewise description:

$$\boldsymbol{v}(\boldsymbol{x},t) \approx \begin{cases} \boldsymbol{v}^{\alpha}(\boldsymbol{x},t) := \sum_{S} N_S^{\alpha}(\boldsymbol{x})\, \boldsymbol{v}_S^{\alpha} & \text{if } \boldsymbol{x} \in \Omega^{\alpha} \\ \boldsymbol{v}^{A}(\boldsymbol{x},t) := \sum_{J} N_J^{A}(\boldsymbol{x})\, \boldsymbol{v}_J^{A} & \text{if } \boldsymbol{x} \in \Omega^{A} \end{cases}$$

$$\dot{\boldsymbol{v}}(\boldsymbol{x},t) \approx \begin{cases} \dot{\boldsymbol{v}}^{\alpha}(\boldsymbol{x},t) := \sum_{S} N_S^{\alpha}(\boldsymbol{x})\, \dot{\boldsymbol{v}}_S^{\alpha} & \text{if } \boldsymbol{x} \in \Omega^{\alpha} \\ \dot{\boldsymbol{v}}^{A}(\boldsymbol{x},t) := \sum_{J} N_J^{A}(\boldsymbol{x})\, \dot{\boldsymbol{v}}_J^{A} & \text{if } \boldsymbol{x} \in \Omega^{A} \end{cases}$$

Approach



- *The Blending Approach*
  - Enforce continuity along Φ

$$\boldsymbol{v}^A(\boldsymbol{x},t) = \boldsymbol{v}^\alpha(\boldsymbol{x},t) \quad \text{and} \quad \dot{\boldsymbol{v}}^A(\boldsymbol{x},t) = \dot{\boldsymbol{v}}^\alpha(\boldsymbol{x},t) \quad \forall\, \boldsymbol{x} \in \Phi$$

  - Leads to the constraint of the form

$$\int_\Phi (\boldsymbol{v}^A - \boldsymbol{v}^\alpha) \cdot \boldsymbol{\lambda}\, d\Phi = 0 \quad \Longrightarrow \quad \boldsymbol{\lambda}(\boldsymbol{x},t) \approx \boldsymbol{\lambda}^h(\boldsymbol{x},t) := \sum_\mu S_\mu(\boldsymbol{x})\, \boldsymbol{\lambda}_\mu(t) \quad \forall \boldsymbol{x} \in \Phi$$

Approach

- *The Blending Approach*
  - Algorithmic implementation:
    1. Use the traditional MPM algorithm to solve for nodal acceleration and velocity at time t_n for those nodes in the boundary grid.
    2. Solve for the nodal accelerations on the standard grid.

$$\begin{bmatrix} m^A_{IJ}\mathbf{1} & R^A_{I\mu}\mathbf{1} \\ R^A_{J\kappa}\mathbf{1} & \mathbf{0} \end{bmatrix} \begin{Bmatrix} \dot{\boldsymbol{v}}^A_J \\ \dot{\boldsymbol{\lambda}}_\mu \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{f}^{A,ext}_I + \boldsymbol{f}^{A,\sigma}_I \\ \hat{\dot{\boldsymbol{v}}}^\alpha_\kappa \end{Bmatrix}$$

$$R^A_{J\kappa} = \int_\Phi N^A_J(\boldsymbol{x}) S_\kappa(\boldsymbol{x})\, d\Phi \quad \text{and} \quad R^\alpha_{S\kappa} = \int_\Phi N^\alpha_S(\boldsymbol{x}) S_\kappa(\boldsymbol{x})\, d\Phi$$

Approach

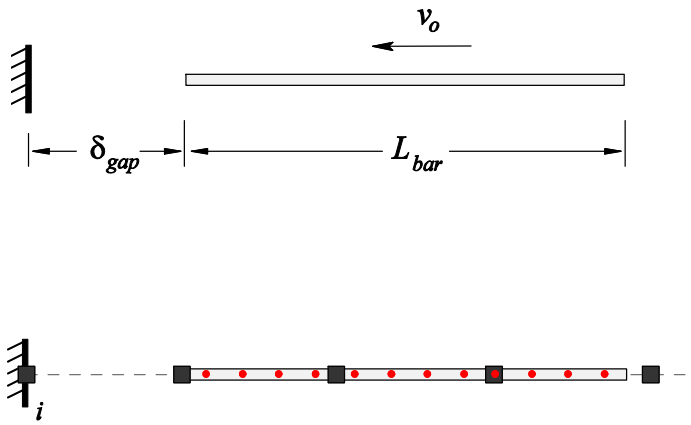
- *The Blending Approach*
  - Algorithmic implementation:
    3. Solve for the nodal velocities on the standard grid.

$$\begin{bmatrix} m^A_{IJ}\mathbf{1} & R^A_{I\mu}\mathbf{1} \\ R^A_{J\kappa}\mathbf{1} & \mathbf{0} \end{bmatrix} \begin{Bmatrix} \boldsymbol{v}^A_{J,n} \\ \boldsymbol{\lambda}_\mu \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{p}^A_I \\ \hat{\boldsymbol{v}}^\alpha_\kappa \end{Bmatrix}$$

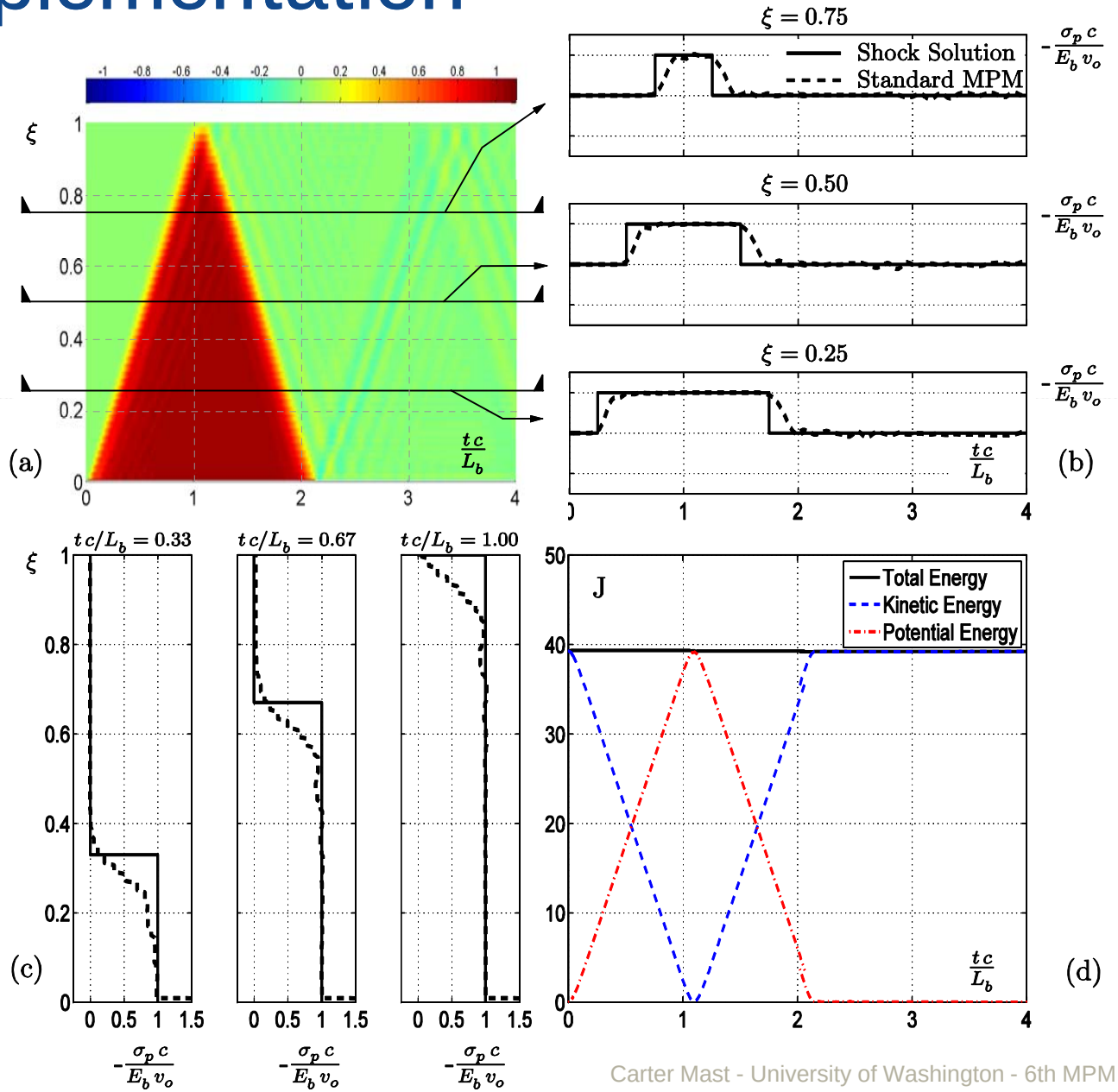
    4. Update nodal values for both grids.
    5. Update particles:
       a. For particles with $\boldsymbol{x}_p \in \Omega^\alpha$ then the update comes from boundary grid nodes.
       b. For particles with $\boldsymbol{x}_p \in \Omega^A$ then the update comes form standard grid nodes.

Implementation

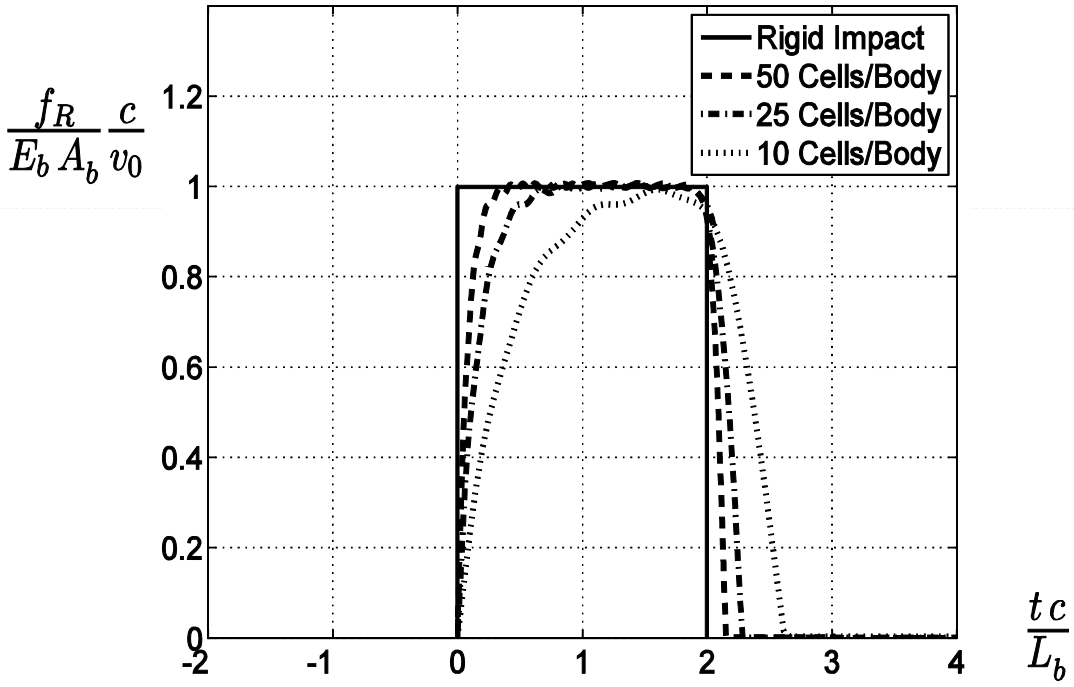
- Evaluate algorithm using one-dimensional test case
  - Uniaxial steel bar subjected to rigid boundary
  - Standard MPM:



Implementation

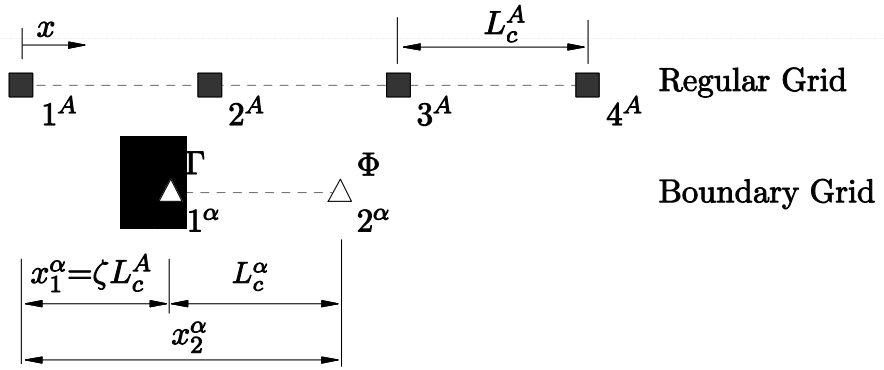


Implementation

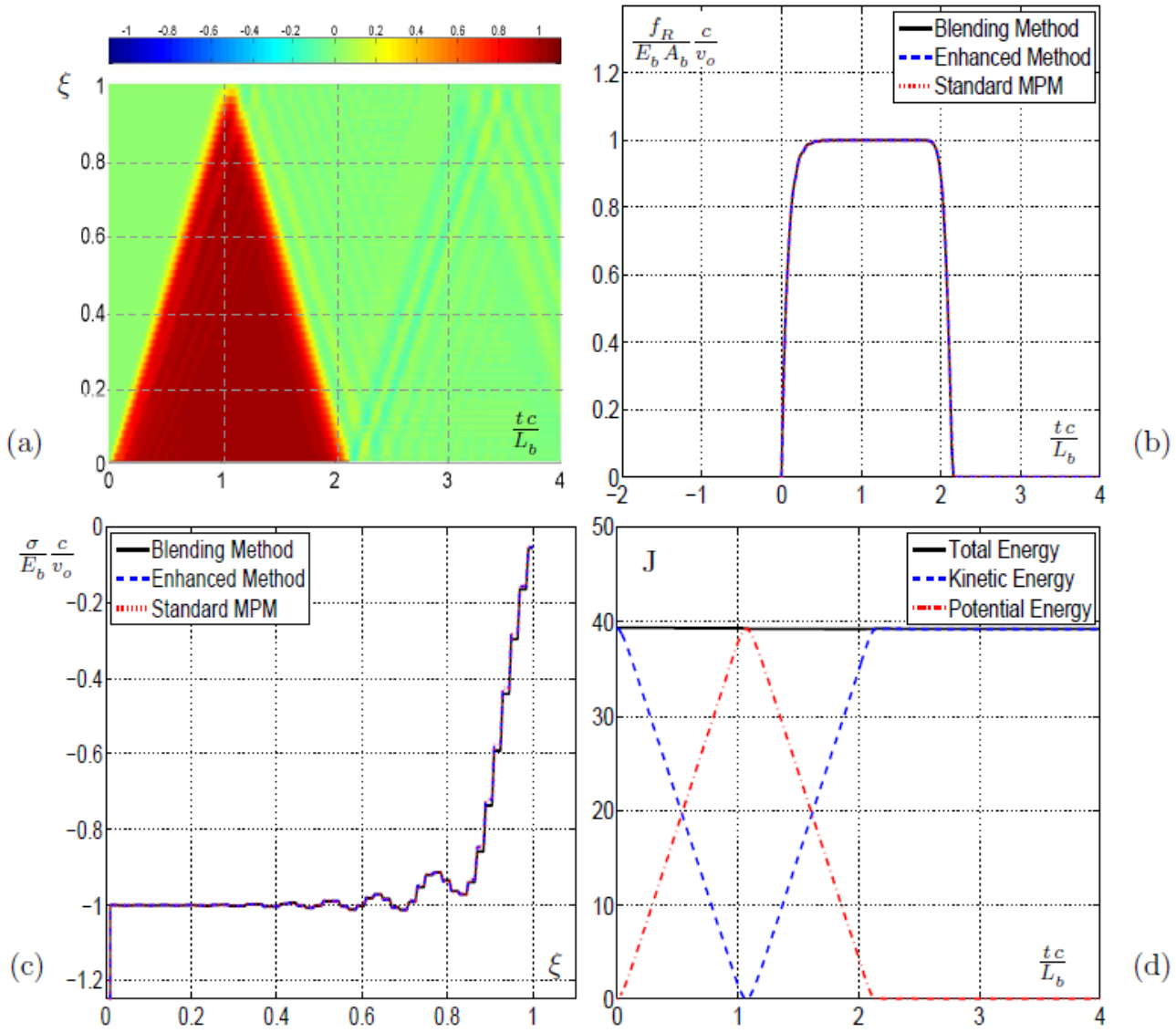


Implementation

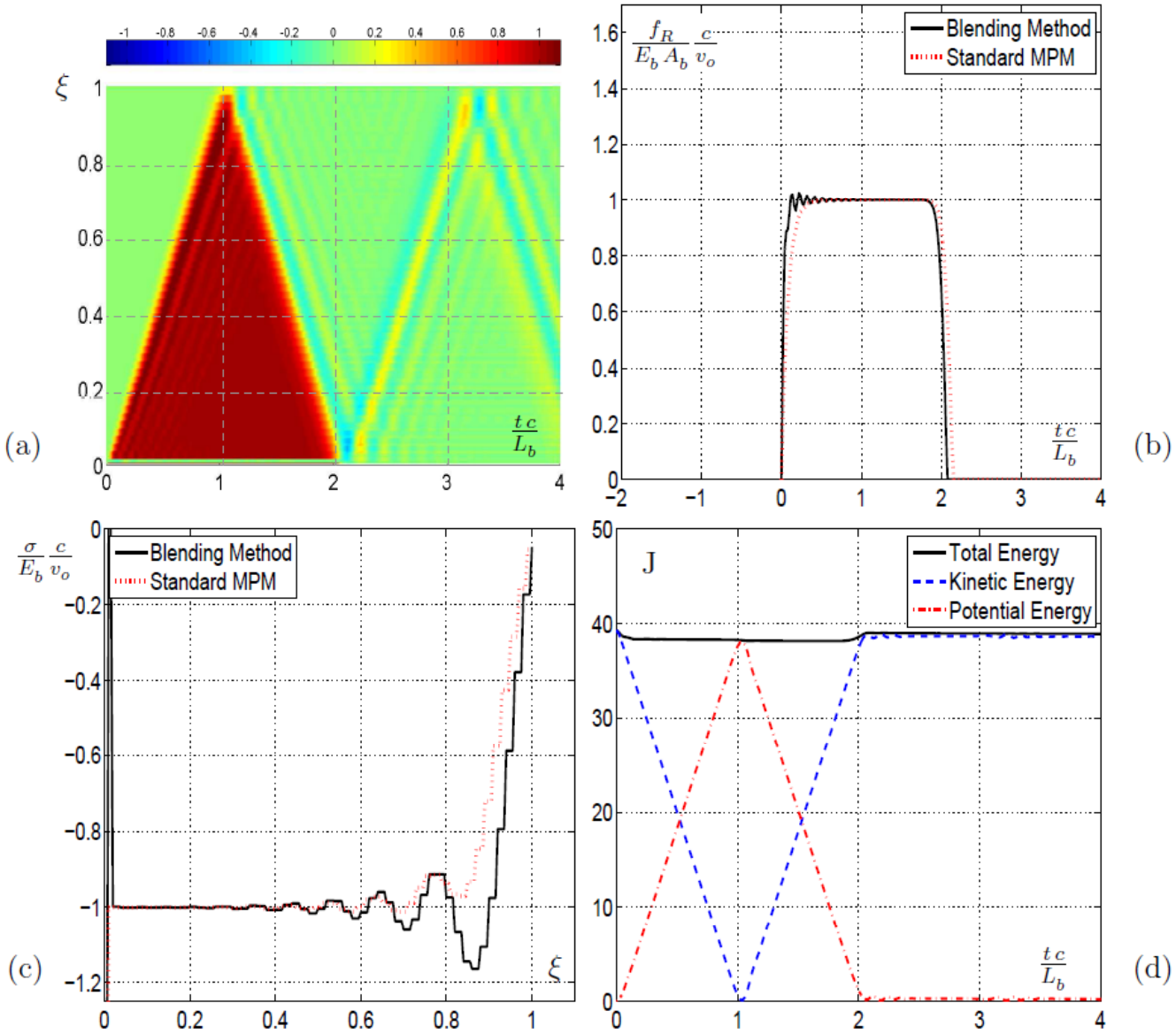
- Arbitrary boundary representation in 1-d:



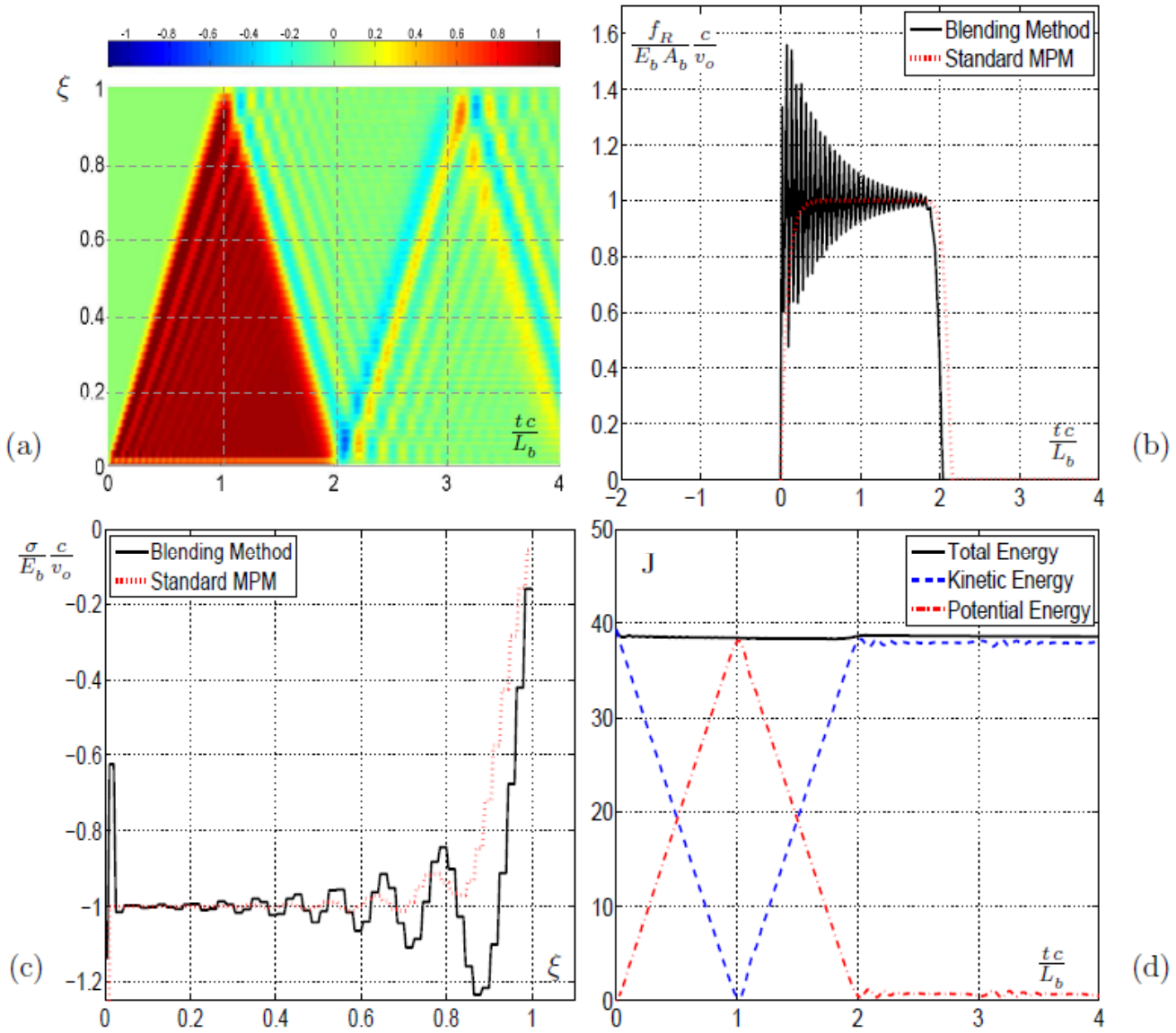
Implementation



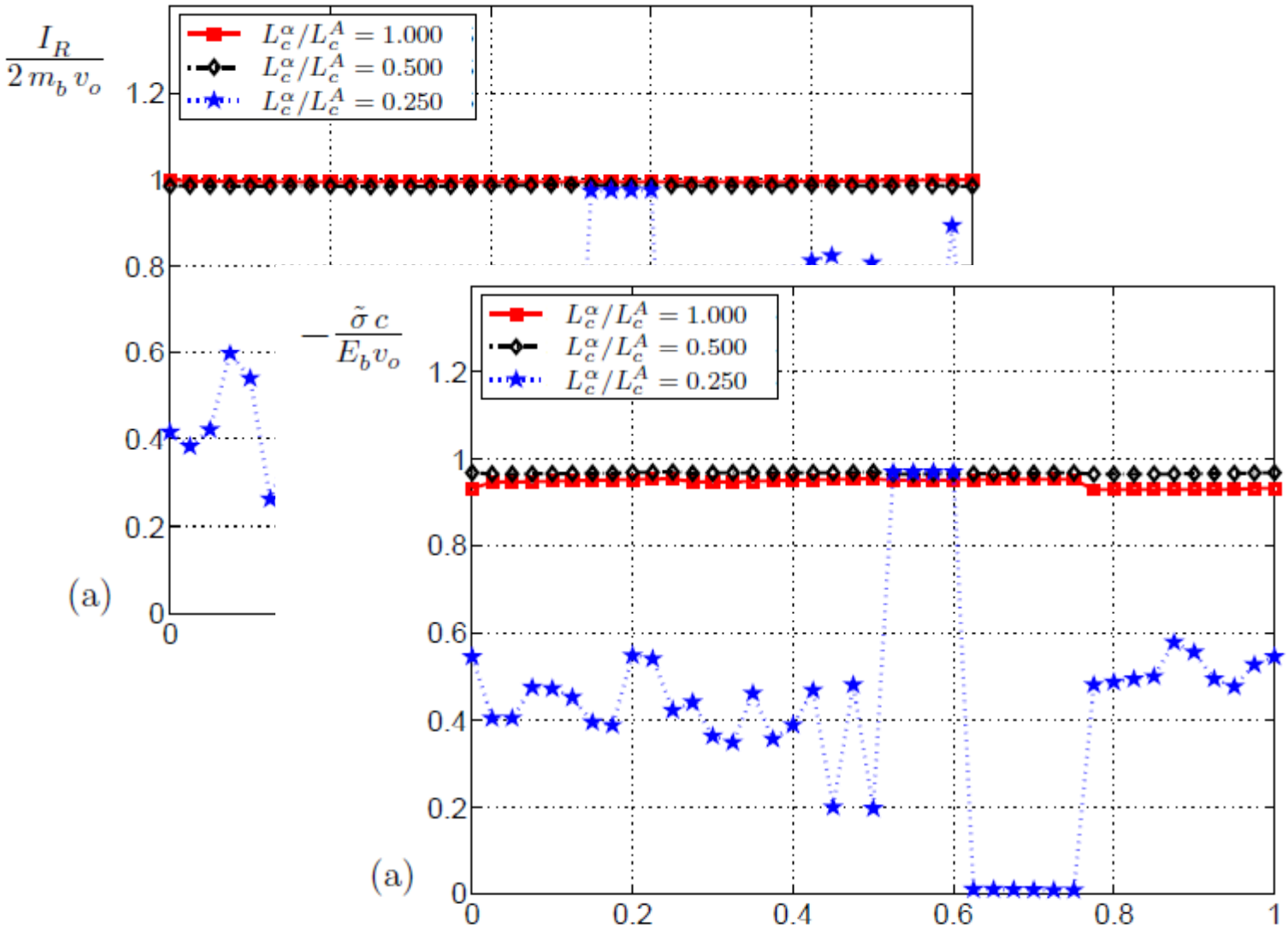
Implementation



Implementation



Implementation



Implementation

- From the 1-d results:
  - Certain configurations (boundary location and boundary cell size) are problematic.
    - Particularly for the *Enhanced* approach.
  - The *Blending* approach provides more consistent results
    - Boundary grid size should be similar to the standard grid size.

Implementation

- From the 1-d results:
  - Certain configurations (boundary location and boundary cell size) are problematic.
    - Boundary grid size should be similar to the standard grid size.

Implementation

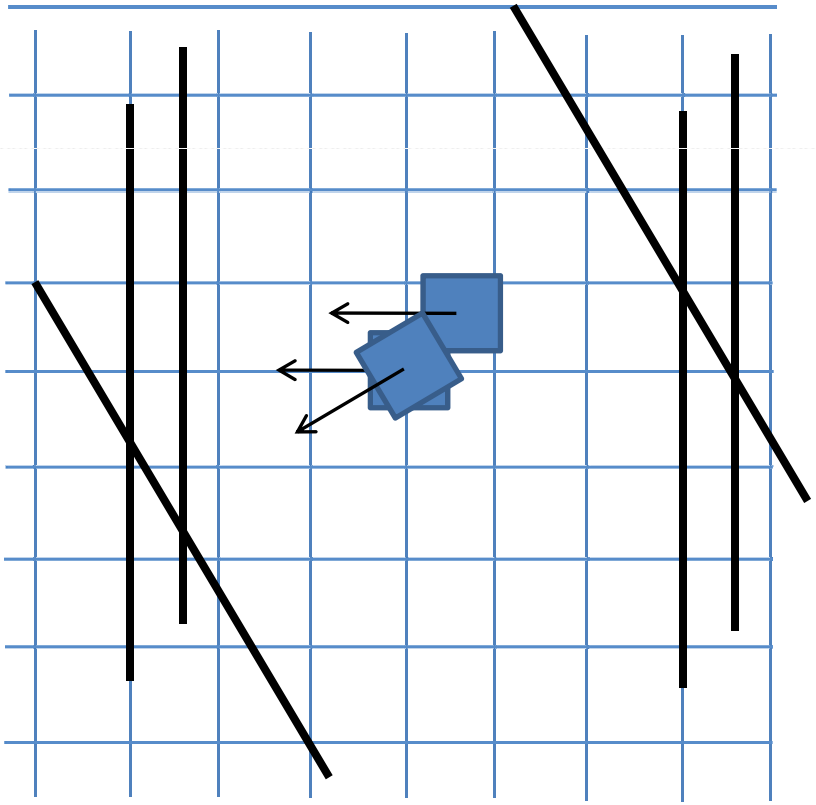
- Moving into 2- and 3-d...
  - Increasing complexity
    - Boundary orientation
    - Evaluation of the integral linking the two grids

$$R^{\alpha}_{S\kappa} = \int_{\Phi} N^{\alpha}_{S}(\boldsymbol{x}) S_{\kappa}(\boldsymbol{x})\, d\Phi$$

  - Fully 3-d code restricted to planar boundaries with regular node spacing

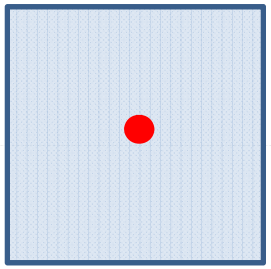
Implementation

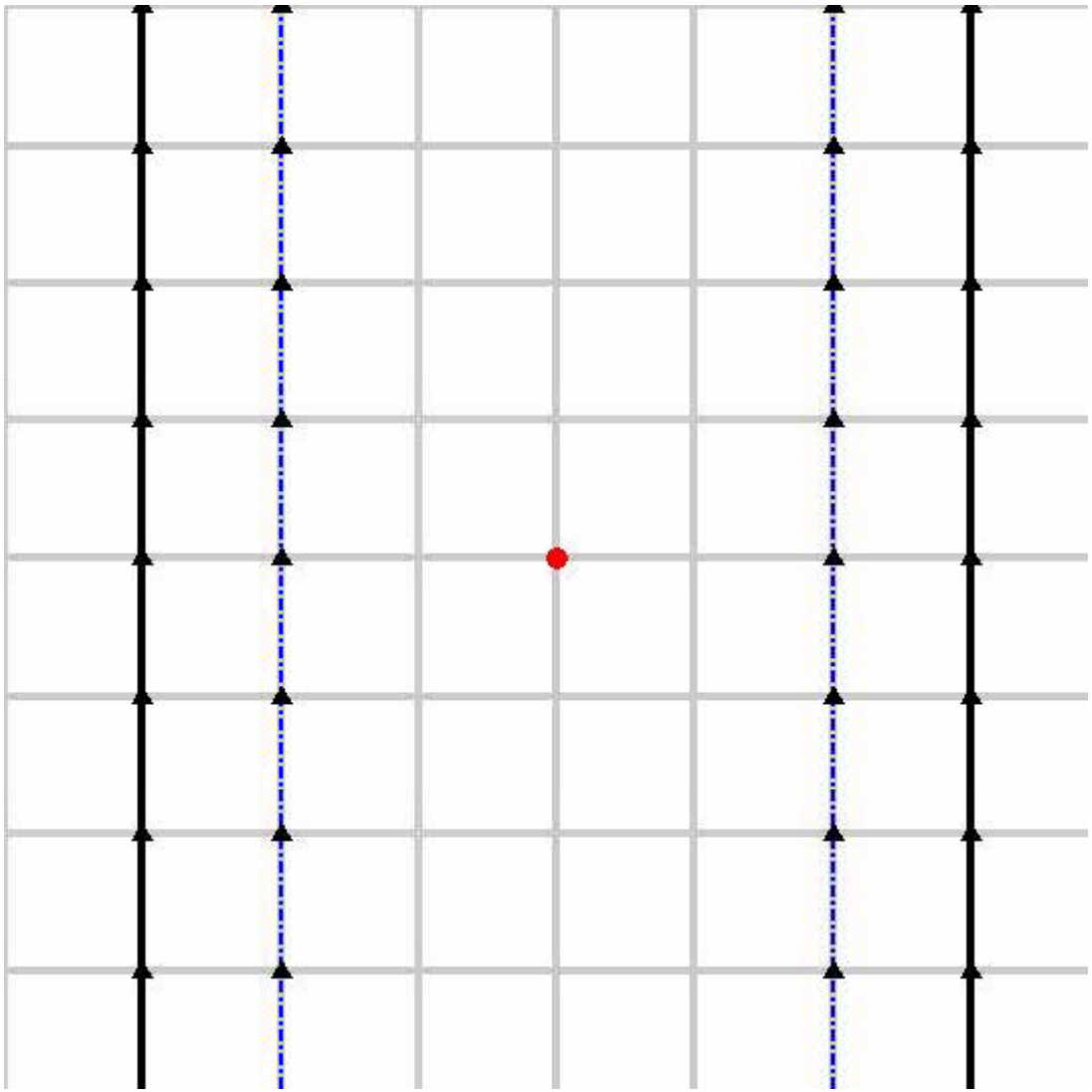
- A relatively straight forward problem...

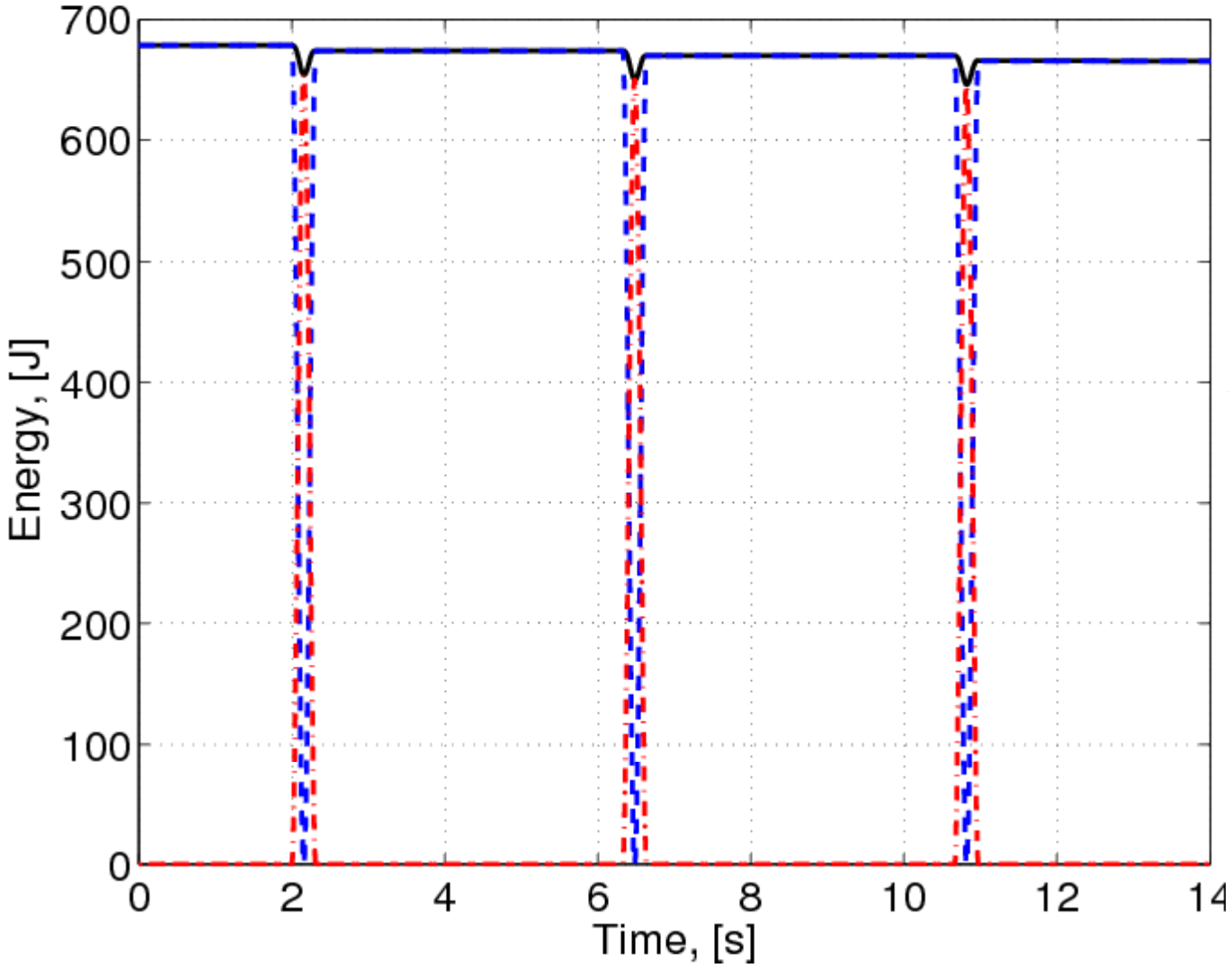


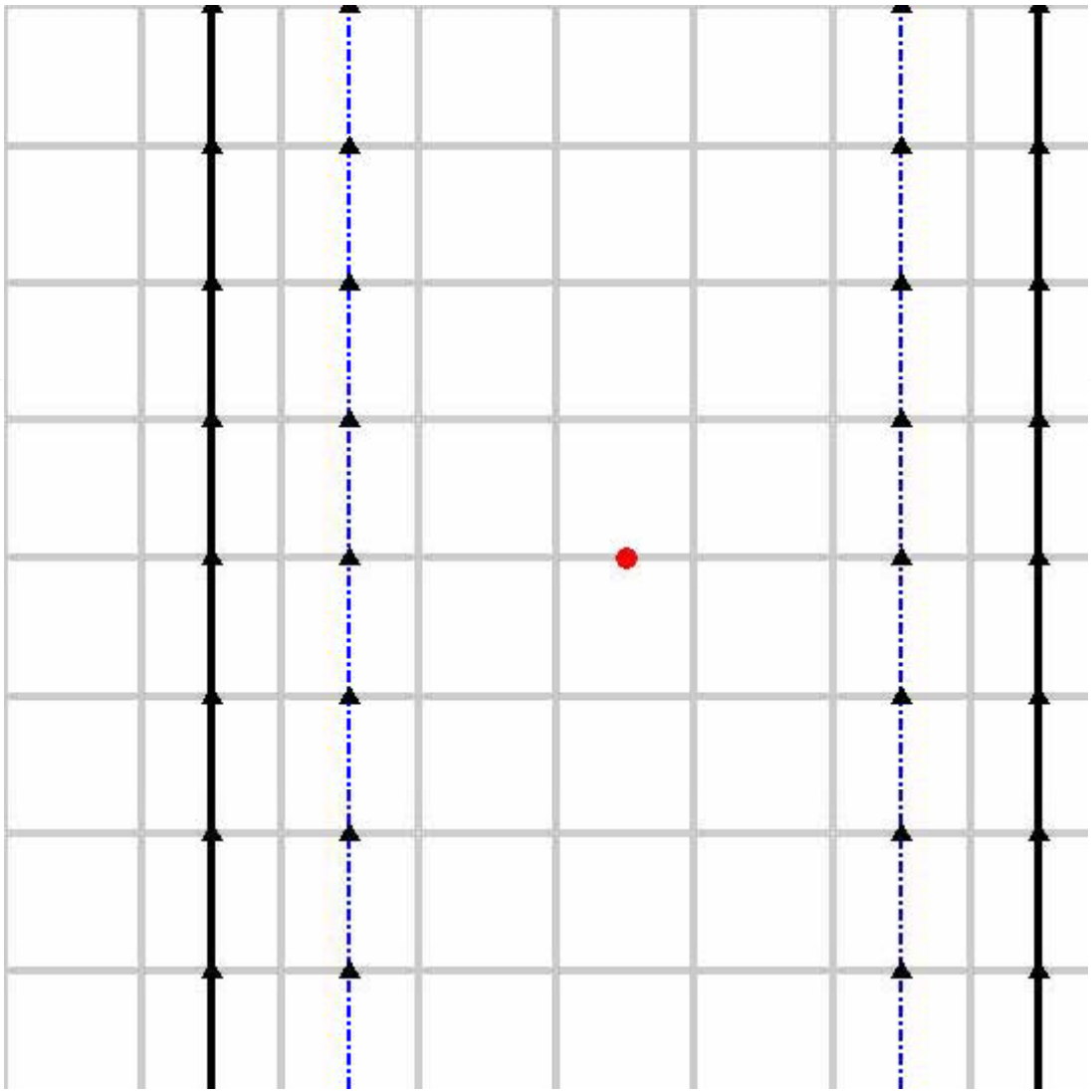
Implementation

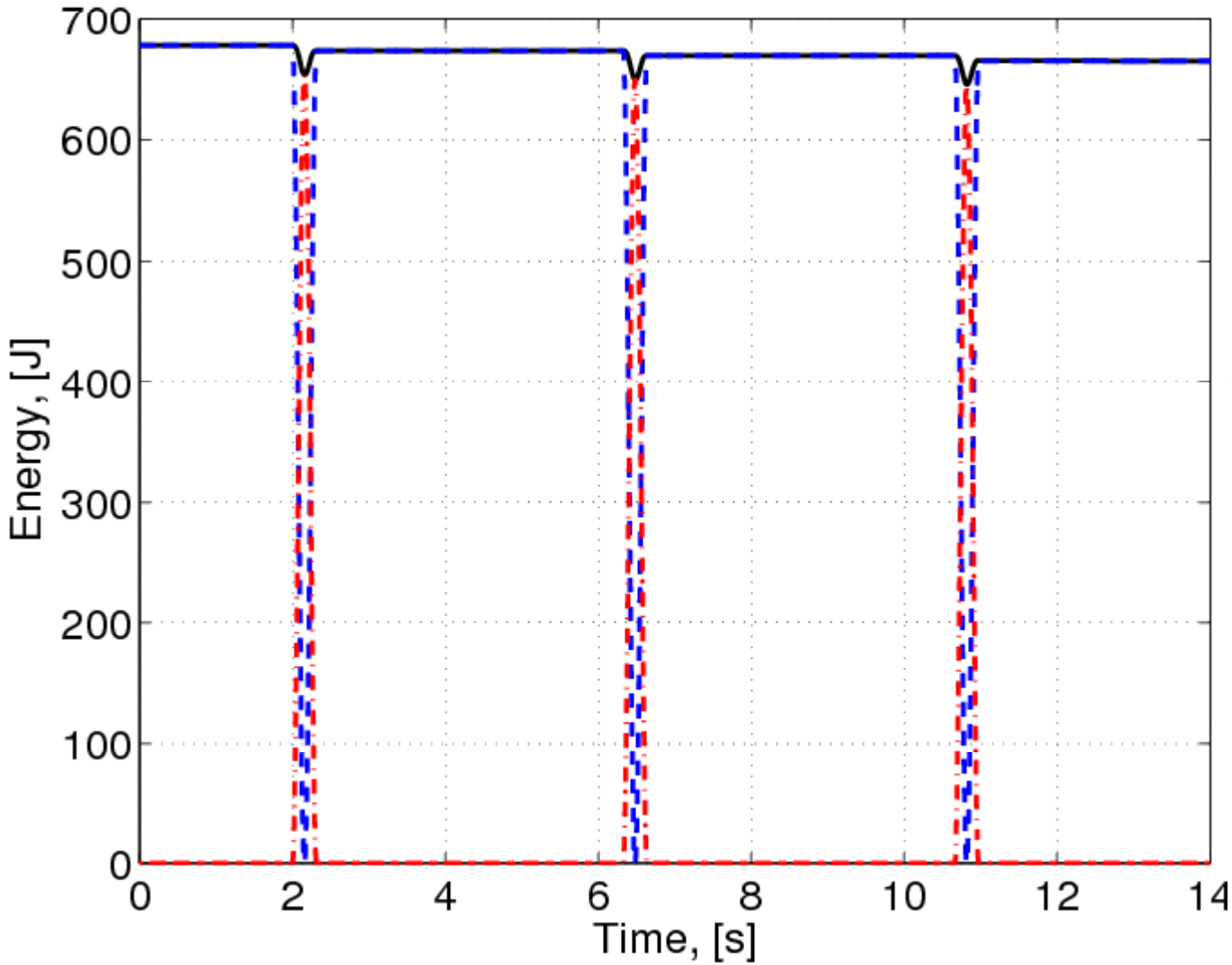
- Single particle analysis

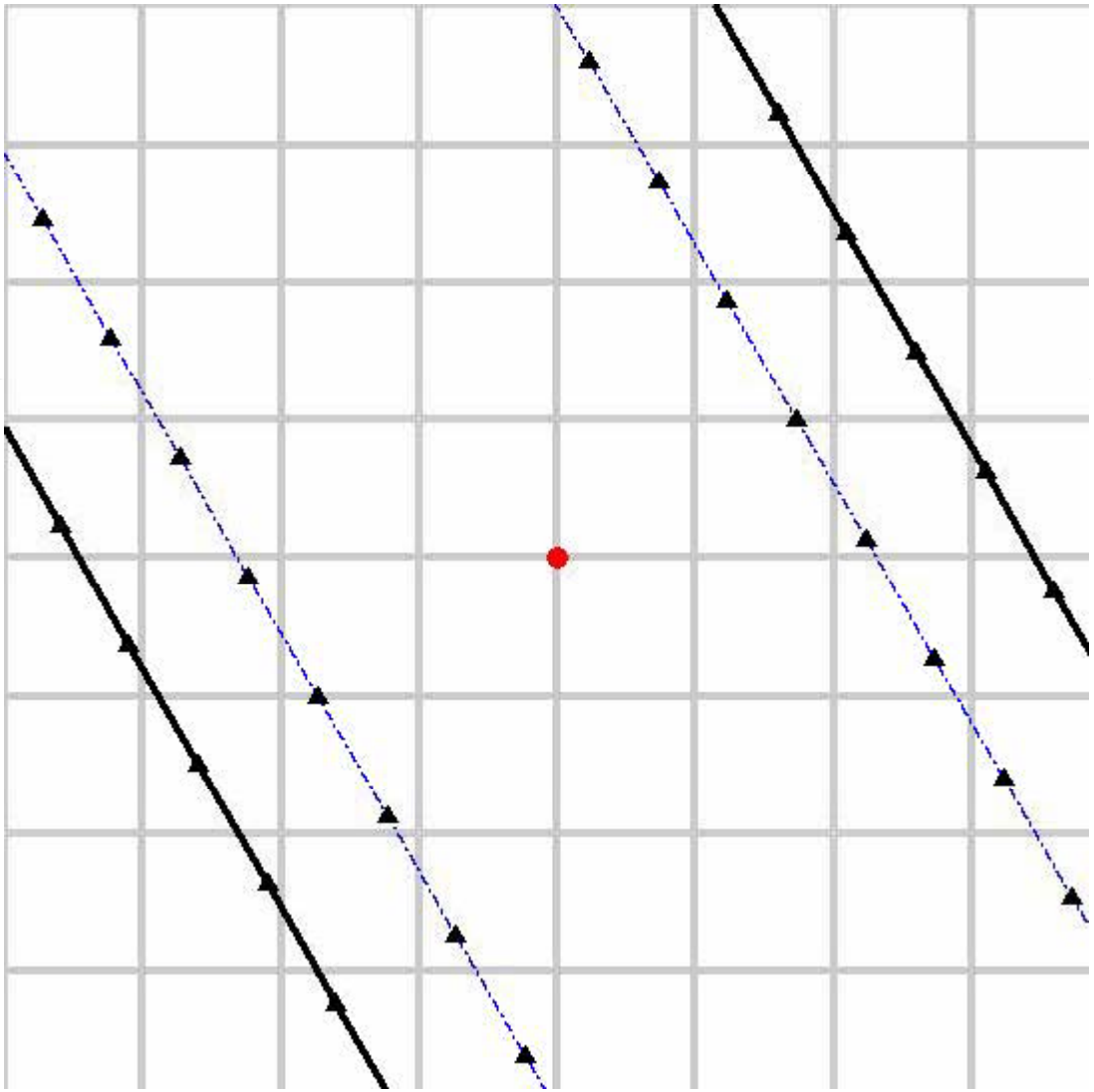


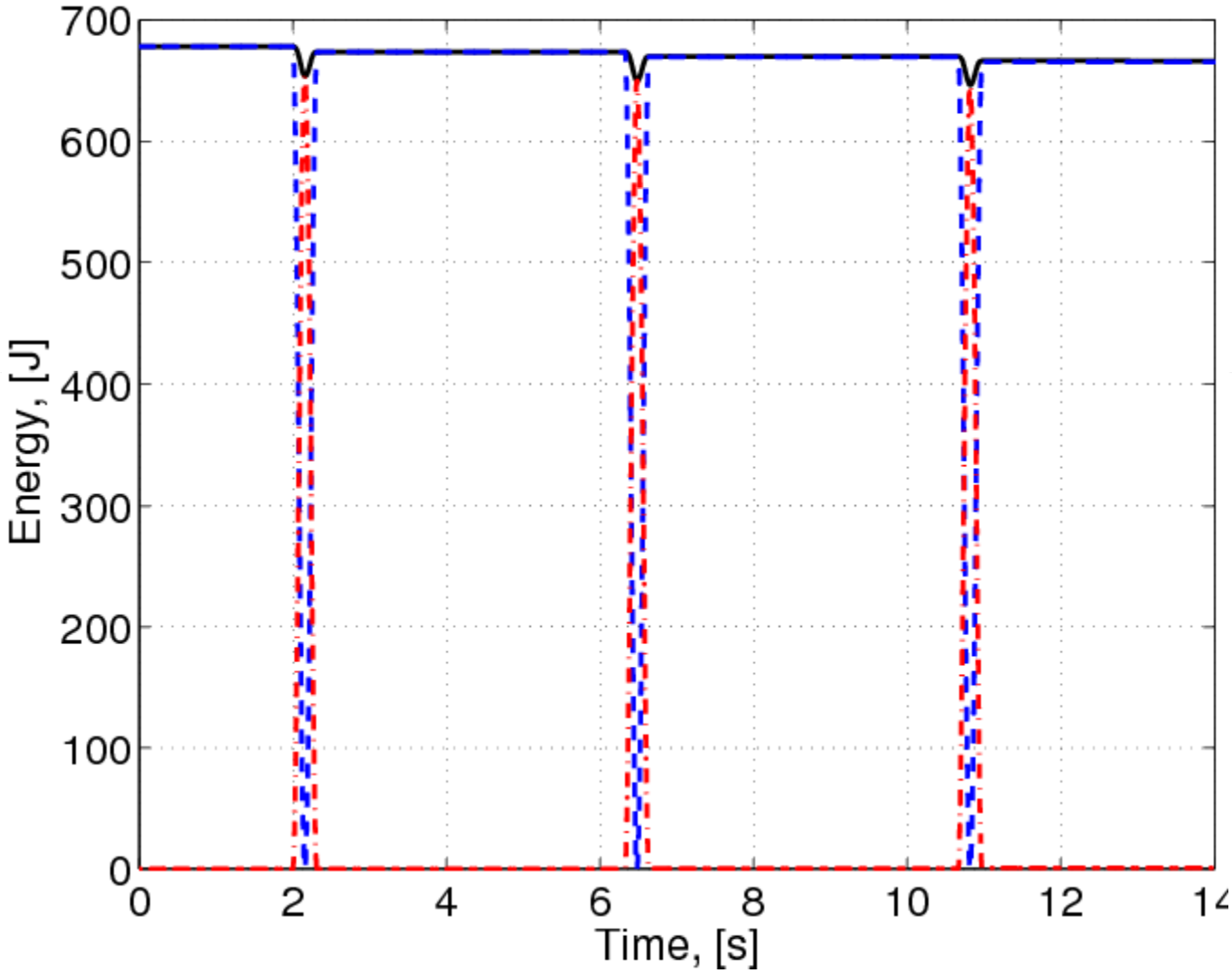



700
600
500
400
300
200
100
0
Energy, [J]
0
2
4
6
8
10
12
14
Time, [s]



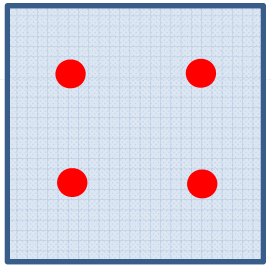

700
600
500
400
300
200
100
0
Energy, [J]
0
2
4
6
8
10
12
14
Time, [s]

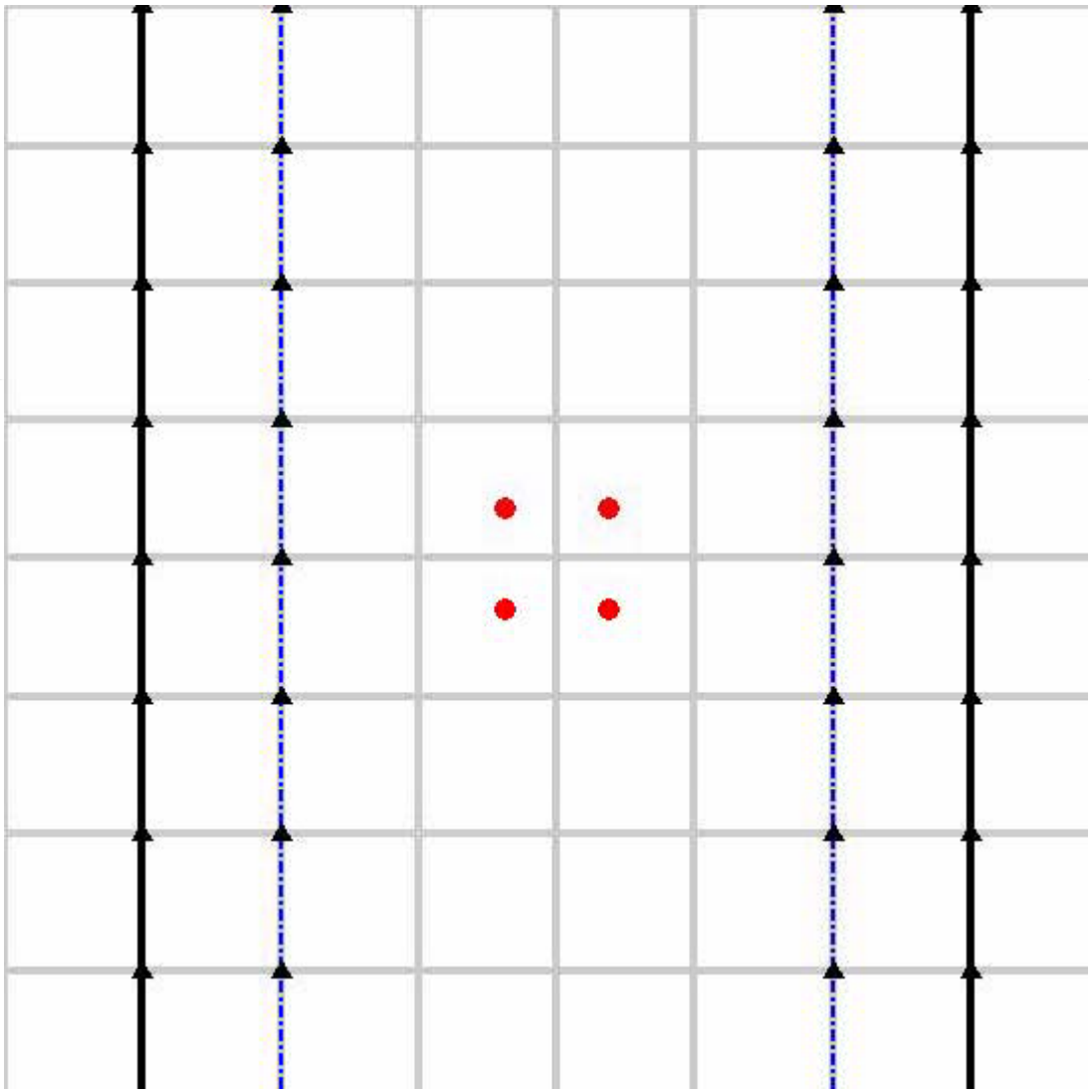


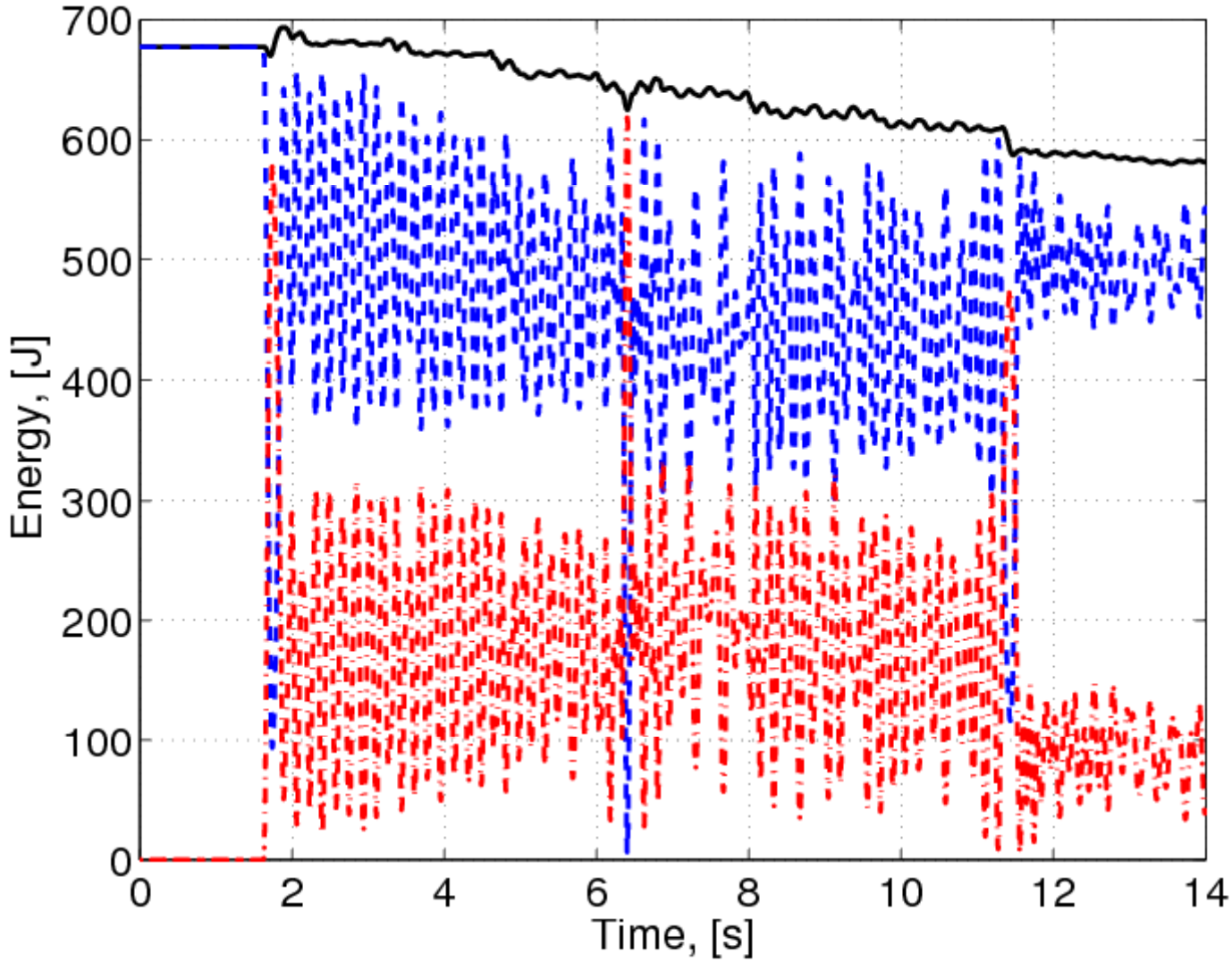

700
600
500
400
300
200
100
0
Energy, [J]
0
2
4
6
8
10
12
14
Time, [s]

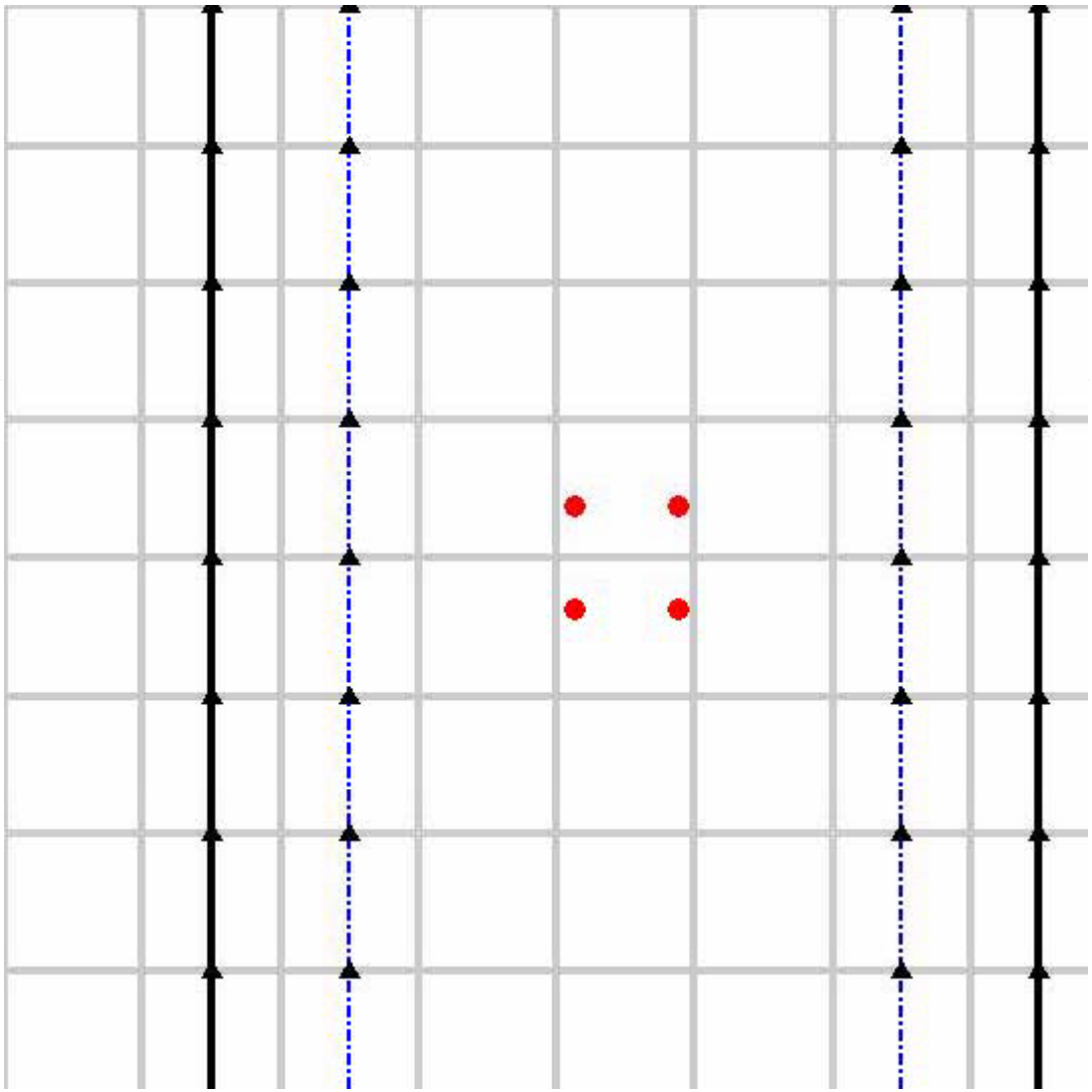
Implementation

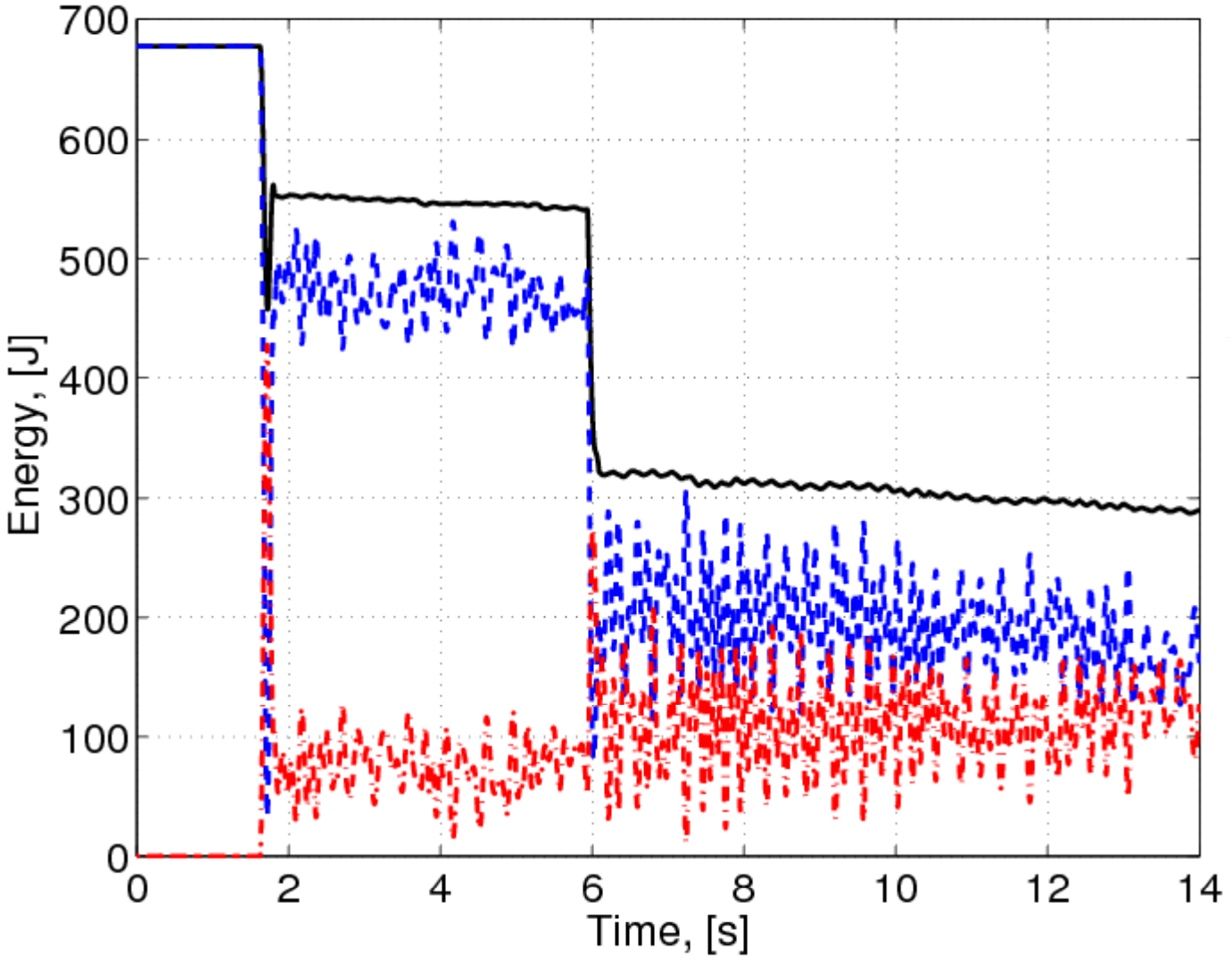
- Single particle analysis
- Discretization A

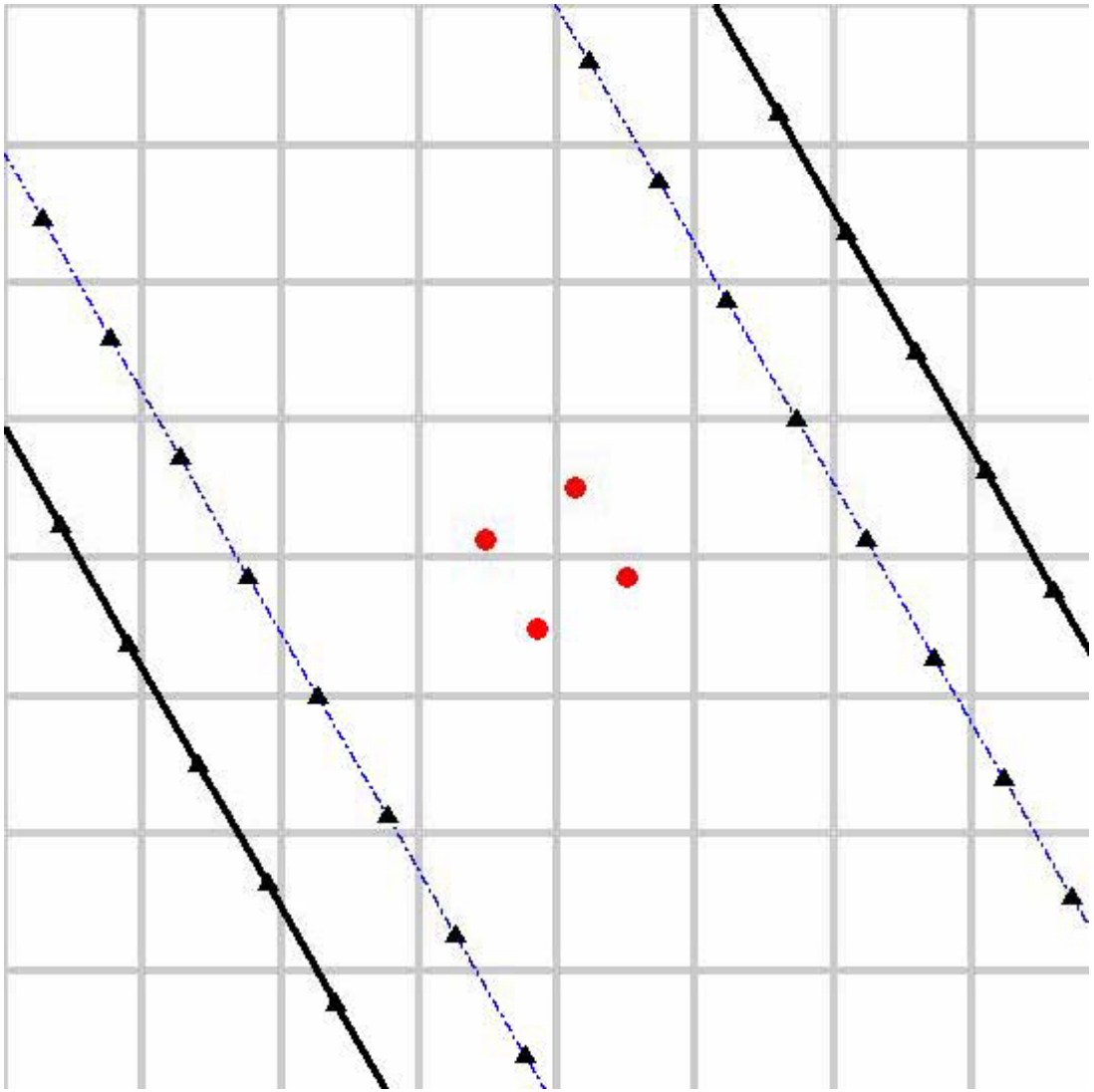


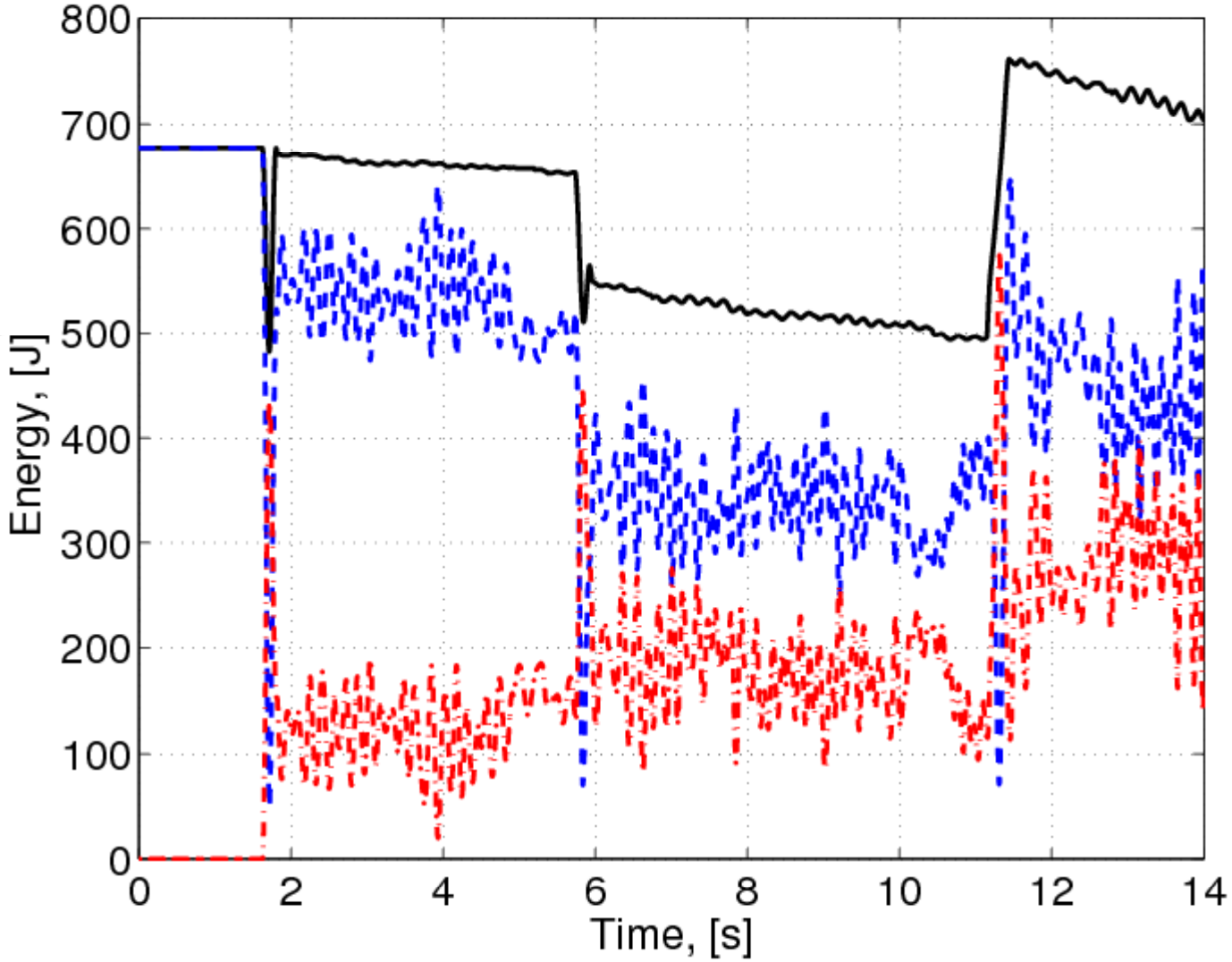



700
600
500
400
300
200
100
0
Energy, [J]
0
2
4
6
8
10
12
14
Time, [s]



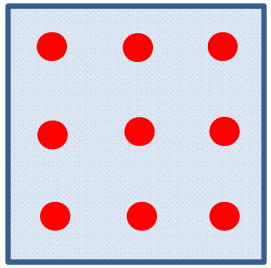

Energy, [J]
Time, [s]
0
100
200
300
400
500
600
700
0
2
4
6
8
10
12
14

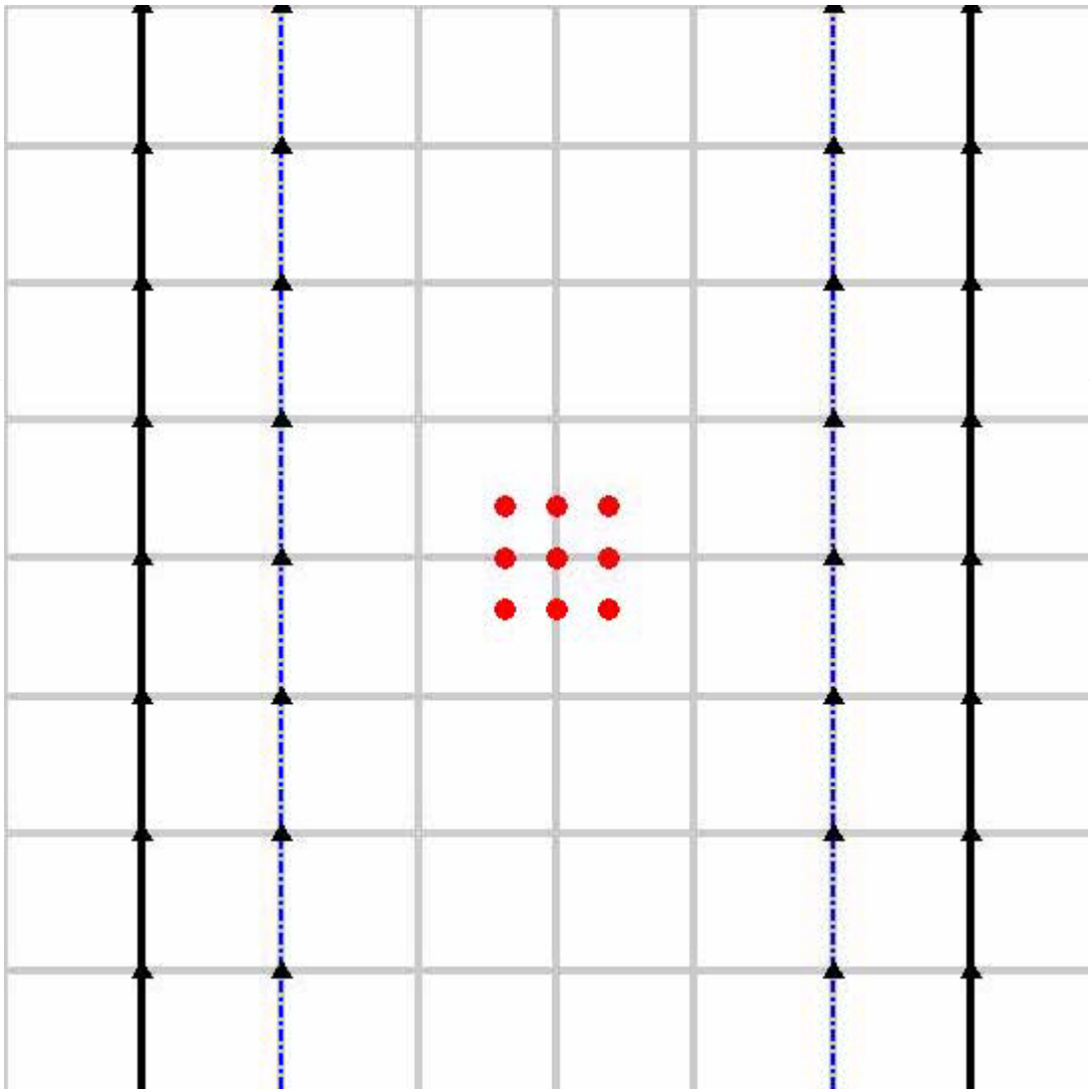


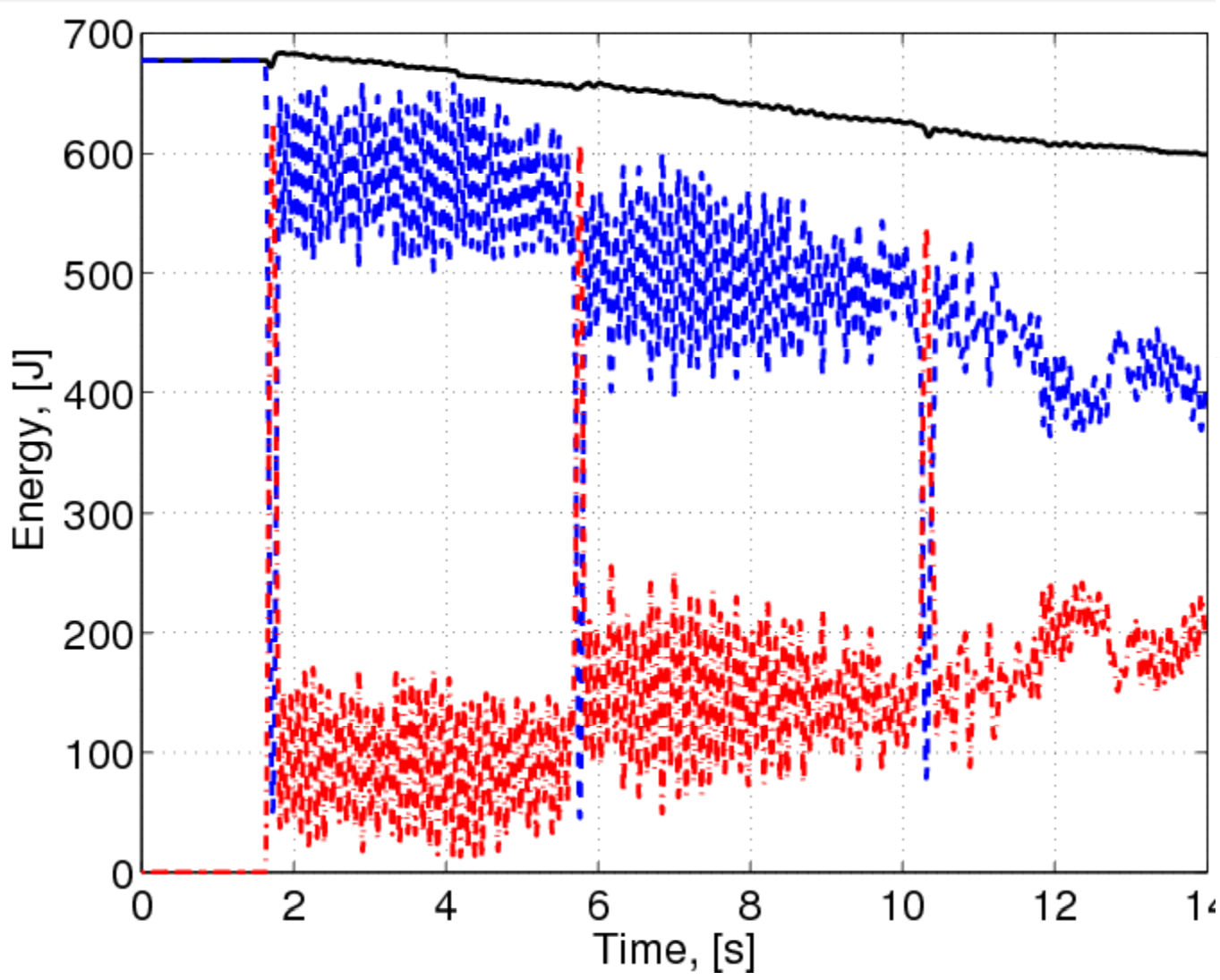

800
700
600
500
400
300
200
100
0
Energy, [J]
0
2
4
6
8
10
12
14
Time, [s]

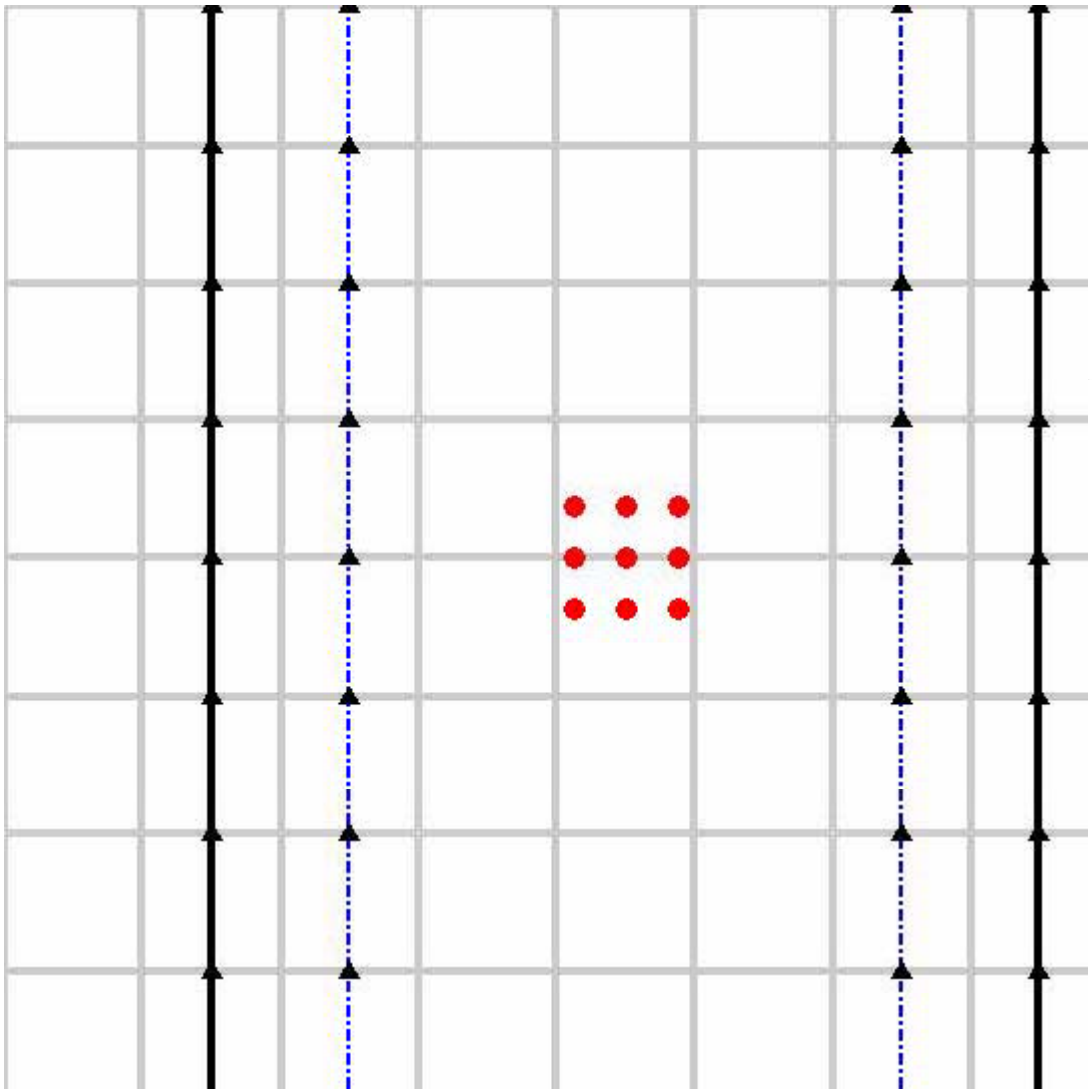
Implementation

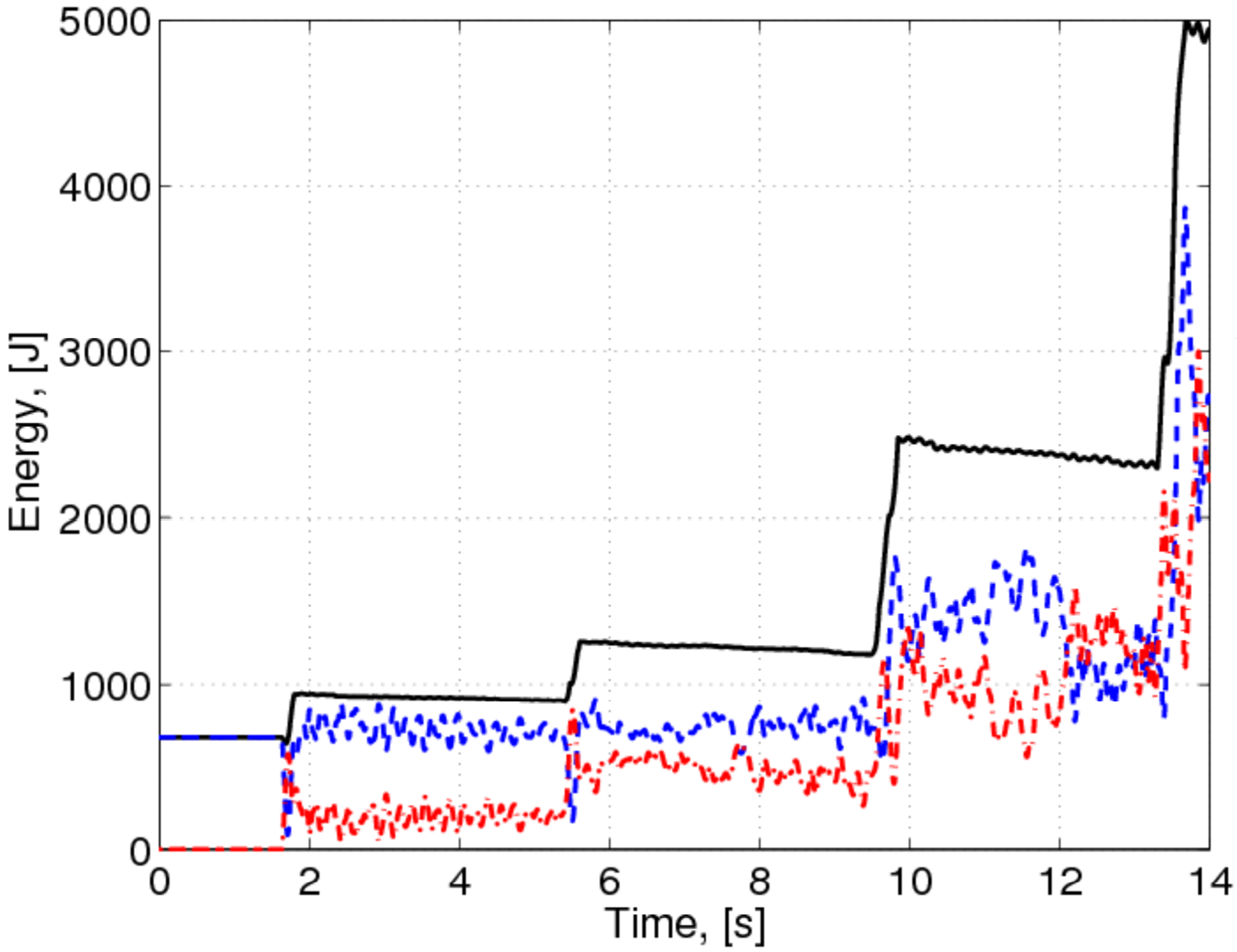
- Single particle analysis
- Discretization A
- Discretization B

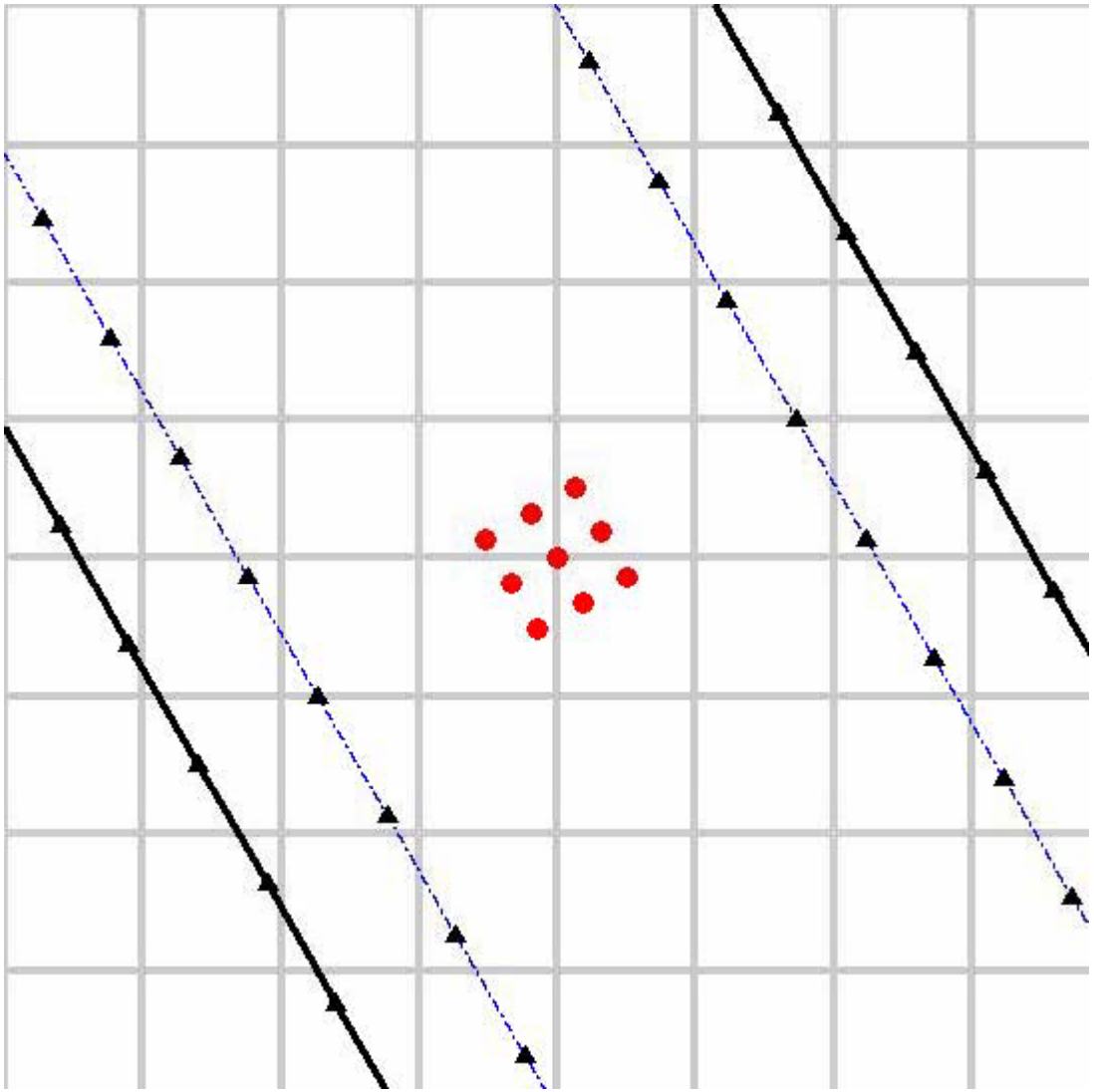


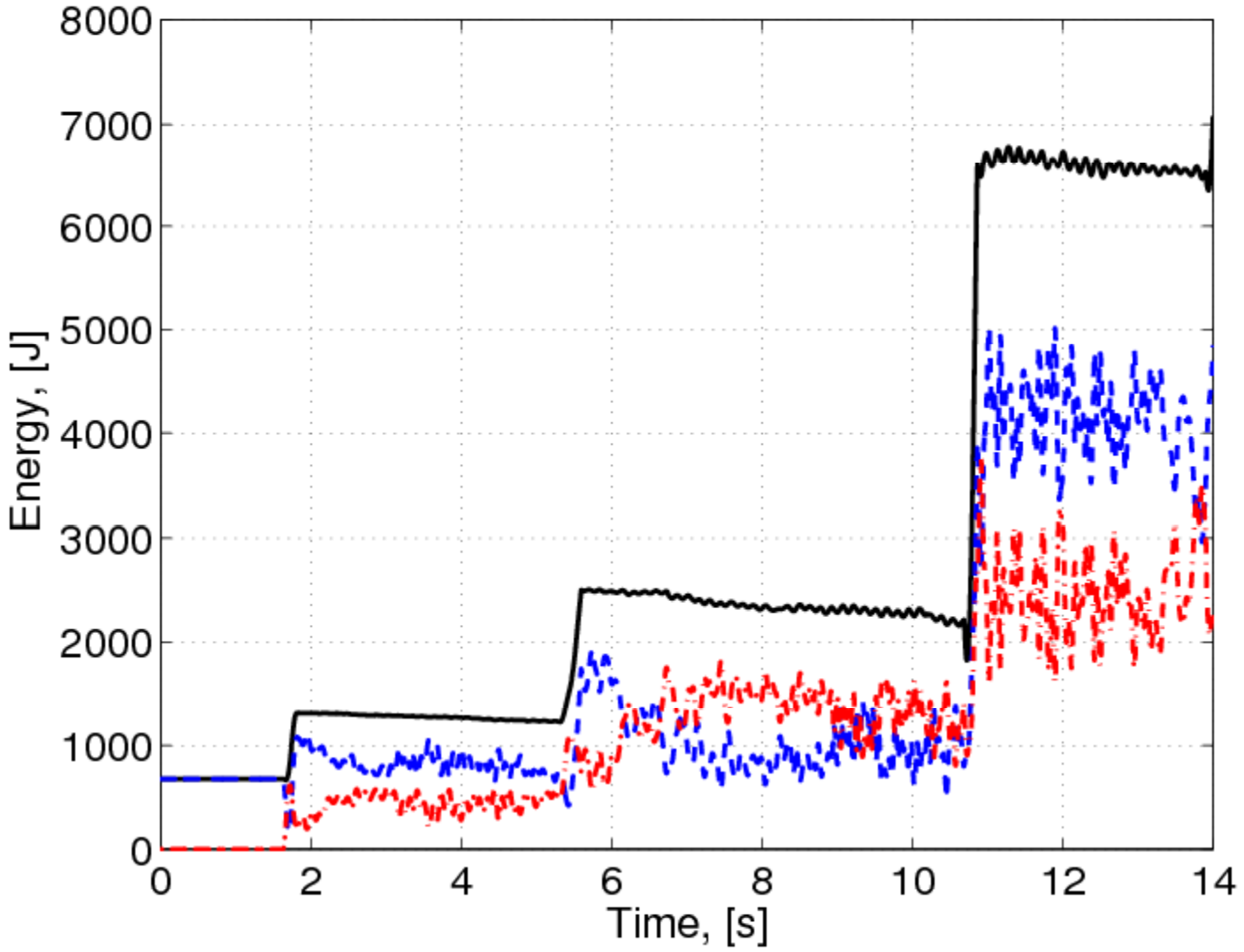



700
600
500
400
300
200
100
0
Energy, [J]
0
2
4
6
8
10
12
Time, [s]




5000
4000
3000
2000
1000
0
Energy, [J]
0
2
4
6
8
10
12
14
Time, [s]




8000
7000
6000
5000
4000
3000
2000
1000
0
Energy, [J]
0
2
4
6
8
10
12
14
Time, [s]

Outlook

- Successfully models all boundary locations/orientations for the single particle
- The standard MPM results are recovered
- Very little consistency from one boundary location to another
  - Body or grid discretization error?
  - Formulation error?
  - Coding error?

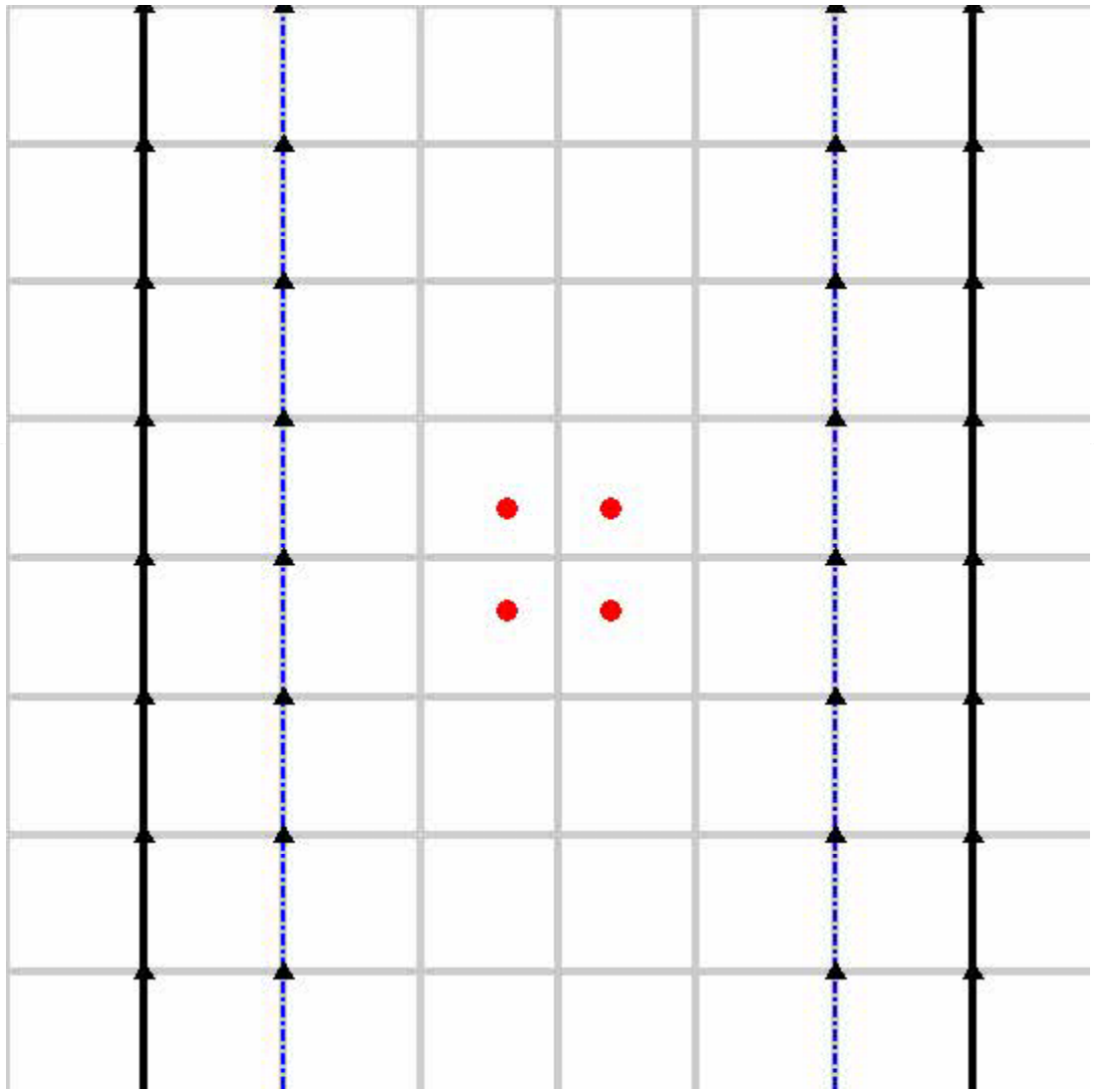
Future Work

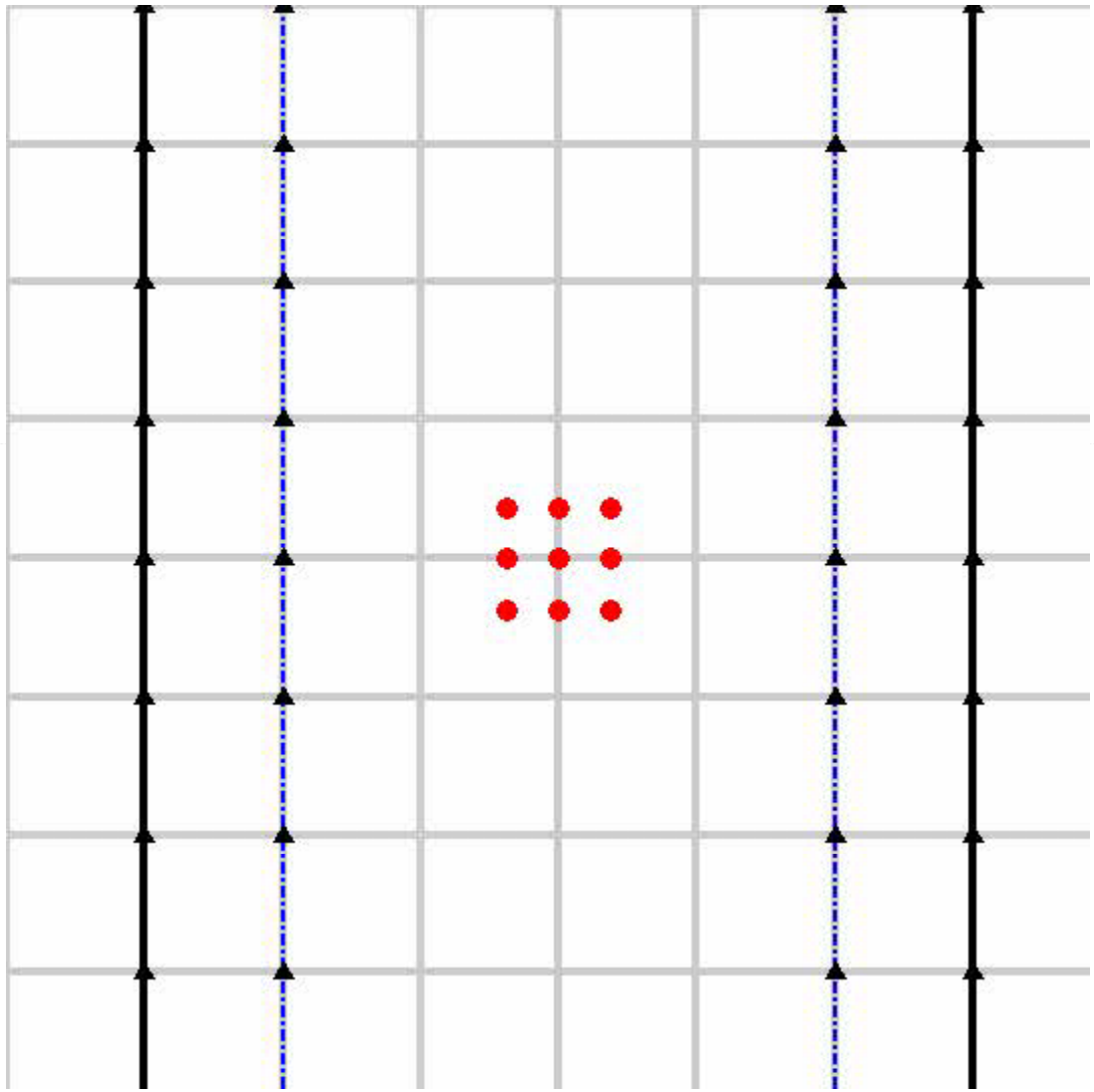
- Continue to try and find out why/what is causing the inconsistency.
- Implement the alternative dual-grid approach in 2- and 3-d.
- Explore alternative methods for incorporating an arbitrary boundary geometry into the Material Point Method.

Thank you!

- All workshop participants
- P.I.'s Peter Mackenzie-Helnwein, Pedro Arduino, and Greg Miller.
- *The National Science Foundation* grant CMMI-0900318

QUESTIONS
????





Approach

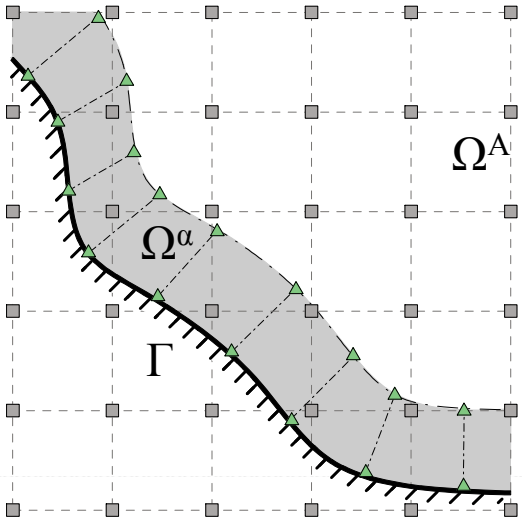
- Dual-Grid methodology
  - *The Enhanced Velocity Field Approach*
    - The total velocity and acceleration field exists as a superposition from both grids

$$\boldsymbol{v}(\boldsymbol{x},t) \approx \boldsymbol{v}^h(\boldsymbol{x},t) := \boldsymbol{v}^A(\boldsymbol{x},t) + \boldsymbol{v}^\alpha(\boldsymbol{x},t)$$

$$\dot{\boldsymbol{v}}(\boldsymbol{x},t) \approx \dot{\boldsymbol{v}}^h(\boldsymbol{x},t) := \dot{\boldsymbol{v}}^A(\boldsymbol{x},t) + \dot{\boldsymbol{v}}^\alpha(\boldsymbol{x},t)$$

    - With the conditions $\boldsymbol{v}^\alpha(\boldsymbol{x},t) = \dot{\boldsymbol{v}}^\alpha(\boldsymbol{x},t) = \boldsymbol{0}$ for $\boldsymbol{x} \in \Omega^A$

Approach



- *The Enhanced Velocity Field Approach*
  - Enforce essential condition along Γ

$$\tilde{\boldsymbol{v}}(\boldsymbol{x},t) = \boldsymbol{v}^A(\boldsymbol{x},t) + \boldsymbol{v}^\alpha(\boldsymbol{x},t) \quad \text{and} \quad \dot{\tilde{\boldsymbol{v}}}(\boldsymbol{x},t) = \dot{\boldsymbol{v}}^A(\boldsymbol{x},t) + \dot{\boldsymbol{v}}^\alpha(\boldsymbol{x},t) \quad \forall\, \boldsymbol{x} \in \Gamma$$

  - Leads to a constraint of the form

$$\int_\Gamma (\boldsymbol{v}^h - \widetilde{\boldsymbol{v}}) \cdot \boldsymbol{\lambda}\, d\Gamma = 0 \quad \Longrightarrow \quad \boldsymbol{\lambda}(\boldsymbol{x},t) \approx \boldsymbol{\lambda}^h(\boldsymbol{x},t) := \sum_\mu S_\mu(\boldsymbol{x})\, \boldsymbol{\lambda}_\mu(t) \quad \forall\, \boldsymbol{x} \in \Gamma$$

Approach

- *The Enhanced Velocity Field Approach*
  - Algorithmic implementation:
    1. Obtain the nodal acceleration at time t_n for those nodes in the boundary grid as well as the standard grid by solving the system.

$$\begin{bmatrix} m^A_{IJ}\mathbf{1} & m^{A,\alpha}_{IS}\mathbf{1} & R^A_{I\mu}\mathbf{1} \\ m^{\alpha,A}_{QJ}\mathbf{1} & m^{\alpha}_{QS}\mathbf{1} & R^{\alpha}_{Q\mu}\mathbf{1} \\ R^A_{J\kappa}\mathbf{1} & R^{\alpha}_{S\kappa}\mathbf{1} & \mathbf{0} \end{bmatrix} \begin{Bmatrix} \dot{\boldsymbol{v}}^A_J \\ \dot{\boldsymbol{v}}^{\alpha}_S \\ \dot{\boldsymbol{\lambda}}_{\mu} \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{f}^{A,ext}_I + \boldsymbol{f}^{A,\sigma}_I \\ \boldsymbol{f}^{\alpha,ext}_Q + \boldsymbol{f}^{\alpha,\sigma}_Q \\ \dot{\hat{\tilde{\boldsymbol{v}}}}_{\kappa} \end{Bmatrix}$$

$$R^A_{J\kappa} = \int_\Gamma N^A_J(\boldsymbol{x}) S_\kappa(\boldsymbol{x})\, d\Gamma \quad \text{and} \quad R^{\alpha}_{S\kappa} = \int_\Gamma N^{\alpha}_S(\boldsymbol{x}) S_\kappa(\boldsymbol{x})\, d\Gamma$$

$$m^{A,\alpha}_{IS} = \sum_p N^A_I(\boldsymbol{x}_p)\, N^{\alpha}_S(\boldsymbol{x}_p)\, m_p \quad \text{and} \quad m^{\alpha,A}_{QJ} = \sum_p N^{\alpha}_Q(\boldsymbol{x}_p)\, N^A_J(\boldsymbol{x}_p)\, m_p$$

Approach

- *The Enhanced Velocity Field Approach*
  - Algorithmic implementation:
    2. Obtain the nodal velocity at time t_n for those nodes in the boundary grid as well as the standard grid by solving the system.

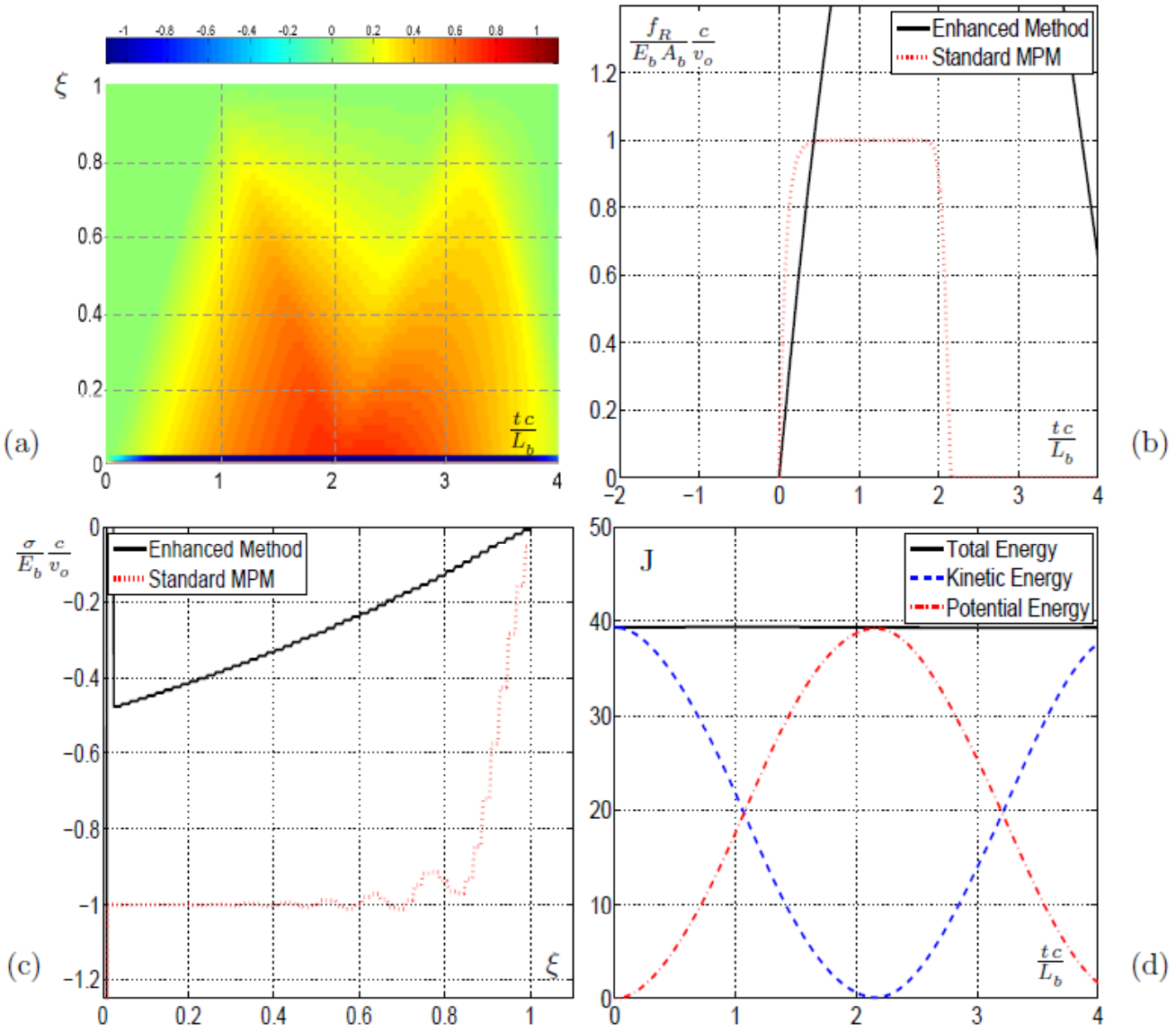
$$\begin{bmatrix} m^A_{IJ}\mathbf{1} & m^{A,\alpha}_{IS}\mathbf{1} & R^A_{I\mu}\mathbf{1} \\ m^{\alpha,A}_{QJ}\mathbf{1} & m^{\alpha}_{QS}\mathbf{1} & R^{\alpha}_{Q\mu}\mathbf{1} \\ R^A_{J\kappa}\mathbf{1} & R^{\alpha}_{S\kappa}\mathbf{1} & 0 \end{bmatrix} \begin{Bmatrix} \boldsymbol{v}^A_{J,n} \\ \boldsymbol{v}^{\alpha}_{S,n} \\ \boldsymbol{\lambda}_{\mu} \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{p}^A_I \\ \boldsymbol{p}^{\alpha}_Q \\ \hat{\hat{\boldsymbol{v}}}_{\kappa} \end{Bmatrix}$$

    3. Update nodal values for both grids.

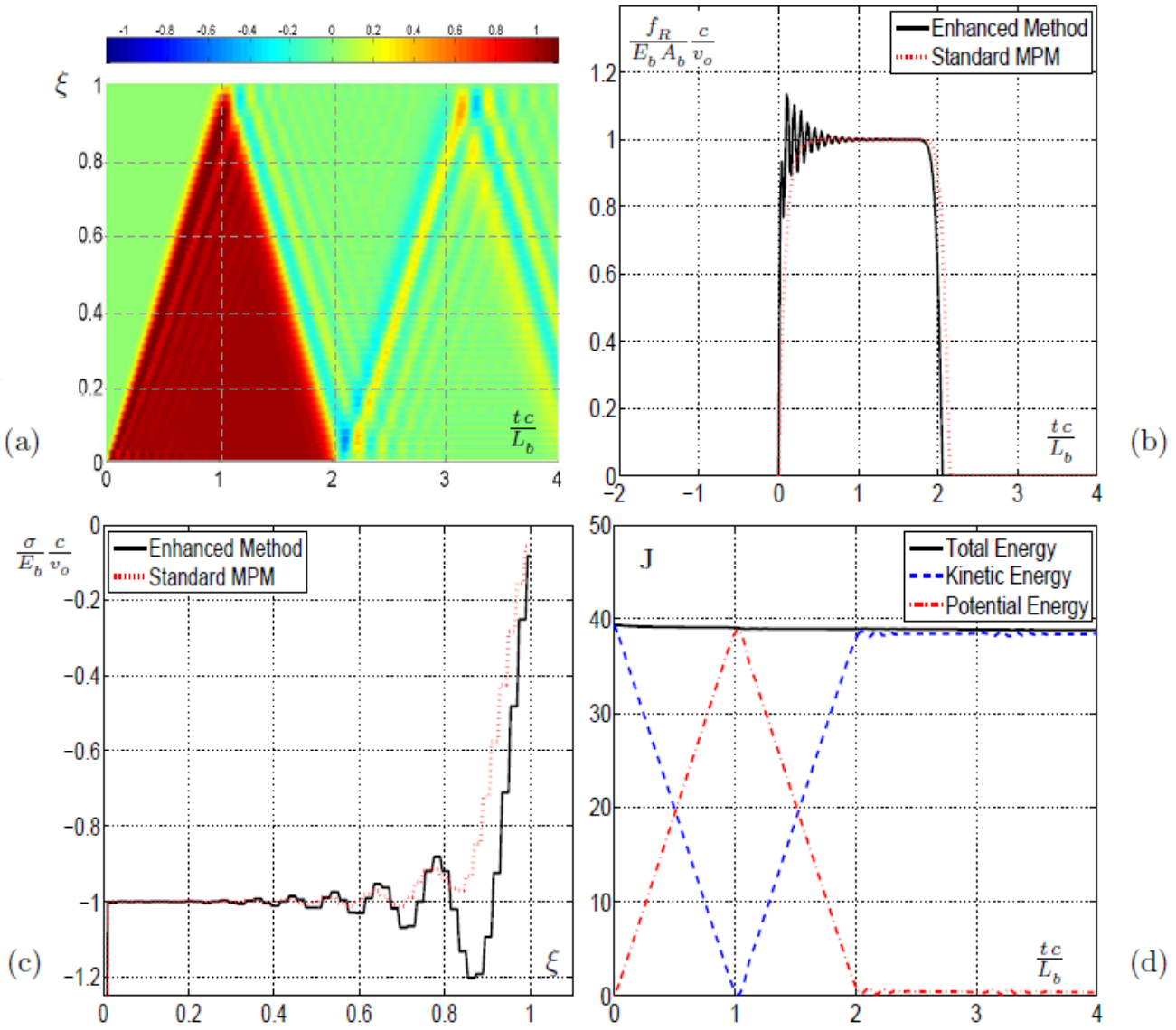
Approach

- *The Enhanced Velocity Field Approach*
  - Algorithmic implementation:
    4. Update all particles using the nodes on the standard grid.
    5. For those particles wit $x_p \in \Omega^\alpha$, perform an additional update using those nodes in the boundary grid.

Implementation



Implementation



Implementation