FINAL MANUSCRIPT SUBMITTED TO THE *JOURNAL OF APPLIED MECHANICS* ON APRIL 20, 2010

TRIANGULAR CELLULAR AUTOMATA FOR COMPUTING TWO-DIMENSIONAL ELASTODYNAMIC RESPONSE ON ARBITRARY DOMAINS

Ryan K. Hopman[1] **Michael J. Leamy**[2]
George W. Woodruff School of Mechanical Engineering
Georgia Institute of Technology
Atlanta, GA, USA 30332-0405

Abstract

This study extends a recently-developed cellular automata (CA) modeling approach [1] to arbitrary two-dimensional geometries via development of a rule set governing triangular automata (cells). As in the previous rectangular CA method, each cell represents a state machine which updates in a stepped-manner using a local, "bottom-up" rule set and state input from neighboring cells. Notably, the approach avoids the need to develop and solve partial differential equations and the complexity therein. The elastodynamic response of several general geometries and loading cases (interior, Neumann and Dirichlet) are computed with the method and then compared to results generated using the earlier rectangular CA and finite element approaches. Favorable results are reported in all cases, with numerical experiments indicating that the extended CA method avoids, importantly, spurious oscillations at the front of sharp wave fronts.

[1]Graduate Student Research Assistant

[2]Corresponding Author and Assistant Professor, 771 Ferst Drive NW, Atlanta, GA 30332-0405. 1-404-385-2828. Michael.Leamy@me.gatech.edu

1. Introduction

Numerical methods for computing wave propagation in elastic solids present challenges due to difficulties associated with accurate and stable tracking of discontinuous wave fronts. In seismic problems involving regular domains, the staggered grid variant of the finite difference time domain (FDTD) approach is the *de facto* standard due to several characteristics: 1) the field variables are represented by what amounts to a discontinuous discretization, 2) boundary conditions can be formulated in a straight-forward manner, and 3) problems can be efficiently parallelized [2]. However, the finite difference approach does not adequately accommodate arbitrarily-shaped domains, and thus in the analysis of engineering structures, the finite element (FE) method has been more commonly applied to study wave propagation in computational solid mechanics (CSM).

In a recent paper, a cellular automata (CA) approach has been presented for modeling wave propagation in elastic media [1]. The approach shares an idea central to all cellular automata modeling, which is domain discretization using uniform cells (usually rectangular or hexagonal) whose state is updated via simple rules. The rules operate on state information collected from local, neighbor cells. The elastodynamic CA approach is discontinuous like the finite difference method, and thus accurately captures propagating wave fronts. In fact, for a uniform rectangular grid, the method reproduces the same interior update equations as the central difference FDTD method [1]; however, it differs from finite difference in that it identifies a single boundary treatment due to the presence of a stress state on each face of the cell. Furthermore, the CA approach has the potential to compute with cells of varying shapes, to include triangular cells. Using a triangular rule set, the CA approach could employ cell assembly to compute on arbitrary geometries, much like the finite element method, while conceivably retaining the advantages of the FDTD method in resolving wave fronts.

With the aforementioned motivation established, this research sets out to extend the CA approach to compute on arbitrary two-dimensional geometries using triangular cells. The extension requires: 1) formulating a rule set governing triangular cells of arbitrary shape and orientation and 2) implementing an object-oriented simulation

approach for cell assembly and state calculations. Results from the new simulator are compared to earlier CA results [1], and to results generated using the commercial finite element package, COMSOL[3]. A detailed analysis of the method (e.g., quantifying stability, accuracy, and convergence) is not presented, and is instead considered appropriate, and necessary, future work.

1.1 An Overview of the Cellular Automata Method

The cellular automata method begins by dividing a domain of interest into discrete automata (or cells) holding states which evolve in time via simple rules based on neighbor interactions. CA is well-established for studying system dynamics in a diverse variety of disciplines. The first CA simulations proposed and implemented are often credited to John von Neumann [4]. An oft-cited example is John Conway's Game of Life [5], in which each cell in the domain perpetuates or dies depending on the state of its eight neighbors. Although von Neumann imagined a deterministic rule set, probabilistic rule sets are also possible and widely implemented [6].

CA models are distinguished by their use of simple, local interaction rules to compute complex global behavior – this global behavior is often termed 'emergent.' The rules are derived from known relationships (such as geometrical constraints, conservation of energy, etc.) and statistical relationships/models, as well as other intuitive relationships. Examples include rules quantifying the rate at which HIV infection destroys T-cells [7], the spatial rate of spread of brushfires [8], and vehicle interactions in traffic models [9]. Physics-based applications include thermal modeling [10], mechanics of carbon nanotubes [11], and electromagnetic wave propagation [12]. Other common applications of CA modeling include grain growth prediction in metal recrystallization [13-15] and computational fluid dynamics (CFD) via Lattice Boltzmann techniques [16, 17]. The latter two techniques have been combined with finite element modeling to simulate complex interactions such as blood flowing through the heart [17] and stress-dependent grain boundary growth [13, 18]. As evidenced in part by these (diverse) applications, the CA approach is notable for its flexibility, compatibility with parallel solution strategies [19-21], and its ability to handle singularities, discontinuities, and inhomogeneities [22-25]. Furthermore, CA computational cost scales linearly with respect to the number of

cells employed [11, 26]. A limitation of CA can be the challenge associated with identifying the proper governing rule set.

Due to difficulty in formulating rule sets, there has been limited use of CA in computational structural mechanics. Most formulations rely on a uniform grid, usually through square or hexagonal cells [27]. Those employing non-uniform grids have typically made a connection with the finite element method [10]. Some cellular automata-inspired CSM models have relied on traditional solution techniques, such as FD [28, 29] and FE [30]. These studies have shown the potential to use the cellular automata paradigm to achieve multi-resolution [28] and multi-scale simulation [31]. Unlike the goal of this study, the rule sets in these previous studies have been derived from a numerical approximation of the global equations [10, 20, 30, 32] in a top-down manner using traditional discretization techniques [FE, FD, and finite volume (FV)].

This work presents a novel CA-based triangular cell class, together with specialized boundary cells, without reusing, or reinterpreting, previous analysis techniques (FD, FE, or FV). As is customary in other methods, triangles are used to represent arbitrarily-shaped two-dimensional domains (see Fig. 1 for an example). Each triangular cell is treated as an autonomous state machine storing pointers to its local neighbors' state, avoiding the need for global, or centralized, control. This lends itself well to object-oriented (OO) programming practices implemented using modern computing languages (*C++*, *Java*, etc.) and is compatible with parallel processing. The order in which cells update is unimportant, furthering the compatibility with parallel computing and multi-resolution analysis.

1.2 Triangular Automata for Modeling Arbitrary Geometry

In this section, the previous CA approach [1] is extended to triangular cells via development of an appropriate rule set. Fig. 2 demonstrates that, contrary to the earlier effort using rectangular cells, triangular cells introduce additional complexity. First, each triangle has a unique set of face lengths and angles. In addition, a line connecting two cell centroids is not necessarily normal to the face that the cells share. This complicates the

evaluation of the normal and shear strains. Finally, the triangular tessellation may produce many neighbors sharing a single vertex, where as in the rectangular case this was limited to three. The importance of this latter consideration becomes evident when attempting to evaluate the Type I and Type II strain components previously identified in the rectangular approach [1].

An overview of the triangular elastodynamic CA method is detailed next. State variables stored by each triangular automata include the displacement, velocity, and applied tractions. In addition, cells store parameters such as material properties (ρ, μ, λ) and cell geometry (centroid, face angles and lengths, area). All state information and applicable parameters reference a global x-y coordinate system. The displacements and applied tractions are determined at each step of the simulation and updated based on a rule set (to be detailed) using the previous state of each automaton and that of its von Neumann and Moore neighbors. As illustrated in Fig. 2, von Neumann neighbors share an edge with the cell of interest, while Moore neighbors share a vertex. It is important to note that not all Moore neighbors are used in the state update proposed herein - only those that are also von Neumann neighbors of the cell's von Neumann neighbor. In what follows, these will be referred to as 'secondary' von Neumann neighbors.

The cells are treated as autonomous entities, and so it suffices to describe the update of any given cell, which we will term the 'target' cell. At the start of each time step, the previous state is transformed into tangential and normal components for each of the target cell's three faces. Strains are obtained by numerically estimating derivatives of displacements across each face by using the target cell and its local neighbors. Similar to the previous work on rectangular cells, these derivatives are classified as either Type I or Type II, as described below. Using Hooke's law, the stresses on the face are then calculated and stored, which, when multiplied by the area of the face, yield forces that can be transformed back to the underlying x-y coordinate system to determine the overall forces on the cells. After new stresses for each face have been obtained, application of Newton's 2nd Law leads to first-order, semi-discrete differential equations governing the cell's velocity and

displacement change. A suitable explicit temporal discretization of these equations yields the final rule set. The target cell's updated state is stored until all cells have also been updated in what amounts to a double buffering technique suitable for parallelization.

2. Development

Development of the triangular automata elastodynamic rule set begins with consideration of stresses acting on each face of a triangular cell present in the simulation domain. In the linearly-elastic case considered herein, these stresses arise from strain components identified as either Type I or Type II [1]. To facilitate strain computations, an angle θ is identified which measures the angle between the x-axis and the face normal ($-\pi \leq \theta \geq \pi$), as shown in Fig. 3. The direction normal to the face is identified with the unit vector $\boldsymbol{e_n}$, while the direction tangent to the face is identified with $\boldsymbol{e_t}$. Unlike in the development of mechanics using an infinitesimal element, the strains and stresses on each face are non-correlated – e.g., two faces separated by ninety degrees do not have the same shear. However, in order to satisfy equilibrium, the shear and normal stress on the face of one cell must be the same as for the neighboring cell sharing the same face.

Strain calculations across faces use displacements in the $\boldsymbol{e_n}$ and $\boldsymbol{e_t}$ directions, denoted by u_n and u_t, which can be obtained from the cell's state by a simple rotation transformation:

$$\begin{bmatrix} u_n \\ u_t \end{bmatrix} = R \begin{bmatrix} u_x \\ u_y \end{bmatrix}, \quad \text{where,} \quad R = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix}. \tag{1}$$

The strains can then be calculated starting from their definitions,

$$\varepsilon_{nn} = \left.\frac{\partial u_n}{\partial n}\right|_\theta, \varepsilon_{tt} = \left.\frac{\partial u_t}{\partial s}\right|_\theta, \varepsilon_{nt} = \left(\left.\frac{\partial u_t}{\partial n}\right|_\theta + \left.\frac{\partial u_n}{\partial s}\right|_\theta \right). \tag{2}$$

Evaluation of these expressions requires numerical differentiation in either the normal (∂n) or tangential (∂s) directions. As in the previous work [1], the derivatives are defined as Type I (across the face) and Type II (parallel to the face) – see Fig. 4. Type I models tensile strain and (what is termed herein) 'direct shear,'

whereas Type II models Poisson effects and 'indirect shear.' Note that only von Neumann neighbors are required to evaluate Type I derivatives, while Type II require inclusion of Moore neighbors which have been pre-identified as secondary von Neumann (see Sec. 1). Type II calculations are made by combining and averaging four calculations, as shown in Fig. 5.

To facilitate the numerical differentiation, additional notation specific to triangular automata is introduced next. In all cases, a superscript will identify the location of a cell while a subscript will denote a direction (n: normal, t: tangential, +: positive theta, –: negative theta). Fig. 5 depicts the neighbor labeling from the perspective of an example face on a target cell (T); other cells identified include von Neumann (N: sharing the target face, N^+: sharing the target cell's positive-theta face, N^-: sharing the target cell's negative-theta face) and Moore (M^+: secondary von Neumann sharing the target cell's positive-theta vertex, M^-: secondary von Neumann sharing the target cell's negative-theta vertex) neighbors. Denoting as Δn and Δs the distance between cell centroids in the normal and tangential directions, respectively, a discrete rule set can now be written where the Type I derivatives are approximated as,

$$\left.\frac{\partial u_t}{\partial n}\right|_\theta \Rightarrow \frac{u_t^N - u_t^T}{\Delta n}; \qquad \text{similarly,} \qquad \left.\frac{\partial u_n}{\partial n}\right|_\theta \Rightarrow \frac{u_n^N - u_n^T}{\Delta n}. \tag{3a, 3b}$$

Similar to the rectangular case, Type II derivatives are determined by calculations spanning differences across both the target cell and the primary and secondary von Neumann neighbors (see Fig. 4 for a physical rationale). These calculations are then averaged to find the difference in the tangential direction at the face. The Type II derivatives are therefore approximated as (e.g.),

$$\left.\frac{\partial u_t}{\partial s}\right|_\theta \Rightarrow \frac{1}{2}\left(\frac{\left(u_t^{N+} - u_t^T\right) - \left(u_t^{N-} - u_t^T\right)}{\Delta s_+^T + \Delta s_-^T} + \frac{\left(u_t^{M+} - u_t^N\right) - \left(u_t^{M-} - u_t^N\right)}{\Delta s_+^N + \Delta s_-^N}\right), \tag{4}$$

which simplifies to,

$$\left.\frac{\partial u_t}{\partial s}\right|_\theta \Rightarrow \frac{1}{2}\left(\frac{u_t^{N+}-u_t^{N-}}{\Delta s_+^T+\Delta s_-^T}+\frac{u_t^{M+}-u_t^{M-}}{\Delta s_+^N+\Delta s_-^N}\right). \tag{5}$$

Substituting the Type I and Type II derivatives into the strain relationships yields:

$$\varepsilon_{nn}=\frac{u_n^N-u_n^T}{\Delta n}, \tag{6a}$$

$$\varepsilon_{tt}=\frac{1}{2}\left(\frac{u_t^{N+}-u_t^{N-}}{\Delta s_+^T+\Delta s_-^T}+\frac{u_t^{M+}-u_t^{M-}}{\Delta s_+^N+\Delta s_-^N}\right), \tag{6b}$$

$$\varepsilon_{nt}=\left(\frac{u_t^N-u_t^T}{\Delta n}+\frac{1}{2}\left(\frac{u_n^{N+}-u_n^{N-}}{\Delta s_+^T+\Delta s_-^T}+\frac{u_n^{M+}-u_n^{M-}}{\Delta s_+^N+\Delta s_-^N}\right)\right). \tag{6c}$$

The stresses σ_{nn}, σ_{tt}, and σ_{nt} follow from Hooke's law:

$$\sigma_{nn}=(\lambda+2\mu)\varepsilon_{nn}+\lambda\varepsilon_{tt} \tag{7a}$$

$$\sigma_{nt}=\mu\varepsilon_{nt} \tag{7b}$$

With stresses computed on each face, forces can next be obtained from the surface area of each face and tallied to find the change in linear momentum of each cell. The two forces present on each face are $F_n=\sigma_{nn}w*l$ and $F_t=\sigma_{nt}w*l$, where w denotes a width in the z-direction and l denotes the face's length:

$$F_n=wl\left((\lambda+2\mu)\frac{u_n^N-u_n^T}{\Delta n}+\frac{\lambda}{2}\left(\frac{u_t^{N+}-u_t^{N-}}{\Delta s_+^T+\Delta s_-^T}+\frac{u_t^{M+}-u_t^{M-}}{\Delta s_+^N+\Delta s_-^N}\right)\right), \tag{8a}$$

$$F_t=wl\mu\left(\frac{u_t^N-u_t^T}{\Delta n}+\frac{1}{2}\left(\frac{u_n^{N+}-u_n^{N-}}{\Delta s_+^T+\Delta s_-^T}+\frac{u_n^{M+}-u_n^{M-}}{\Delta s_+^N+\Delta s_-^N}\right)\right). \tag{8b}$$

The distances between cell centroids in the normal and tangential directions (Δn and Δs) can be established early in the simulation before performing state updates. This is accomplished by reading standard triangular finite element meshes composed of nodal coordinates and connectivity lists, and then determining the geometry of each triangle in the simulation. Requisite information such as centroid locations, face lengths and angles, and cell neighbors follow. In addition, vectors linking centroids yield the normal and tangential distances (see Fig. 6). For Type I derivatives, the vector connecting the two cell centers has a magnitude denoted by r and an angle relative to the x-axis denoted by φ. Thus Δn can be approximated by $r\cos(\varphi-\theta)$. For the Type II derivatives, the various Δs_+ and Δs_- appearing in the denominators of (8) combine to yield approximations $r^N\cos\left(\varphi^N-\theta-\frac{\pi}{2}\right)$ or $r^T\cos\left(\varphi^T-\theta-\frac{\pi}{2}\right)$:

$$F_n = wl\left((\lambda+2\mu)\frac{u_n^N-u_n^T}{r\cos(\phi-\theta)}+\frac{\lambda}{2}\left(\frac{u_t^{N+}-u_t^{N-}}{r^T\cos\left(\phi^T-\theta-\pi/2\right)}+\frac{u_t^{M+}-u_t^{M-}}{r^N\cos\left(\phi^N-\theta-\pi/2\right)}\right)\right) \tag{9a}$$

$$F_t = wl\mu\left(\frac{u_t^N-u_t^T}{r\cos(\phi-\theta)}+\frac{1}{2}\left(\frac{u_n^{N+}-u_n^{N-}}{r^T\cos\left(\phi^T-\theta-\pi/2\right)}+\frac{u_n^{M+}-u_n^{M-}}{r^N\cos\left(\phi^N-\theta-\pi/2\right)}\right)\right) \tag{9b}$$

Since the deformation of the cells is negligible for small strains, no adjustment to the normal and tangential displacements is required during state updates. Finally, the forces acting on the example face in the x- and y-directions are obtained by the reverse of the earlier rotation transformation, yielding

$$\begin{bmatrix} F_x \\ F_y \end{bmatrix} = R^T \begin{bmatrix} F_n \\ F_t \end{bmatrix}. \tag{10}$$

Following computation of the forces on each, these internal forces, together with external body loading, are used in the balance of momentum,

$$\sum \mathbf{F} = \sum \mathbf{F}_{internal} + \sum \mathbf{F}_{external} = m\frac{d\mathbf{v}}{dt}, \tag{11}$$

where bold face denotes a vector, m denotes the cell's mass and **v** denotes the cell's velocity. Using a forward-Euler approximation to the time derivative, the velocity update equations yield the rules governing the velocity state update,

$$v_x^{k+1} = v_x^k + \frac{1}{d\rho wA}\sum F_x\,, \tag{12a}$$

$$v_y^{k+1} = v_y^k + \frac{1}{d\rho wA}\sum F_y\,, \tag{12b}$$

where d is the discrete number of steps per unit time, ρ is the density, w is the thickness of the domain, and A is the area of a cell. The superscript k represents the k^{th} time iteration. Displacements can be obtained from the updated velocities using a second first-order approximation, which completes the requisite rule set,

$$u_x^{k+1} = u_x^k + \frac{1}{d}v_x^{k+1}, \quad \text{and,} \quad u_y^{k+1} = u_y^k + \frac{1}{d}v_y^{k+1}. \tag{13a, 13b}$$

It is noted, but detailed elsewhere [33], that the rule set obtained in the present context, when applied to four orthogonal faces, recovers exactly that of the previous rectangular CA. Note further that the temporal discretization employed is one of many possible choices – the method is not limited to the one presented.

3. Implementation

The rule set described by (9-13) has been implemented as a stand-alone *Java* code in a fully object-oriented manner typical of a CA architecture. Pseudo-code for the simulation is included in Fig. 7. The stand-alone simulator makes use of finite element meshes for geometry generation and cell discretization. The simulator also allows for specification of Dirichlet and Neumann boundary conditions, as detailed below.

The simulation begins with a *setup* method establishing the triangular automata. A list of automata can be loaded from a saved configuration or, as indicated in Fig. 7, a text file containing nodes and connectivity generated by a commercial meshing package. For the case of a mesh file, the element and node information determines each cell's geometry, immediate neighbors, and centroid distances r. The simulator employs this

information to establish, for each triangular automaton, an array of pointers with access to its neighbors' state and *r* values. These automata are objects instantiated from the same triangular class. A simulation consists of the automata objects and an accompanying *main* object whose job it is to call upon each automata to update its state at an appropriate increment of time. Since two triangles share a face, approximately half the calculations can be avoided by storing an additional state variable that indicates when one cell has already calculated the new forces on the face. At present, the simulator uses fixed time steps; however, this can be easily modified to include adaptive time stepping. It may also be advantageous in future work to explore automata updates based on the launching of discrete events [34], which has the potential to significantly decrease the computational domain in regions not experiencing wave disturbances (e.g., before and after a wave has passed). Output is stored based on a user-defined number of time steps.

3.1 Boundary Treatment

At setup, any cell lacking a von Neumann neighbor is assigned a boundary cell. The added cell amounts to an equilateral triangle with each side matching the length of the domain cell (Fig. 8); centroids and other parameters are calculated accordingly. To accommodate a variety of boundary conditions, two general types of boundary cells are used: Dirichlet, which specifies displacement, and Neumann, which specifies stress. Both specifications can be time-dependent. Dirichlet cells avoid the use of Eqs. (12-13) and instead update based upon user-prescribed displacement profiles. Additional neighbors are required (shown as dotted outlines in Fig. 8) for Type II calculations performed by cells in the interior; as indicated in Fig. 8, the necessary geometry is calculated, yet the cells need not be instantiated since they share the same state and material properties. The Neumann cells allow for prescribed tractions. Since the stress on the face shared with the interior domain is set by this cell, neighboring cells are not required. Instead, the extra state variable storing whether or not the face has been updated is set to true, and the interior cell sharing the face automatically receives the stress. However, the displacement state of the added Neumann cell is used in a Type II calculation by the non-shared faces of its interior adjoining neighbor. Thus, it requires position updates. The calculation is similar to that outlined in the previous rectangular CA

work [1] where the displacements are chosen to faithfully reproduce the applied tractions via (7a-b); when the notation is modified for the triangular geometry herein, the displacement updates take the form,

$$u_n^{Neumann} = u_n^N - \frac{\sigma_n \Delta n}{(\lambda + 2\mu)} + \frac{\lambda \Delta n}{(\lambda + 2\mu)\left(\Delta s_+^N + \Delta s_-^N\right)}\left(u_t^{M+} - u_t^{M-}\right), \tag{14a}$$

$$u_t^{Neumann} = u_t^N - \frac{\sigma_t \Delta n}{\mu} + \frac{\Delta n}{\Delta s_+^N + \Delta s_-^N}\left(u_n^{M+} - u_n^{M-}\right), \tag{14b}$$

where σ_n and σ_t denote the prescribed normal and tangential tractions, respectively, and the states of von Neumann and Moore neighbors are that at the updated time iteration.

4. Validation and Discussion

This section presents example numerical results useful in validating the presented triangular CA simulator. Accuracy is assessed by comparing generated results with results obtained using the previous rectangular CA simulator [1] and the commercial finite element package COMSOL [3].

4.1 Elastic Half-Space: Comparison to Rectangular CA

The first validation study considers an elastic half-space subjected to a differentiated Gaussian pulse applied on its free surface. This test case is characterized by regular geometry and therefore can be simulated using the previous rectangular CA simulator, which for interior calculations is known to be equivalent to a central-difference variant of the finite difference time domain method [1]. The material properties and loading parameters can be found in the previous study. Uniform triangular cells were created by dividing square cells from the previous study into two equal triangular cells, as depicted in Fig. 9. Also shown are the relevant neighbors of a target cell. The von Neumann neighbors of the (i,j) cell are given by $(i,j+1)$, $(i,j-1)$, and $(i+f(i,j),j)$ where

$$f(i,j) = \left(\frac{\mathrm{i}^j}{\left|\mathrm{i}^j\right|}\right)(-1)^i . \tag{15}$$

For the results presented, loading was divided equally between two triangular cells that made up the corresponding square cell. For the results reported, an array of 200 X 100 square cells were used in the computations.

To facilitate comparison of the results obtained using triangular and rectangular automata, computed states from two triangular cells were averaged to recover equivalent square states. Fig. 10 provides the *x*- and *y*-components of displacement at a snapshot in time computed using both methods. The time shown is sufficient for the pulse to complete its application. The figure documents very good agreement between the two methods. Both predict shear, pressure, and surface waves with magnitudes within 1% of each other. Closer inspection of the displacement components (see inset sub-figures detailing the *y*-components of displacement) reveal that the method employing triangular automata generates results with a slight loss of symmetry. Note that this loss is less-significant than the loss of symmetry encountered when using staggered-grid finite difference schemes – see [1] for examples.

4.2 Non-Uniform Meshes and Arbitrary Geometry: Comparison to FE-Generated Results

The previous results document very good performance using uniform triangular meshes. To look at non-uniform meshes on arbitrarily-shaped domains, a series of comparison cases were analyzed using the COMSOL commercial FEA tool. Table 1 documents the material and loading parameters used in each of the three considered cases: interior harmonic loading, and application of Neumann and Dirichlet boundary conditions. The meshes generated by COMSOL were also used by the CA simulation tool. Although the same meshes are employed by both tools, the storage of state information is very different. Each cellular automaton holds a *unique* state considered to be the field values at the cell's centroid. In contrast, each finite element accesses state information (shared by other elements) at multiple nodal locations. In fact, this lack of state encapsulation prevents FE schemes from being classified as CA. As a further consequence of these differences, the simulations are loaded and evaluated at different points in space. This introduces a minor degree of difficulty in comparing the two methods, as addressed during the discussion below. All results are post-processed using

Matlab scripts which employ the raw data (centroidal or nodal states) and generate plots showing, usually, the raw data as markers on top of Matlab-fitted surface meshes. Note that these surface meshes are not the underlying finite element mesh. Note also that in results where large spatial variation is present, the raw data does not necessarily lay on the generated surface meshes due to the quadratic polynomials used to best-fit the surfaces to the raw data.

Harmonic Interior Load

The first case examines performance of the triangular automata for smooth, interior loading on a simply connected domain. It also assesses the convergence behavior of the method. For all results generated, an interior harmonic load was simulated on a simple rectangular domain (see Fig. 11). In the FE model, the load was applied to a single vertex (i.e., node); in the CA model, the four cells sharing this same vertex were each given one-fourth of the load. Three meshes were employed, referred to herein as coarse, fine, and finer. Each edge of the domain is considered to be fixed. The domain is also considered to be 1 meter thick and composed of aluminum (see Table 1 for the material properties employed). Identical parameters were used in the FE model. The generalized-alpha solver was selected in COMSOL with a high-frequency amplification factor of one, implying no damping of high frequencies (i.e., alpha = 0). Additional solver damping is still present, and in all comparison cases presented, the solver was allowed to choose this in its default, automated mode. The same time step employed in the CA simulations was also employed in the FE simulations.

Fig. 12 documents good agreement between the CA- and FE-generated y-components of displacement at the chosen point in time, 3.5E-4 seconds, for all three meshes. Each sub-figure presents the CA data as dark markers overlaid on the FE-fitted surface. For the coarsest mesh, the CA results predict wave amplitudes a few percent larger than the FE results. The source of this difference is not clear; however, it could be accounted for by the solver damping employed by COMSOL. As the mesh fineness increases,

this difference decreases such that with the finest mesh, no significant difference can be observed between the two methods.

Neumann Boundary Condition

Next the effectiveness of the CA Neumann boundary treatment is assessed by simulating a multiply-connected domain subjected to an applied traction of 10.0 kN/m^2. The geometry and mesh employed are shown in Fig. 1. The mesh consists of 2,704 triangles and is similar in feature size to the fine mesh used in the harmonic interior load case. The Neumann boundary treatment is employed for cells lining the circular hole, the free edges on the top and bottom of the domain, and the left edge where the traction is applied. The right edge is modeled as fixed. Aluminum is again chosen for the material properties.

Fig. 13 presents several plots comparing the CA- and FE-predicted *x*-component of displacement at time 5.0E-4 seconds. This is an adequate amount of time in order for a pressure wave to propagate the length of the domain and reflect back from the right end. The top row provides two views of the CA-generated data, while the middle row provides similar views of the FE-generated data. Note that the FE results include data at the nodes and in the interior of the triangular elements, and hence a much larger number of raw data markers (green) appear in the FE row than in the CA row (dark markers). Side-by-side comparisons of the first two rows reveal that both methods are predicting very similar amplitudes and response shapes. When the CA raw data is superimposed on the FE surface mesh (bottom row, left sub-figure), some amplitude difference is noted between the two methods. This is likely a result of the difference in where the applied traction is represented in each of the two models. In CA, the load is applied, in effect, to additional cells appended to the domain (see Sec. 3.1), thus causing a time-delay for waves to reach the interior domain. Since shear and pressure waves travel at different speeds, and since the traverse (*y*-direction) is unaffected by the boundary treatment, the offset cannot be fully-matched by a simple time shift in the FE results. However, when comparing the CA results to FE results generated at an earlier time (bottom row, right sub-figure), very good agreement between the two methods can, in fact, be observed.

Dirichlet Boundary Condition

The last case examined considers the response due to application of a Dirichlet boundary condition. The same geometry, mesh, and material properties as used in the Neumann study are used in the Dirichlet study. Instead of an applied traction at the left boundary, the present case considers a uniform one meter displacement of the left edge starting at time zero. This challenging case results in the propagation of sharp, broad-band wave-fronts (pressure, shear, etc.) primarily in the *x*-direction.

Fig. 14 presents the results for the *x*-component of displacement, shortly after the application of the load (time 1.3E-4 seconds), computed using the two methods. Both methods exhibit broad frequency content, as expected, with displacement profiles exhibiting similar behavior envelopes. However, significant differences are present. In particular, the FE results exhibit spurious oscillations (also termed Gibbs phenomena) in front of the wave-front. These oscillations are well-known to occur in continuous-Galerkin finite element approaches [35]. It is important to note that they are not an artifact of the time-stepping algorithm employed by COMSOL, but rather a result of the spatial discretization employed – i.e., exact solution of the semi-discrete FE equations will exhibit the same Gibbs phenomena. The CA simulation results, in comparison, do not exhibit these spurious oscillations. Instead, the wave front more-faithfully captures the sharp, nearly-discontinuous wave-front. Although the source of this positive CA behavior is not completely understood at this time, it can be explained in-part by the discontinuous state representation inherent to the CA method – essentially, the CA representation of the state lies in the correct space of discontinuous functions. This positive feature of the CA method may make it attractive for studying the long-term behavior of propagating elastic waves, particularly in scattering problems and nonlinear media where mode and frequency conversion can be erroneously identified if spurious oscillations are present.

5. Concluding Remarks

The elastodynamic CA method has been extended to accurately model arbitrarily-shaped two-dimensional geometries using triangular automata. As in the earlier rectangular approach, a local rule set is developed without the need to first formulate partial differential equations. The resulting formulation fits naturally with object-oriented programming practices, and as such, has been implemented in a stand-alone *Java* simulator. Simulations of the elastodynamic response for a several general geometries and loading cases (interior, Neumann and Dirichlet) yield results in good agreement with those generated using rectangular CA and finite element approaches. Furthermore, numerical experiments indicate that the presented method avoids, importantly, spurious oscillations (i.e., Gibbs phenomena) at the front of sharp wave fronts. The CA's straight-forward formulation, its potential for multi-resolution and parallel simulation, and its notable accuracy may make the presented CA approach an attractive option for simulating elastodynamic response in a wide-array of applications, to include: seismic behavior, acoustic wave propagation, structural wave propagation (e.g., for non-destructive evaluation), blast waves in materials, and performance characterization of devices such as wave guides, phononic crystals, and surface-acoustic wave (SAW) devices.

Acknowledgments

The authors would like to acknowledge support from the Defense Threat Reduction Agency under contract HDTRA1-09-C-0018.

References

1. Leamy, M.J., 2008, "Application of cellular automata modeling to seismic elastodynamics," *International Journal of Solids and Structures*, **45**(17): p. 4835-4849.
2. Schröder, C.T., *On the interaction of elastic waves with buried land mines: an investigation using the finite-difference time-domain method*. 2001, Georgia Institute of Technology.

3. Pryor, R.W., *Multiphysics modeling using COMSOL : a first principles approach*, Boston: Jones and Bartlett Publishers.

4. von Neumann, J., *Theory of Self-reproducing Automata*. 1966, Univ. of Illinois Press: Urbana, IL.

5. Gardner, M., 1970, "Fantastic Combinations of John Conway's New Solitaire Game Life," *Scientific American*, **223**(4): p. 120-123.

6. Raabe, D., 2002, "Cellular automata in materials science with particular reference to recrystallization simulation," *Annual Review of Materials Research*, **32**: p. 53-76.

7. dos Santos, R.M.Z. and S. Coutinho, 2001, "Dynamics of HIV infection: A cellular automata approach," *Physical Review Letters*, **87**(16): p. 168102.

8. Green, D.G., A. Tridgell, and A.M. Gill, 1990, "Interactive Simulation of Bushfires in Heterogeneous Fuels," *Mathematical and Computer Modelling*, **13**(12): p. 57-66.

9. Schreckenberg, M., et al., 1995, "Discrete Stochastic-Models for Traffic Flow," *Physical Review E*, **51**(4): p. 2939-2949.

10. Honma, T. and N. Tosaka, 2003, "Autonomous decentralized finite element method and its applications," *International Journal for Numerical Methods in Engineering*, **57**(6): p. 853-874.

11. Ryoo, J., et al., 2007, "Estimation of Young's modulus of single-walled carbon nanotube using cellular automata," *Advances in Engineering Software*, **38**(8-9): p. 531-537.

12. Simons, N.R.S., et al., 1994, "Cellular-Automata as an Environment for Simulating Electromagnetic Phenomena," *Ieee Microwave and Guided Wave Letters*, **4**(7): p. 247-249.

13. Lan, Y.J., et al., 2005, "Mesoscale simulation of deformed austenite decomposition into ferrite by coupling a cellular automaton method with a crystal plasticity finite element model," *Acta Materialia*, **53**(4): p. 991-1003.

14. Raghavan, S. and S.S. Sahay, 2007, "Modeling the grain growth kinetics by cellular automaton," *Materials Science and Engineering a-Structural Materials Properties Microstructure and Processing*, **445**: p. 203-209.

15. Yang, B.J., L. Chuzhoy, and M.L. Johnson, 2007, "Modeling of reaustenitization of hypoeutectoid steels with cellular automaton method," *Computational Materials Science*, **41**(2): p. 186-194.

16. Bernsdorf, J., F. Durst, and M. Schafer, 1999, "Comparison of cellular automata and finite volume techniques for simulation of incompressible flows in complex geometries," *International Journal for Numerical Methods in Fluids*, **29**(3): p. 251-264.

17. Krafczyk, M., et al., 2001, "Two-dimensional simulation of fluid-structure interaction using lattice-Boltzmann methods," *Computers & Structures*, **79**(22-25): p. 2031-2037.

18. Das, S., et al., 2007, "A combined neuro fuzzy-cellular automata based material model for finite element simulation of plane strain compression," *Computational Materials Science*, **40**(3): p. 366-375.

19. Rothman, D.H., 1987, "Modeling Seismic P-Waves with Cellular Automata," *Geophysical Research Letters*, **14**(1): p. 17-20.

20. Hajela, P. and B. Kim, 2001, "On the use of energy minimization for CA based analysis in elasticity," *Structural and Multidisciplinary Optimization*, **23**(1): p. 24-33.

21. Slotta, D.J., et al., 2002, "Convergence analysis for cellular automata applied to truss design," *Engineering Computations*, **19**(7-8): p. 953-969.

22. Eugenio, A. and M. Rasetti, 1996, "A cellular automaton for elasticity equations," *International Journal of Modern Physics B*, **10**(2): p. 203-218.

23. Abdellaoui, M., A. El Jai, and M. Shillor, 2002, "Cellular automata model for a contact problem," *Mathematical and Computer Modelling*, **36**(9-10): p. 1099-1114.

24. Zhong, Y.M., et al., 2006, "A cellular neural network methodology for deformable object simulation," *Ieee Transactions on Information Technology in Biomedicine*, **10**(4): p. 749-762.

25. Psakhie, S.G., et al., 2001, "Movable cellular automata method for simulating materials with mesostructure," *Theoretical and Applied Fracture Mechanics*, **37**(1-3): p. 311-334.

26. Rappaz, M. and C.A. Gandin, 1993, "Probabilistic Modeling of Microstructure Formation in Solidification Processes," *Acta Metallurgica Et Materialia*, **41**(2): p. 345-360.

27. Fabero, J.C., A. Bautista, and L. Casasus, 2001, "An explicit finite differences scheme over hexagonal tessellation," *Applied Mathematics Letters*, **14**(5): p. 593-598.

28. Psakhie, S.G., et al., 2004, "Modeling the behavior of complex media by jointly using discrete and continuum approaches," *Technical Physics Letters*, **30**(9): p. 712-714.

29. Hirsekorn, M., et al., 2006, "Elastic wave propagation in locally resonant sonic material: Comparison between local interaction simulation approach and modal analysis," *Journal of Applied Physics*, **99**(12): p. 124912.

30. Kwon, Y.W. and S. Hosoglu, 2008, "Application of lattice Boltzmann method, finite element method, and cellular automata and their coupling to wave propagation problems," *Computers & Structures*, **86**(7-8): p. 663-670.

31. Zheng, C.W., et al., 2008, "Microstructure prediction of the austenite recrystallization during multi-pass steel strip hot rolling: A cellular automaton modeling," *Computational Materials Science*, **44**(2): p. 507-514.

32. Delsanto, P.P., et al., 1994, "Connection machine simulation of ultrasonic wave propagation in materials. II: The two-dimensional case," *Wave Motion*, **20**: p. 295-303.

33. Hopman, R., *Arbitrary Geometry Cellular Automata for Elastodynamics*. 2008, Georgia Institute of Technology: Atlanta.

34. Fujimoto, R.M., 1990, "Parallel Discrete Event Simulation," *Communications of the Acm*, **33**(10): p. 30-53.

35. Idesman, A., et al., 2009, "Benchmark problems for wave propagation in elastic materials," *Computational Mechanics*, **43**(6): p. 797-814.

Figure and Table Captions

Table 1: Loading parameters used in generating simulation results.

Figure 1: Non-uniform triangular mesh of a two-dimensional domain with an included hole. Geometry and mesh used in the studies of the Neumann and Dirichlet boundary conditions (see Figs. 13-14).

Figure 2: Illustration of geometrical differences between rectangular automata and triangular automata.

Figure 3: Angle θ, relative to the global x-axis, for each face determined by a line originating at the centroid and perpendicular to the face of interest. From this angle the rotation tranformation is determined for the normal and tangent directions.

Figure 4: Identification of Type I and Type II strain components (rectangular automata depicted).

Figure 5: Illustration of the Type I and Type II calculations for both rectangular and triangular automata.

Figure 6: Illustration of the parameters used to obtain Δn and Δs from centroid and face angle information.

Figure 7: Psuedo-code depicting the fully object-oriented simulation approach used to update the automata states.

Figure 8: Method of applying a boundary cell. Any cell that fails a neighbor check gets a cell added to it with a parameter set reflecting that it is either displacement-imposed (Dirichlet) or traction-imposed (Neumann).

Figure 9: Geometry of the uniform triangular mesh used in the elastic half-space study.

Figure 10: Comparison of displacement results computed for an elastic half-space using rectangular automata (top) and triangular automata (bottom): x-component (left) and y-component (right). The loading consists of a differentiated Gaussian pulse centered on the free surface.

Figure 11: Triangular meshes used in the interior harmonic loading study: (a) coarse – 976 triangles, (b) fine – 3904 triangles, (c) finer – 15,616 triangles.

Figure 12: Results at time 5.0E-4 seconds for the y-component of displacement generated using CA (dark markers) and FE (surfaces) for the case of a harmonic interior load. Top to bottom: increasing mesh fineness.

Figure 13: Results at time 5.0E-4 seconds resulting from a Neumann boundary condition. Shown are the *x*-components of displacement: two perspectives for the CA simulation (top row), FE (center row), and CA markers on top of FE surfaces (bottom row). The bottom row compares the CA results (markers) with the FE results (surfaces) at the same instant of time (left sub-figure) and when the FE results plotted are 2.0E-5 seconds earlier than the CA results (right sub-figure).

Figure 14: Results at time 1.3E-4 seconds resulting from a Dirichlet boundary condition. Shown are the *x*-components of displacement: results displayed using isometric and *x*-*z* views for CA (top row) and FE (bottom row).

Table 2: Loading parameters used in generating simulation results.

Parameter	***Interior Load***	***Neumann Boundary Condition***	***Dirichlet Boundary Condition***
E	70.0 GPa	70.0 GPa	70.0 GPa
ν	0.33	0.33	0.33
ρ	2710.0 kg/m^3	2710.0 kg/m^3	2710.0 kg/m^3
Load Location	(0.0, -2.5) m	x = -2.2 m	x = -2.2 m
Load Amplitude	1.0 kN	10.0 kN/m^2	1.0 m
Load Frequency	3.0 kHz	*N/A*	*N/A*
CA Time Step	1.0E-6 sec	1.0E-6 sec	1.0E-6 sec
Time at Comparison	3.5E-4 sec	5.0E-4 sec	1.3E-4 sec

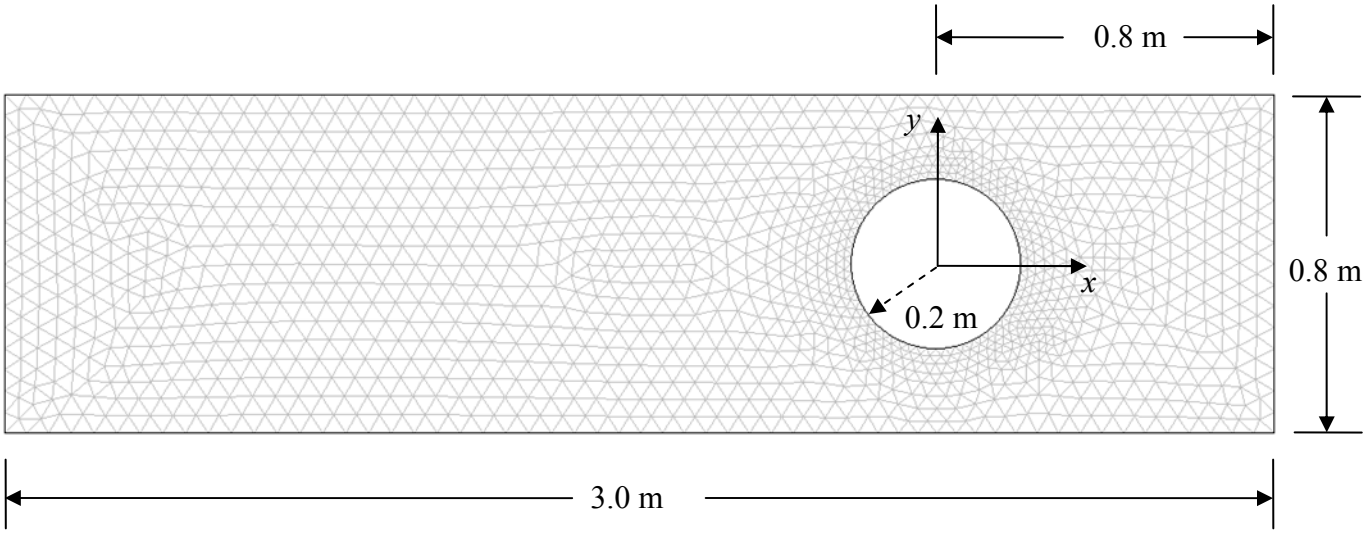


Figure 5: Non-uniform triangular mesh of a two-dimensional domain with an included hole. Geometry and mesh used in the studies of the Neumann and Dirichlet boundary conditions (see Figs. 13-14).

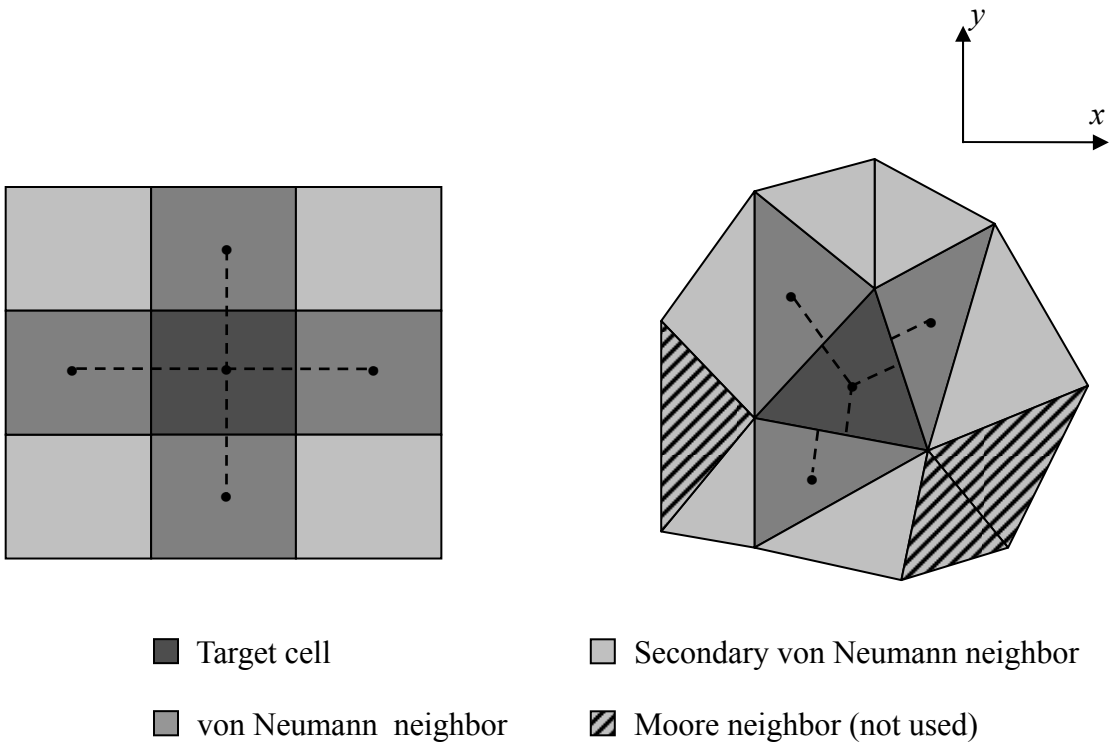


Figure 6: Illustration of geometrical differences between rectangular automata and triangular automata.

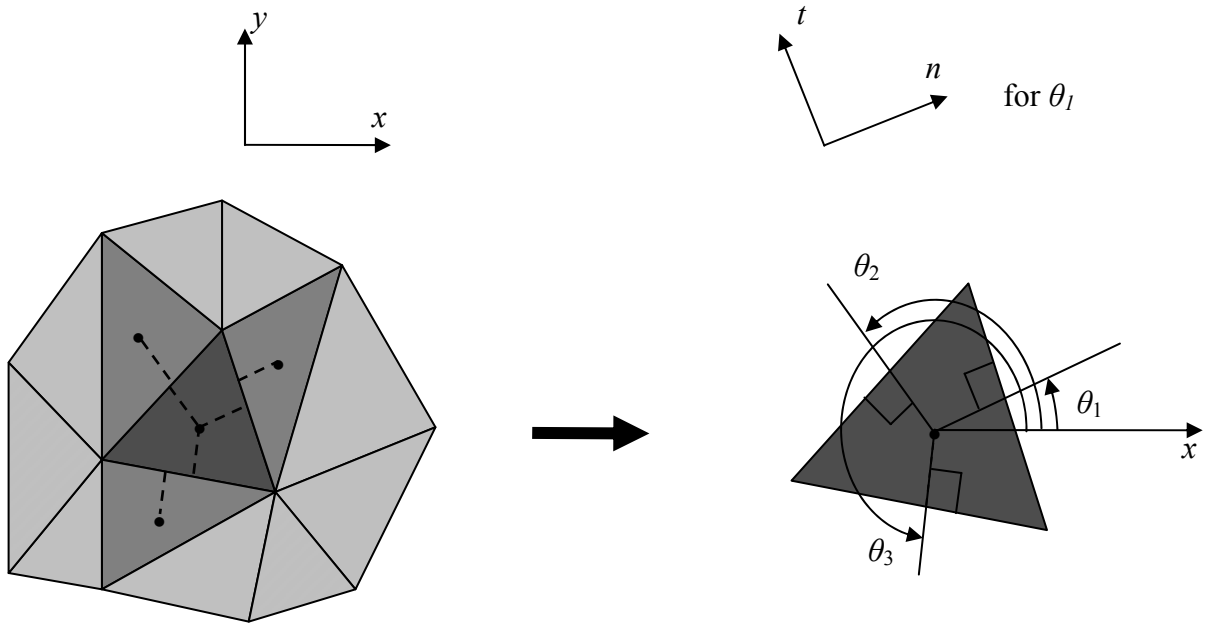


Figure 7: Angle θ, relative to the global x-axis, for each face determined by a line originating at the centroid and perpendicular to the face of interest. From this angle the rotation tranformation is determined for the normal and tangent directions.

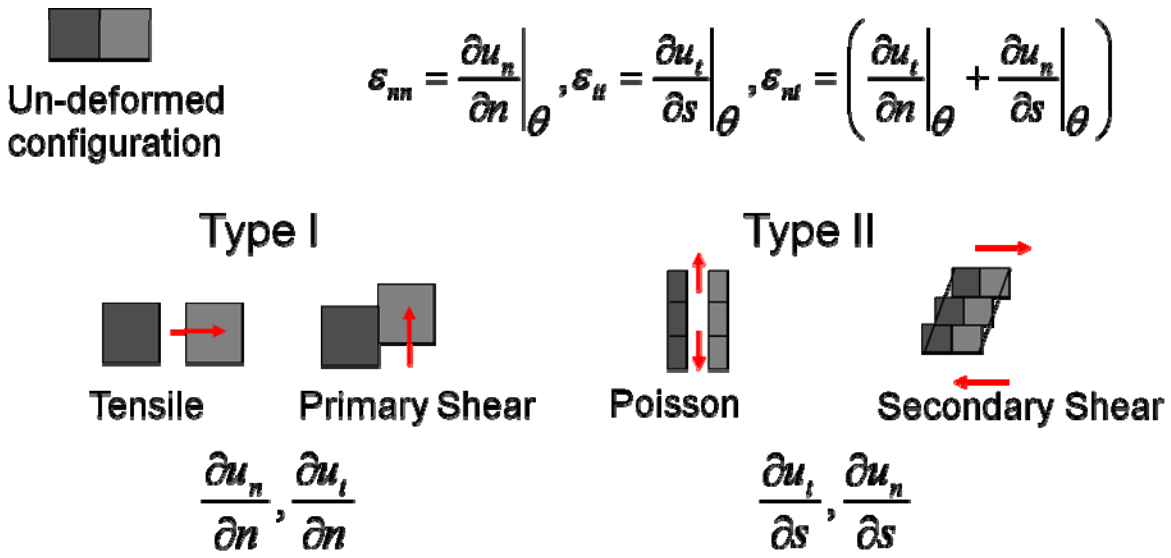
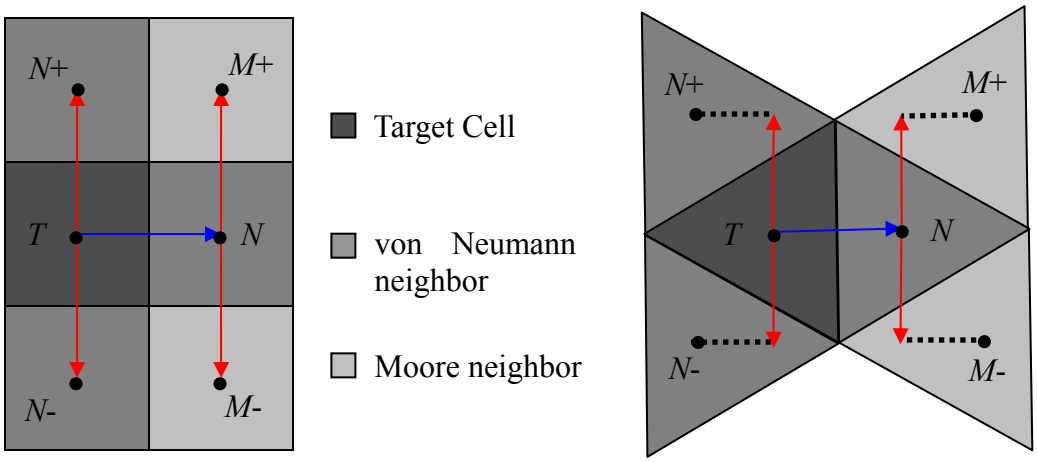


Figure 8: Identification of Type I and Type II strain components (rectangular automata depicted).



Type I (neighbor – target)

Type II (average of the difference across neighbor and difference across target)

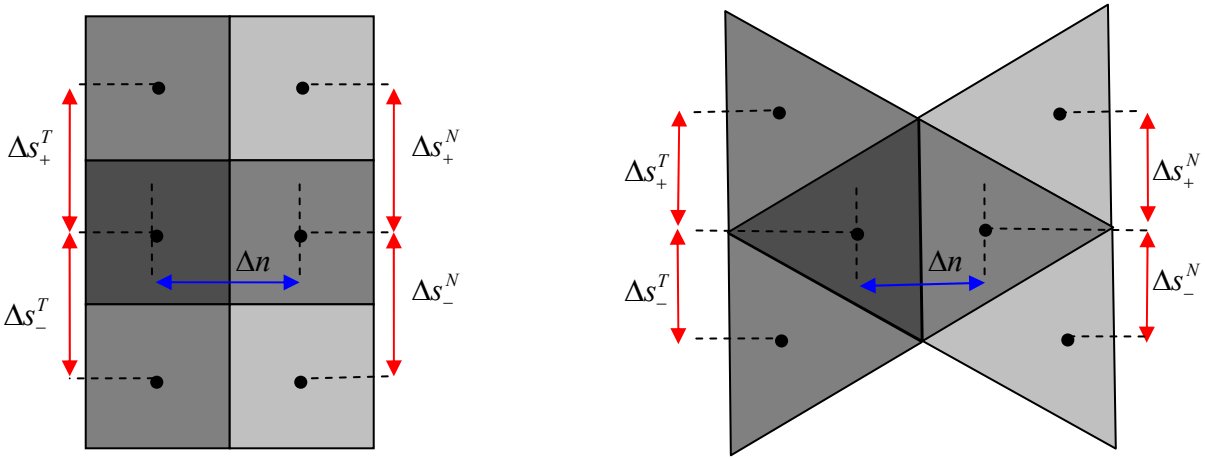


Figure 5: Illustration of the Type I and Type II calculations for both rectangular and triangular automata.

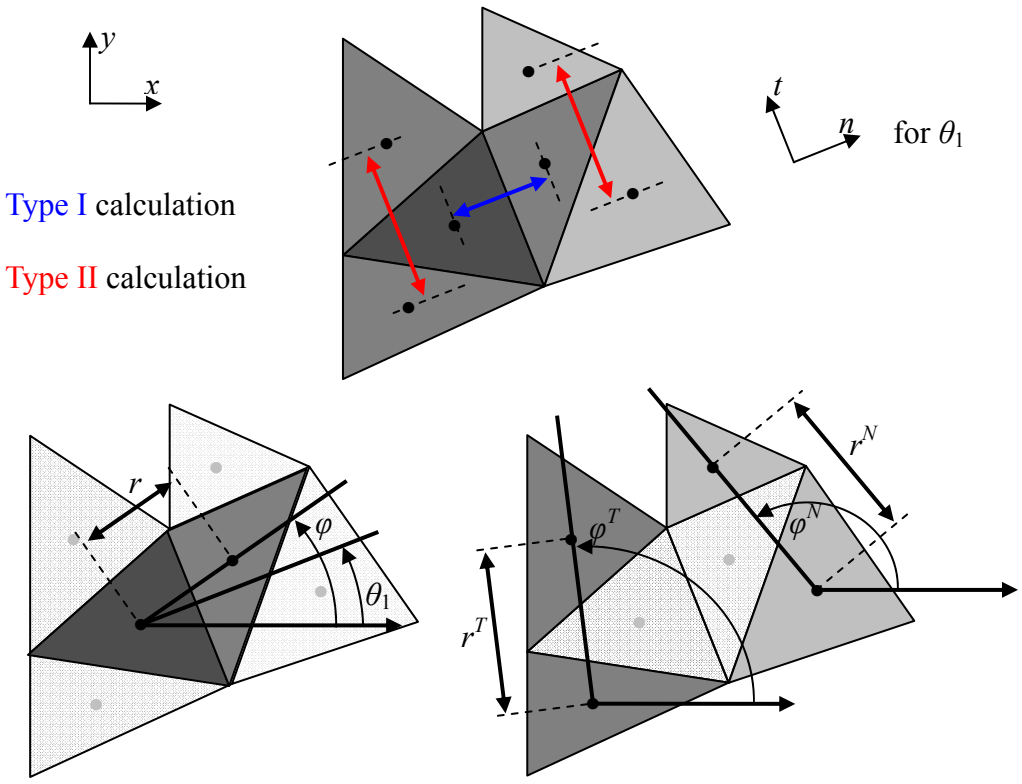


Figure 6: Illustration of the parameters used to obtain Δn and Δs from centroid and face angle information.

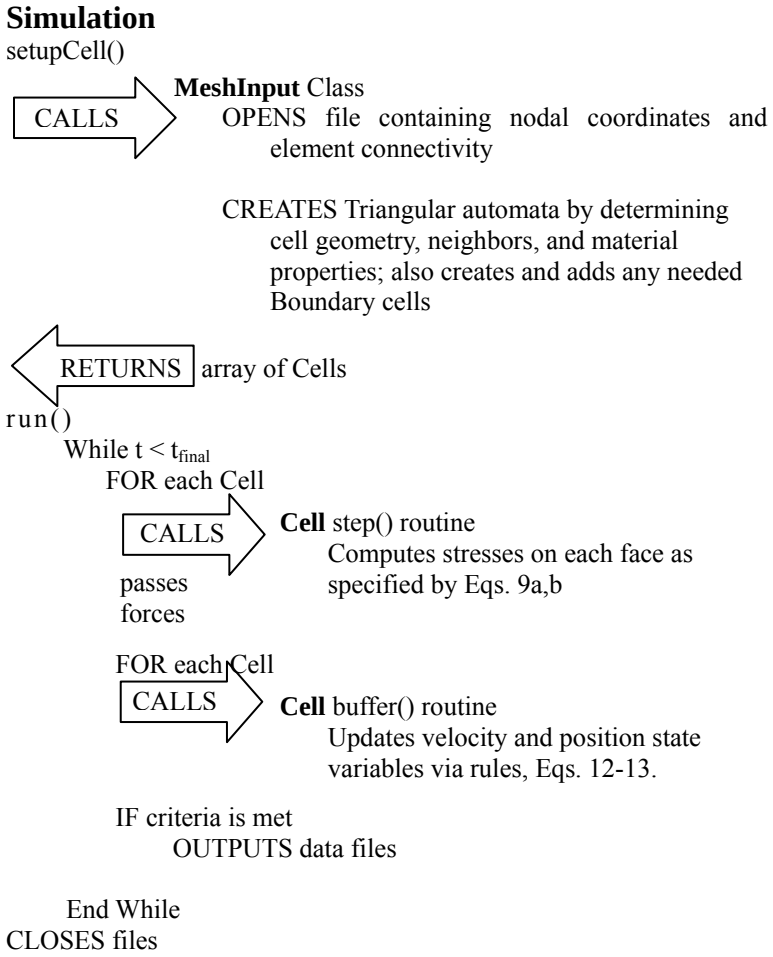


Figure 7: Psuedo-code depicting the fully object-oriented simulation approach used to update the automata states.

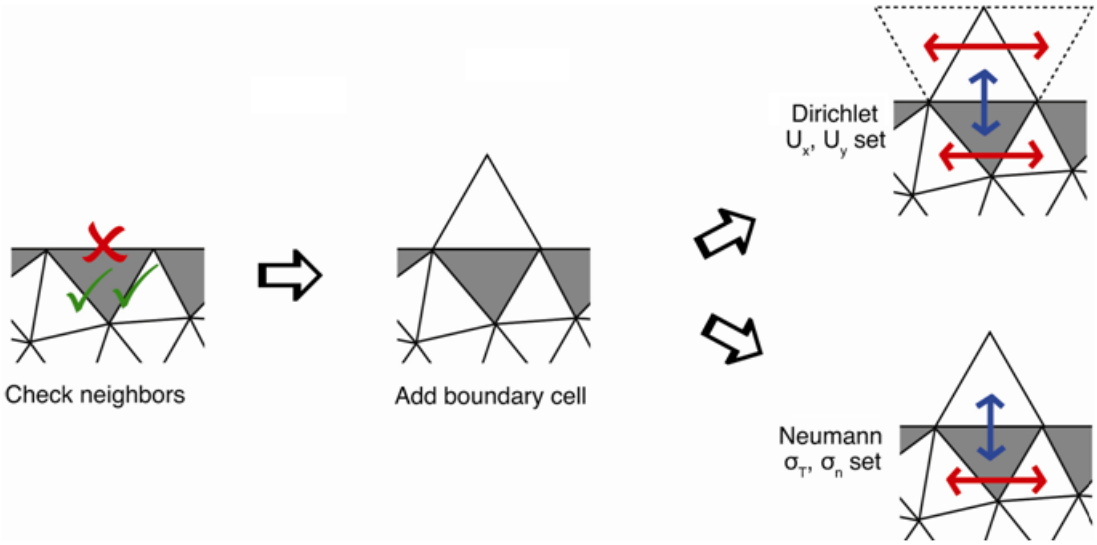


Figure 8: Method of applying a boundary cell. Any cell that fails a neighbor check gets a cell added to it with a parameter set reflecting that it is either displacement-imposed (Dirichlet) or traction-imposed (Neumann).

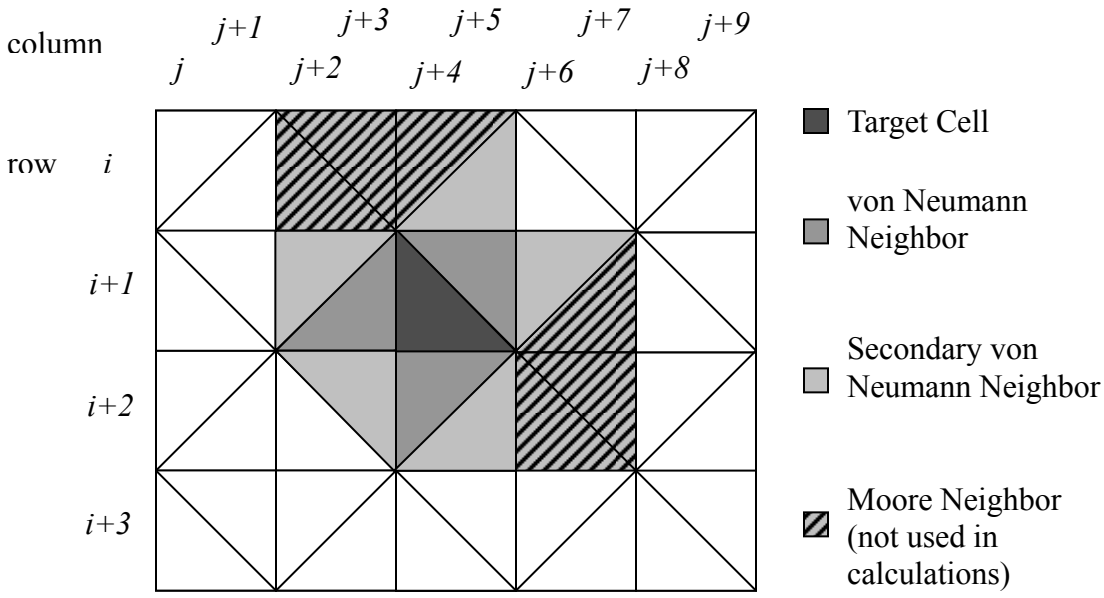


Figure 9: Geometry of the uniform triangular mesh used in the elastic half-space study.

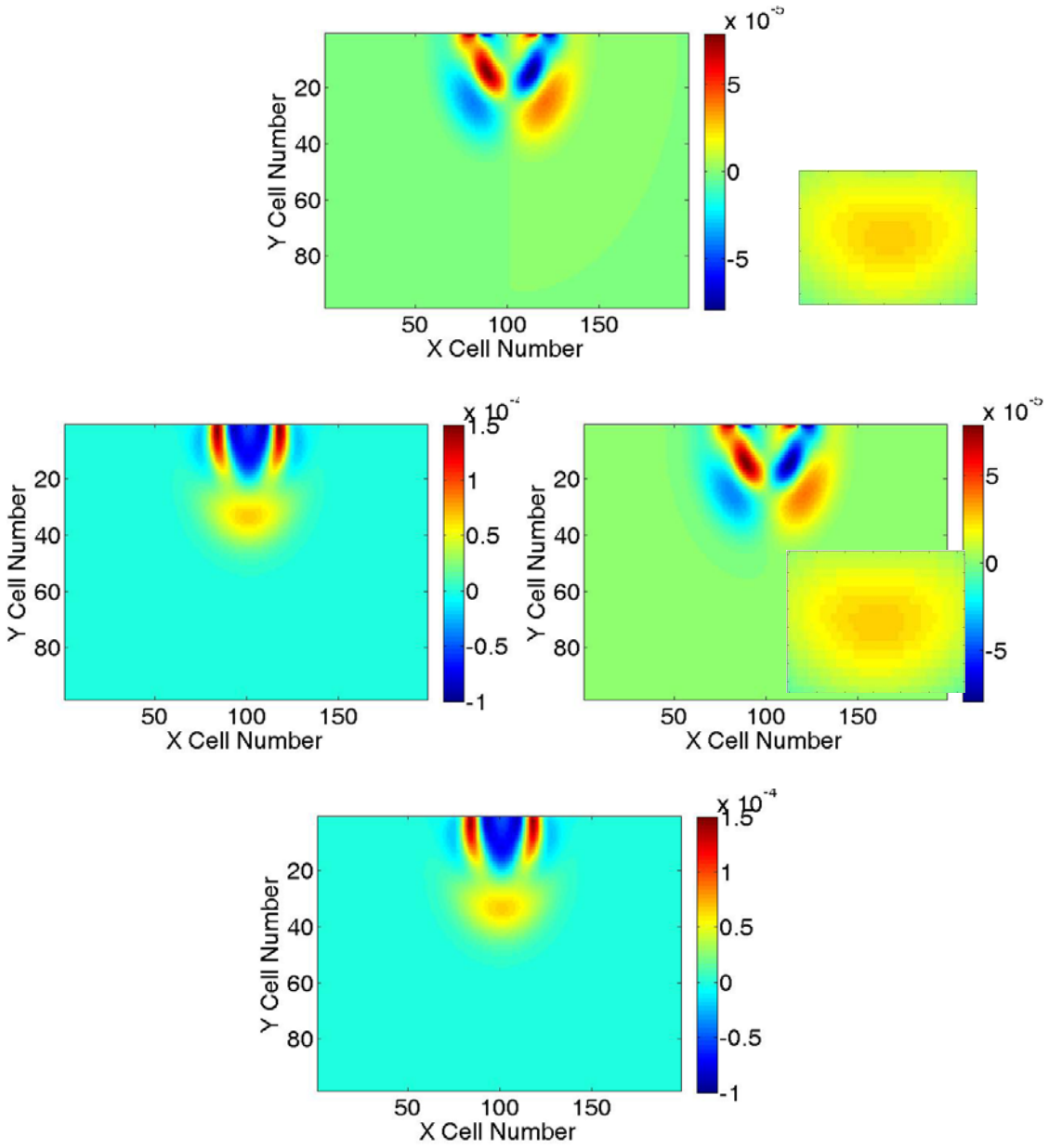


Figure 10: Comparison of displacement results computed for an elastic half-space using rectangular automata (top) and triangular automata (bottom): *x*-component (left) and *y*-component (right). The loading consists of a differentiated Gaussian pulse centered on the free surface.

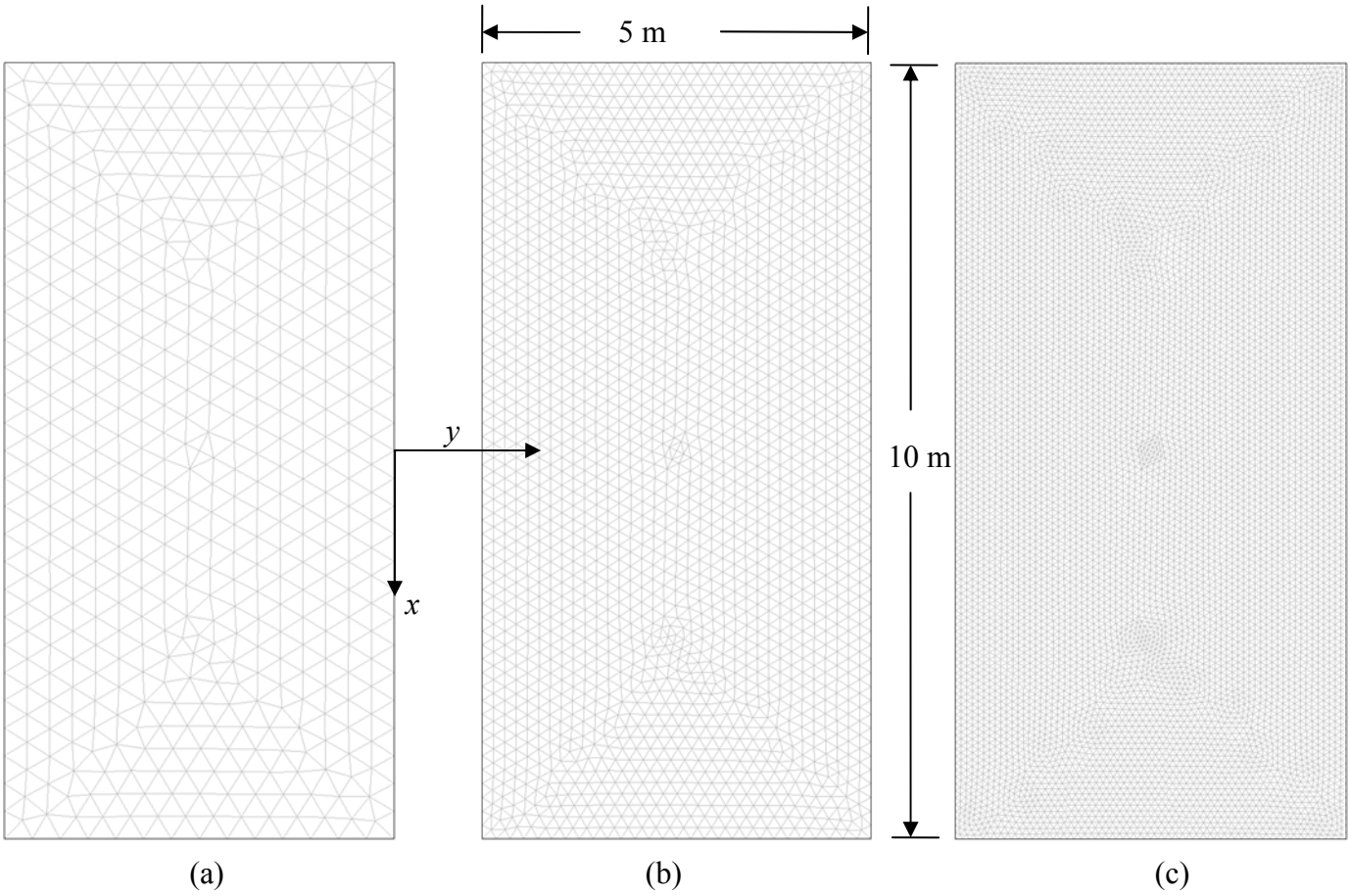


Figure 11: Triangular meshes used in the interior harmonic loading study: (a) coarse – 976 triangles, (b) fine – 3904 triangles, (c) finer – 15,616 triangles.

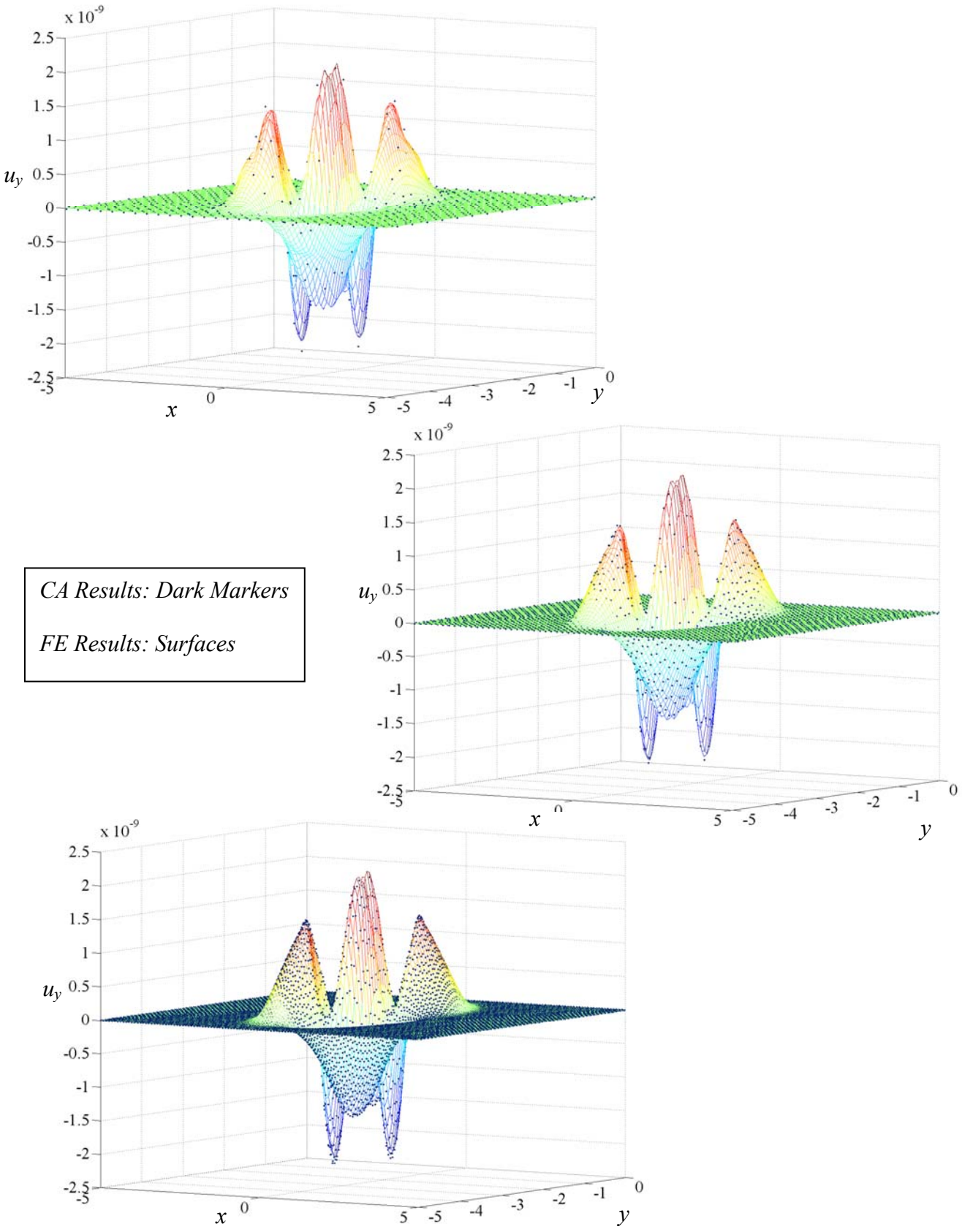


Figure 12: Results at time 5.0E-4 seconds for the *y*-component of displacement generated using CA (dark markers) and FE (surfaces) for the case of a harmonic interior load. Top to bottom: increasing mesh fineness.

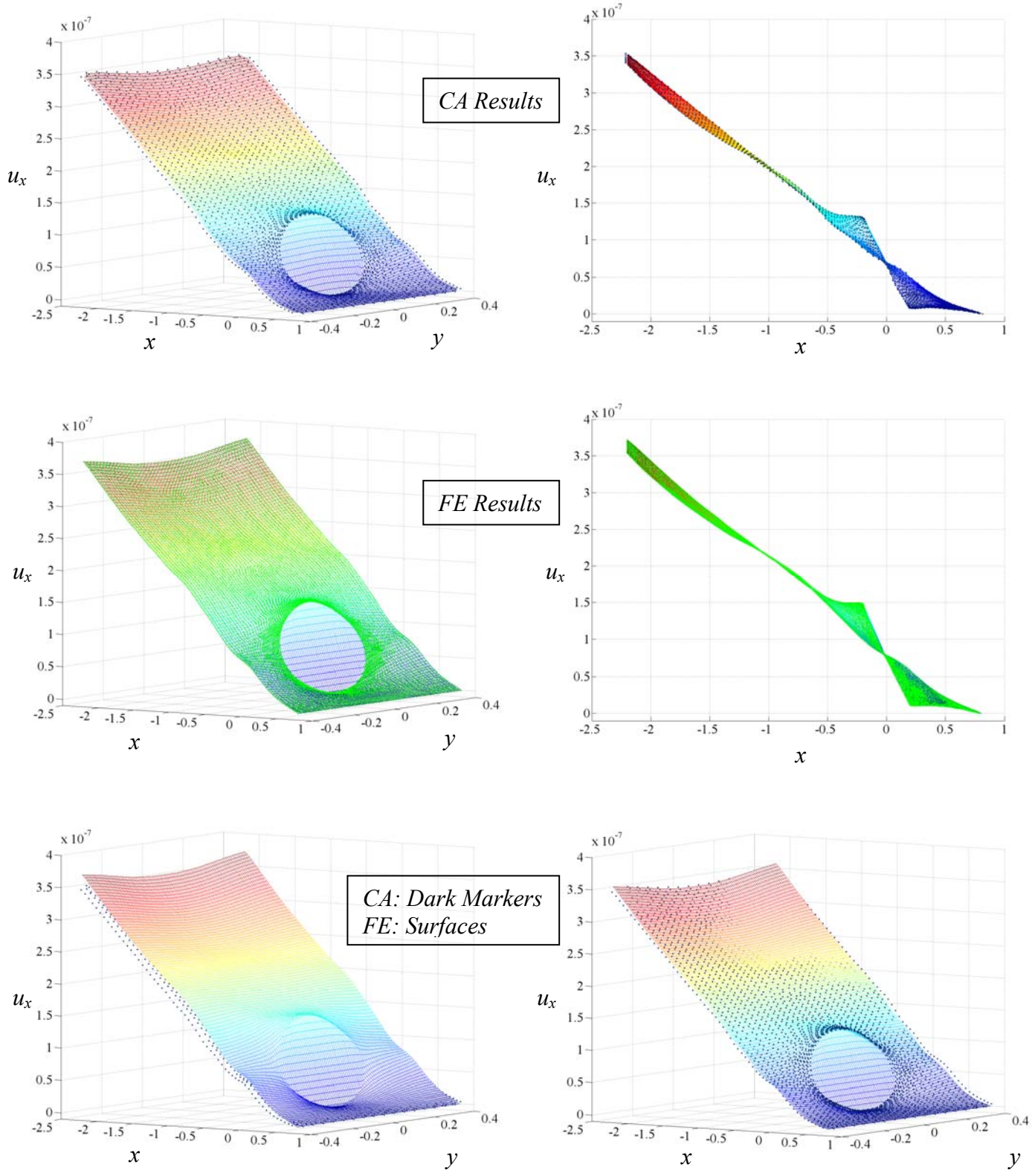


Figure 13: Results at time 5.0E-4 seconds resulting from a Neumann boundary condition. Shown are the *x*-components of displacement: two perspectives for the CA simulation (top row), FE (center row), and CA markers on top of FE surfaces (bottom row). The bottom row compares the CA results (markers) with the FE results (surfaces) at the same instant of time (left sub-figure) and when the FE results plotted are 2.0E-5 seconds earlier than the CA results (right sub-figure).

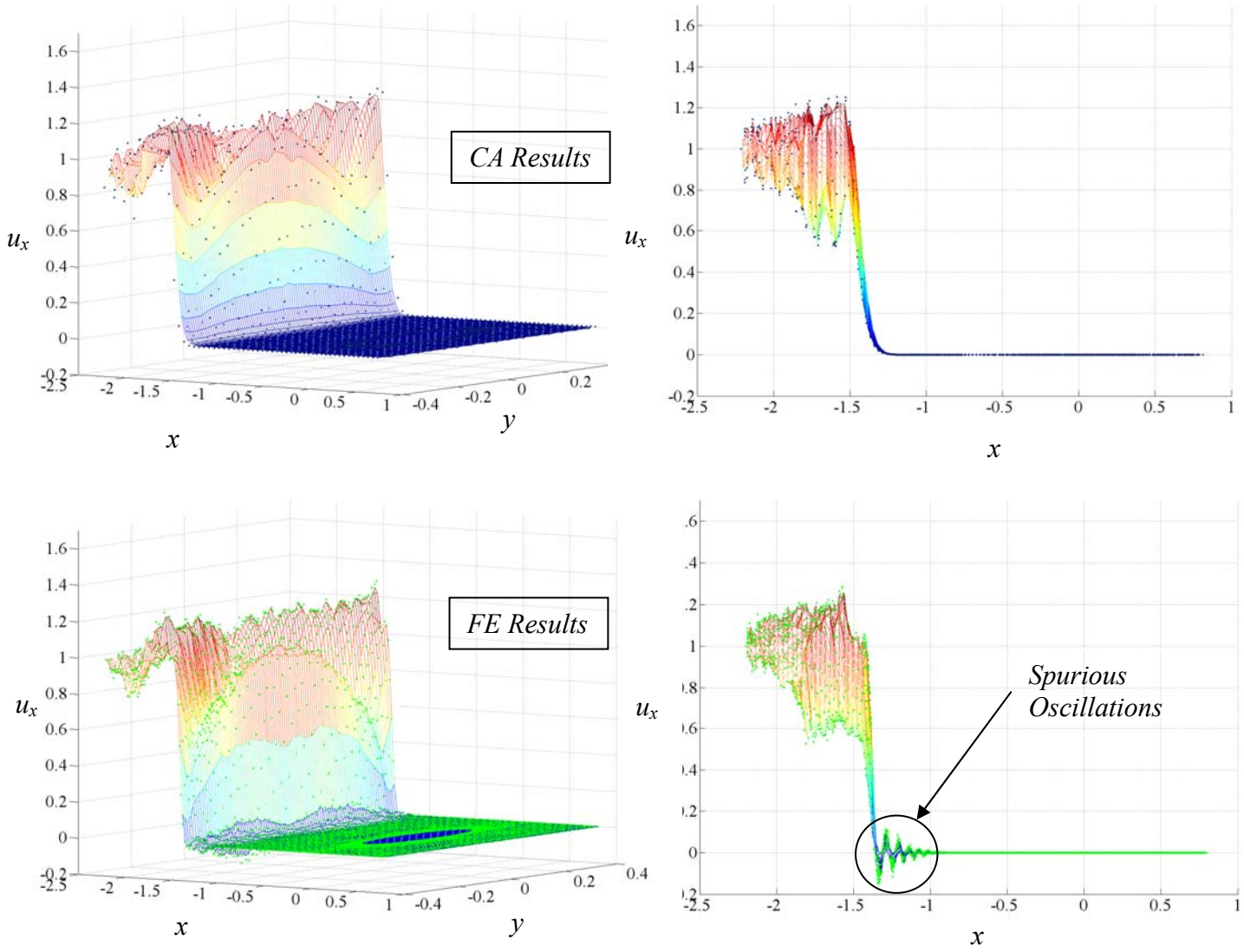


Figure 14: Results at time 1.3E-4 seconds resulting from a Dirichlet boundary condition. Shown are the *x*-components of displacement: results displayed using isometric and *x*-*z* views for CA (top row) and FE (bottom row).