LARGE GROWTH DEFORMATIONS OF THIN TISSUE USING SOLID-SHELLS

A Thesis Submitted to the
College of Graduate and Postdoctoral Studies
in Partial Fulfillment of the Requirements
for the degree of Master of Science
in the Department of Computer Science
University of Saskatchewan
Saskatoon

By
Danny Huang

PERMISSION TO USE

In presenting this thesis in partial fulfilment of the requirements for a Postgraduate degree from the University of Saskatchewan, I agree that the Libraries of this University may make it freely available for inspection. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my thesis work or, in their absence, by the Head of the Department or the Dean of the College in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Saskatchewan in any scholarly use which may be made of any material in my thesis.

Requests for permission to copy or to make other use of material in this thesis in whole or part should be addressed to:

Head of the Department of Computer Science
University of Saskatchewan
176 Thorvaldson Building, 110 Science Place
Saskatoon, Saskatchewan S7N 5C9 Canada

Or

Dean
College of Graduate and Postdoctoral Studies
University of Saskatchewan
116 Thorvaldson Building, 110 Science Place
Saskatoon, Saskatchewan S7N 5C9 Canada

ABSTRACT

Simulating large scale expansion of thin structures, such as in growing leaves, is challenging. Sold-shells have a number of potential advantages over conventional thin-shell methods, but have thus far only been investigated for small plastic deformation cases. In response, we present a new general-purpose FEM growth framework for simulating large plastic deformations using a new solid-shell growth approach while supporting morphogen diffusion and collision handling. Large plastic deformations are handled by augmenting solid-shell elements with *plastic embedding* and strain-aware adaptive remeshing. Plastic embedding is an approach to model large plastic deformations by modifying the rest configuration in response to displacement strain. We exploit the solid-shell's ability of describing both stretching and bending in terms of displacement strain to implement both plastic stretching and bending using the same plasticity model. The large deformations are adaptively remeshed using a strain-aware criteria to anticipate buckling and eliminate low-quality elements. We perform qualitative investigations on the capabilities of the new solid-shell growth approach in reproducing buckling, rippling, rolling, and collision deformations, relevant towards animating growing leaves, flowers, and other thin structures. The qualitative experiments demonstrates that solid-shells are a viable alternative to thin-shells for simulating large and intricate growth deformations.

ACKNOWLEDGEMENTS

I would like to thank my supervisor Dr. Ian Stavness for providing me the opportunity and enthusiastic support to work on my graduate studies. I would also like to thank Dr. John Lloyd from University of British Columbia; Dr. Przemyslaw Prusinkiewicz, Mikolaj Cieslak, and Andrew Owens from University of Calgary for their valuable discussions and feedback on the research project. The research project has also received support from the bright members of Dr. Ian Stavness' Biomedical Interactive Graphics (BIG) lab, and I am very grateful of having worked alongside with them.

This thesis is dedicated to my family and friends.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

List of Abbreviations

FEM	Finite Element Method
PDE	Partial Differential Equation
CCD	Continuous Collision Detection
LCP	Linear Complementarity Problem
SPD	Symmetric Positive Definite
AABB	Axis-Aligned Bounding-Box
BVH	Bounding Volume Hierachy
VT	Vertex-Triangle
EE	Edge-Edge
LHS	Left-Hand Side
RHS	Right-Hand Side
LOF	List of Figures
LOT	List of Tables

CHAPTER 1

INTRODUCTION

A common feature that is exhibited across all living things is growth. As growth occurs, their bodies undergo deformations, often resulting in the formation of intricate shapes and patterns. The morphological diversity found in nature has intrigued researchers towards learning the factors that govern and influence morphology. Knowing the factors can provide insights in applications, including plant breeding and tissue engineering. Plant breeders can modify and exploit the factors on a genetic level or environmental level for creating new varieties of plants with specific morphological traits [35, 48]. In tissue engineering, understanding the factors can assist researchers in developing tissue growth and regeneration techniques, particularly in terms of growth factor release [18, 40], scaffolding [19, 16], and external mechanical stress [5, 20]. Altogether, this raises an important scientific question of how morphological factors can be hypothesized and studied.

One area of development is computer physics simulators that can experimentally simulate the growth of tissue. These growth simulators are constructed using a number of components that model various aspects of growing tissue, such as using mass-spring elements with adjustable rest length springs to model tissue elasticity and plasticity. Within simulation, researchers have reproduced a number of objects that closely resemble counterparts found in nature. Some examples include the formation of rippling leaves [29]; blooming coral and flowers [37]; and fruits and vegetables [50]. Simulated plant models offer an accessible approach for researchers to experiment and analyze the impact of simulated processes on tissue morphology development [6, 38].

1.1 Overcoming the Challenges of Simulating Growth

Compared to elastic deformations, the animation of growing structures has received much less attention in the physics-based simulation literature. This is likely due to the unique and difficult challenges of simulating growing materials. A primary challenge is handling the complex deformations associated with growth. These deformations are large, permanent, and for thin tissue, often involve intricate shapes and patterns. The secondary challenge is accounting the factors that govern and influence morphology development. These factors can include the chemicals, referred as *morphogen*, that spread (i.e. diffuse) throughout tissue and signal where growth is to occur [44, 25], and also the collisions that occur between adjacent growing tissues [9, 42].

There are previous works on addressing the primary challenge of handling the complex deformations of growth. These previous works can be broken down into the components that were used: (1) constitutive model, (2) plasticity

model, and (3) remeshing algorithm, if any. Each component has their own set of advantages for supporting growth deformations.

Early physics-based simulations used mass-spring elements as the tissue constitutive model [29]. Mass-springs are popular for their simplicity, but are limited by their discontinuous nature. This limitation can be overcome using the finite element method (FEM), which is capable of modelling processes that are governed by partial differential equations (PDE) over a continuous domain. FEM-based tissue can model elasticity based on 3D stress-strain relationships, while supporting diffusion and anisotropic behavior [22]. However, these advantages come at a price of increased computational complexity, which allude to newer FEM frameworks that opt for simplified or reduced elements, such as thin-shell elements and solid-shell elements. These shell elements are specialized to handle steep bending deformations efficiently, allowing intricate deformations to be modeled. Thin-shells are simplified such that the thickness is incompressible and no shearing is assumed, making them specialized for modelling thin tissue [11]. In contrast, solid-shells are a reduced variation of the volumetric element that are notable for their support for thickness compressibility and shearing, and compatibility with existing 3D material laws.

Given a constitutive model, there are a number of methods in how plasticity can be modelled. Mass-springs can have adjustable spring rest lengths to induce permanent deformation [29]. For FEM simulations, an easy approach is *multiplicative decomposition* where the deformation gradient is decomposed into an elastic component and multiplicative plastic offset: $\mathbf{F} = \mathbf{F}_\mathrm{e}\mathbf{F}_\mathrm{p}$ [14]. For larger plastic deformations where the multiplicative plastic offset can cause numerical instability, growth simulations involving membrane[1] and volumetric elements use an alternative approach, referred as *plastic embedding* by Wicke et al. [49], where the stress-free rest configuration of the FEM model is permanently deformed using forces associated with the growth strain until the strain is minimized. Plastic embedding has been successful in modelling large plastic deformations for volumetric [22], membrane [23], and thin-shell elements [13]. However, for solid-shells, incorporating plastic embedding is largely unexplored.

When growth is prolonged, it is necessary to adaptively remesh the model to prevent mesh degradation. A common and simple method is to bisect edges that become too long [22]. Edges can also be flipped or collapsed to help avoid triangles that are too small or have poor aspect ratios [10, 23]. In existing growth simulators, these remeshing operations are triggered according to the geometry of the mesh. Outside of the growth literature, there exists strain-aware remeshers that can selectively and preemptively refine a mesh where surface buckling[2] is imminent [32].

Alongside with handling complex deformations, previous works have integrated other components together to govern and influence growth. A prominent component is morphogen diffusion which models the spreading of chemicals that signal growth [29, 22, 23]. Another component that influences growth are collisions. Especially for large deformations, growing tissue can take up specific shapes due to the space-constraints posed by neighboring tissue. Collision handling has been experimented with growth simulations involving mass-springs models [37] and FEM thin-filament models [46]. For growing FEM thin tissue, collision handling remains largely unexplored.

[1] Membrane elements are similar to thin-shell elements but do not model bending forces.

[2] Throughout the thesis, the term "buckling" refers to the sudden occurrence of bending due to the compression of elements.

1.2 Problem

Solid-shells remain undeveloped for handling the large plastic deformations to complement the intricate growth deformations. Much of the literature in growing solid-shells utilize the multiplicative decomposition approach and lack adaptive remeshing [50, 52, 51], which limit the solid-shell experiments within the scope of simulating small growth deformations. Consequently, solid-shells remain largely overlooked compared to thin-shells for handling large growth deformations despite the unique benefits that solid-shells offer.

1.3 Contribution

In this thesis, we present a general-purpose FEM growth framework that introduces a new solid-shell growth approach to handle large plastic deformations while integrating morphogen diffusion and collision handling. Specifically, we make the following contributions:

1. We enable large plastic deformations in solid-shells through plastic embedding. The plastic displacement strain is continuously embedded into the solid-shell's rest configuration to avoid large and volatile forces. Both plastic stretching and bending are supported under the same embedding procedure by exploiting the solid-shell's design of representing both stretching and bending using displacement strain.

2. We introduce strain-aware adaptive remeshing to growth simulation. The remesher preemptively ensures that intricate shapes and patterns can emerge, even from a starting coarse mesh. The adaptive remesher also eliminates low-quality elements to improve stability as large deformations occur.

3. We extend the growth tensor routine proposed by Kennaway et al. [22] to support plastic bending in solid-shells. The routine was originally designed to calculate the stretching growth strain for volumetric elements. We extend this routine to allow a growth tensor that can specify bending strain.

4. We introduce morphogen diffusion and collision handling to growing solid-shells. Both procedures are included to more accurately model the large growth deformations. Diffusion is handled using discrete differential geometry. Collisions are detected discretely and continuously, and are resolved using a linear complementarity problem (LCP) solver.

These contributions are brought together to enable and accommodate the formation of large growth deformations in solid-shells.

We perform qualitative investigations on the capabilities of the solid-shell growth framework in a number of contexts relevant to simulating the growth of leaves, fruits, and flower petals. These simulations illustrate the wide range of emergent patterns that the augmented solid-shells are able to achieve, including buckling, rippling, and rolling deformations. Beyond existing FEM growth frameworks, we include novel demonstrations of large and intricate plastic deformations with collision handling, and the ability to generate detailed growth models starting from a coarse mesh

by using the strain-aware remesher. Altogether, we provide qualitative experiments to demonstrate that solid-shells are a viable alternative to thin-shells elements for simulating growth involving large and intricate plastic deformations.

1.4 Organization of Thesis

The thesis is organized as follows: Chapter 2 provides a survey of related works and a summary of how our framework distinguishes itself from existing growth frameworks. The methods deployed by our framework are covered in Chapter 3. In Chapter 4, a number of simulated growth experiments are showcased to investigate the capabilities of the new solid-shell growth approach. Lastly, Chapter 5 summarizes the thesis contributions, limitations, and the future work that lies ahead.

CHAPTER 2

RELATED WORK

There are a number of existing contributions towards physics-based simulation of growth. In this Chapter, we present a survey of the recent related works. The survey primarily highlights the methods used and the advantages of each method. The existing frameworks and our framework are then compared in terms of methodology to distinguish and summarize the novelty of our framework.

2.1 Mass-Spring Models

The early studies of simulating growth have used mass-spring elements to model tissue. In those models, the physical properties of tissue is approximated as a set of point-masses where neighboring points are attached using springs. The springs can have their rest lengths modified to simulate plastic deformation, namely growth. In addition, springs can be attached between elements such that bending forces are accounted. Starting from a flat surface of mass-spring elements, researchers have been successful in simulating the growth of bulges and ripples that are similarly found in plant leaves [29, 36].

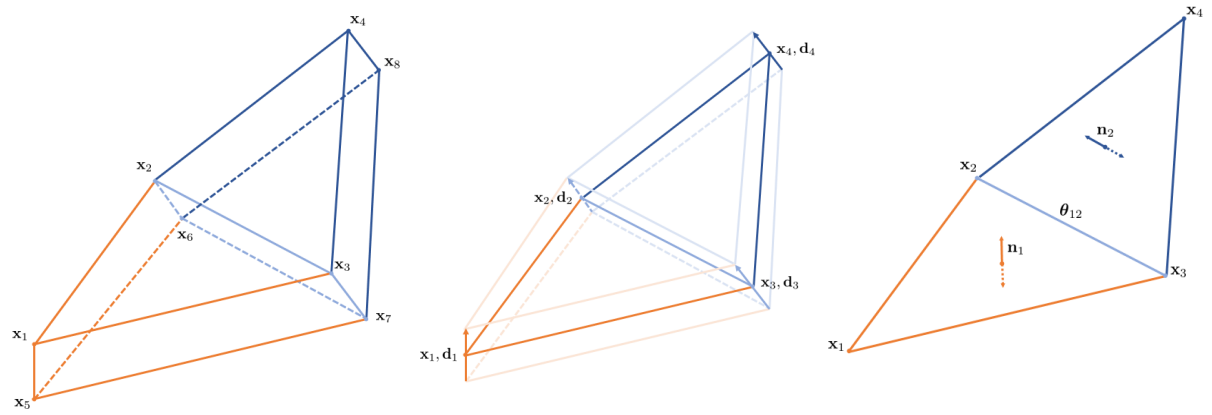
2.2 Finite Element Method

Growth can be more accurately simulated using the FEM [1]. The advantage of the FEM is that it can model PDE-governed processes over a continuous domain. 3D stress-strain relationships and diffusion are particularly modeled using the FEM and therefore can be coupled with FEM-based growth simulations. One of the pioneering works of simulating growth using FEM was done by Kennaway et al. [22]. They have created models of the *Antirrhinum* flower by diffusing growth-inducing morphogen over tissue composed of volumetric elements.

Beyond volumetric elements, there are other element types available, including the thin-shell element and solid-shell element. Figure 2.1 provides a quick comparison between these three element types. The thin-shell and solid-shell are elaborated in the subsequent sections.

2.3 Thin-Shell Elements

When simulating growth of thin tissues using the FEM, thin-shell elements can be used instead of volumetric elements. Thin-shells are notable for decoupling bending dynamics from stretching dynamics. This allows steep bends between



Volumetric (A) Solid-Shell (B) Thin-Shell (C)

Figure 2.1: Schematic Comparison between Volumetric, Solid-Shell, and Thin-Shell Elements. (A) shows a connected pair of volumetric wedge elements. Stretching and bending are both captured by the 3D position of each node $\mathbf{x}_k$ where $k \in [1..8]$. (B) shows a connected pair of solid-shell elements using a mid-surface based implementation [1]. Stretching is captured by the mid-surface nodes $\mathbf{x}_k$ where $k \in [1..4]$. Each mid-surface node $\mathbf{x}_k$ is accompanied by a director vector $\mathbf{d}_k$, which represents the thickness and rotation at node $\mathbf{x}_k$; bending dynamics are decoupled and delegated to the director vectors. (C) shows a connected pair of thin-shell elements. Likewise with the solid-shell, the thin-shell has mid-surface nodes to handle stretching dynamics. The thickness is always assumed to be constant and perpendicular to the mid-surface normal (e.g. $\mathbf{n}_1$) for a given thin-shell element. How rotation is decoupled depends on the thin-shell implementation; the figure particularly shows a hinge-based thin-shell implementation where bending is recorded as an angle between two neighboring elements (e.g. θ_{12}).

elements to occur without significantly impacting their stretching strain. Consequently, the risk of element inversion is greatly reduced when bending occurs, which is needed for simulating complex morphologies. Models composed of volumetric elements can tolerate large bends if they are high enough in resolution, but this comes at a price of increased computational complexity relative to using thin-shell elements.

Growth simulations involving thin-shell elements were successful in reproducing the behavior of thin structures subjected to growth. Experiments conducted by Chen et al. [7] were able to simulate the curling of paper and rippling of a disc by diffusing moisture into them, and were comparable to real-life experiments. In the works of Vetter [46], they produced fractal patterns (rippling within rippling) similar to those found in torn plastic sheets and beet leaves.

2.4 Solid-Shell Elements

Thin structures can alternatively be modelled using solid-shell elements. Solid-shells are a specialization of volumetric elements where stretching and bending are decoupled and represented by the translation and rotation of its *directors*. These directors, which can be thought as extensible normal vectors, can be described exclusively in terms of node displacement, allowing bending dynamics to be described without introducing notions of angular displacement.

[1] Our framework uses the alternative front and back node implementation where the solid-shell directors are described using the front and back node positions. The front and back node implementation is elaborated later in Chapter 3. The mid-surface based implementation is shown in Figure 2.1 to compare with the thin-shell more easily.

Consequently, solid-shells are able to (1) interface with existing linear and hyperelastic 3D material laws, (2) retrain thickness compression and shearing dynamics, (3) and describe both stretching and bending in terms of displacement strain. In contrast to (3), thin-shells have a separate formulation for bending dynamics based on angular strain (Narain et al. [31]) or second fundamental form (Vetter et al. [47], Chen et al. [7]). This separation particularly complicates adapting different materials and therefore thin-shells are usually fixed to the St. Venant-Kirchhoff material which is only valid for small strains [2]. In essence, solid-shells can be thought as a compromise between volumetric and thin-shell elements: solid-shells trade off some of simplified thickness assumptions of thin-shells to gain benefits from volumetric elements.

2.5 Plasticity using Multiplicative Decomposition

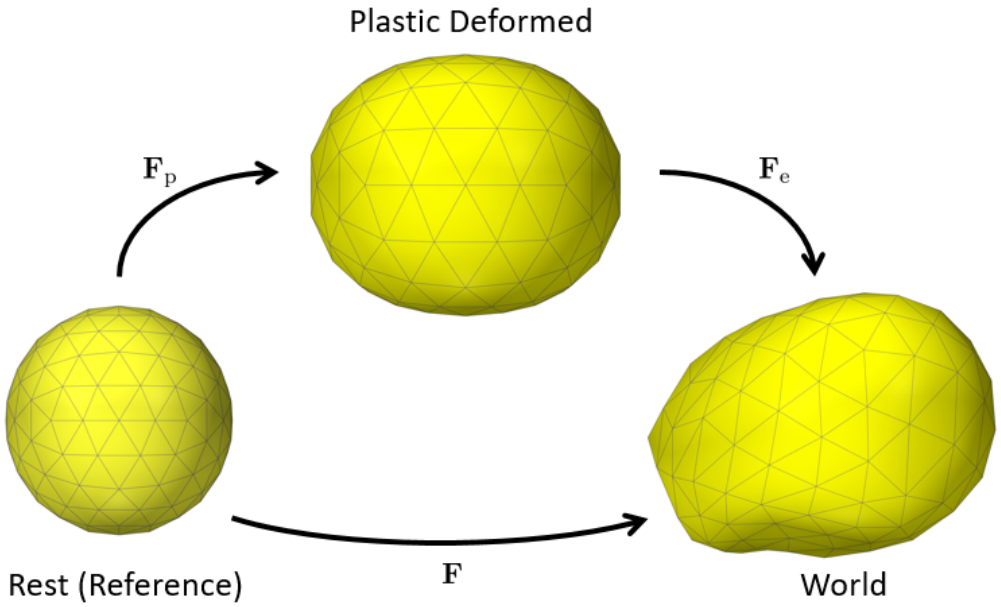


Figure 2.2: Deformation Gradient Decomposition to Handle Plasticity. The deformation gradient $\mathbf{F}$ is a description of the object's world configuration (right) with respect to its stress-free rest configuration (left). To account for plastic deformations, $\mathbf{F}$ can be decomposed into an elastic and plastic component. The plastic component $\mathbf{F}_p$ describes the object's permanently deformed configuration (top middle), which can serves as the reference configuration for the elastic component $\mathbf{F}_e$ to describe the world configuration.

Existing FEM growth simulators have their own approaches for modelling plastic deformations. A simple approach is to decompose the *deformation gradient* into an elastic component and multiplicative plastic offset: $\mathbf{F} = \mathbf{F}_e\mathbf{F}_p$ [14, 46, 51]. The deformation gradient by itself is a mapping from the object's stress-free rest configuration to its elastically deformed world configuration. By decomposing the deformation gradient into a plastic and elastic component, the object's plastically/permanently deformed state can be tracked via the plastic offset $\mathbf{F}_p$, which serves as a reference for the elastic component $\mathbf{F}_e$ to describe the world configuration. This plasticity approach is visualized in Figure 2.2. While this allows both elasticity and plasticity to be modelled together easily, the problem is that numerical instability

can arise when the plastic strain associated with $\mathbf{F}_\mathrm{p}$ becomes too large.

2.6 Plasticity using Rest Configuration Deformation

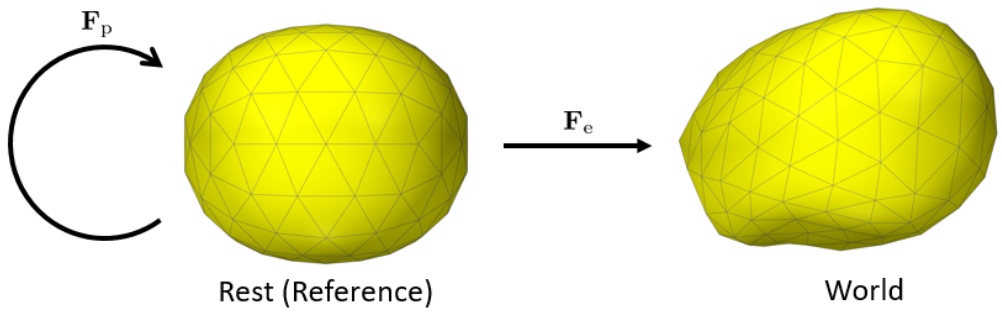


Figure 2.3: Modifiable Rest Configuration to Handle Plasticity. The object's stress-free rest configuration is allowed to alter according to the plastic deformation gradient $\mathbf{F}_\mathrm{p}$. The new rest configuration described by $\mathbf{F}_\mathrm{p}$ is always with respect to the current rest configuration, which is discarded and replaced whenever the new rest configuration is solved. The world configuration, which accounts for elastic deformations, is with respect to the current rest configuration.

In other works, the rest configuration is allowed to deform. See Figure 2.3 for an overview of this alternative plasticity approach. The advantage of this approach is that $\mathbf{F}_\mathrm{p}$ decrements whenever the rest configuration deforms; $\mathbf{F}_\mathrm{p}$ gradually becomes represented by the rest configuration. In effect, the plastic strain associated with $\mathbf{F}_\mathrm{p}$ is actively minimized, allowing large plastic deformations to occur without encountering numerical instability. Termed as *plastic embedding*, Wicke et al. [49] provides a formal description of this plasticity approach for volumetric elements. Plastic embedding and its variations have been adopted for growing volumetric elements [22], membrane elements [23], and thin-shell elements [13]. For solid-shell elements, plastic embedding is largely unimplemented but could be achieved by modifying the rest configuration of its directors.

A caveat to plastic embedding is that the rest configuration of the model is restricted within Euclidean space. For example, given two cube elements that are attached side-by-side (i.e. sharing four vertices), it is not possible to represent a cube that is 2x bigger than the other cube while still sharing four vertices. To get around this caveat, the non-embeddable portion of the plastic strain (i.e. residual strain) is retained and carried until it can be embedded into the rest configuration [22, 31]. Other works utilize non-Euclidean plates developed by Efrati et al. [11] that represent the non-Euclidean membrane rest configuration in a modifiable reference metric tensor, which can be extended to include the bending rest configuration [7]. The reference metric tensor is assumed to be always known and is manipulated directly according to customized rules, such as rest curvature being linearly dependent on moisture concentration [7].

2.7 Adaptive Remeshing

Growth models should be periodically remeshed to maintain a high quality topology. This involves routinely updating the mesh such that its resolution does not degrade during growth. Otherwise, small and intricate morphologies, such

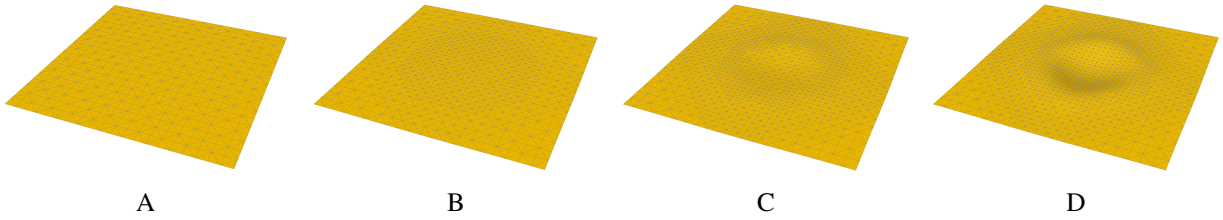


Figure 2.4: Strain-Aware Adaptive Remeshing. (A) shows a starting coarse mesh. When the center of the mesh is subjected to stretching strain, a strain-aware remesher can adaptively increase the topology resolution where strain is present, as shown in (B), (C), and (D).

as wrinkles and ripples, would be prevented from emerging. In the simulations by Kennaway et al. [22], remeshing is carried out by subdividing edges that either exceed a length threshold or morphogen concentration threshold. Along with subdivision, edges can be flipped to help avoid "skinny" elements that have poor aspect ratios [37, 23]. As well, edges can be collapsed to eliminate elements that either become too skinny or too small [10]. In more sophisticated remeshing schemes, strain-aware thresholding is supported: heavily strained elements are refined to anticipate potential surface buckling, and flat areas with little strain are coarsened into lower resolution elements to reduce computational complexity [32]. Figure 2.4 provides a visualization of strain-aware adaptive remeshing.

2.8 Diffusion

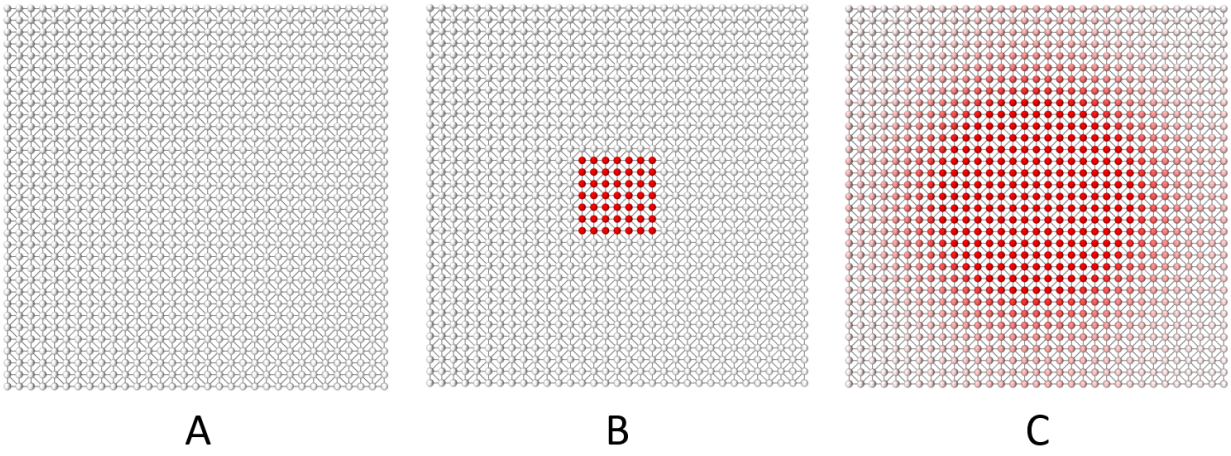


Figure 2.5: Diffusion Across a Mesh Surface. (A) shows a square mesh composed of connected white spherical nodes. (B) shows the same square mesh but with morphogen applied in the center, indicated by the red nodes. The red morphogen is allowed to diffuse throughout the mesh for some amount of time, resulting in (C).

The spreading of growth-inducing morphogen is modeled as a diffusion process. The visual effect of diffusion is shown in Figure 2.5. Diffusion is commonly simulated by solving the heat equation. The heat equation is a second-order PDE that is dependent on the mathematical Laplacian operator Δ. At a high-level, solving the heat equation refers to double integrating the heat equation to measure the quantity change of the diffusing substance at discrete points. This double integration can be performed using the FEM which involves analytically integrating the discrete weak

form of the Laplacian and performing the second integration using a numerical time-stepping method (e.g. backward Euler). When integrating the Laplacian weak form, discrete differential geometry can be exploited to perform the integration, essentially allowing a mass matrix and *discrete cotangent Laplacian matrix* to be constructed quickly [39]. These two matrices are then passed to the time-stepping method which can be solved in very close to linear time due to the symmetric positive definite (SPD) property of both matrices. Altogether, this particular diffusion method has been incorporated to simulate morphogen diffusion efficiently [23]. In another work, geodesic distances are computed using the discrete cotangent Laplacian matrix, enabling a specific growth model where the growth rate is dependent on the distance from the tissue boundary [37].

2.9 Collision Handling

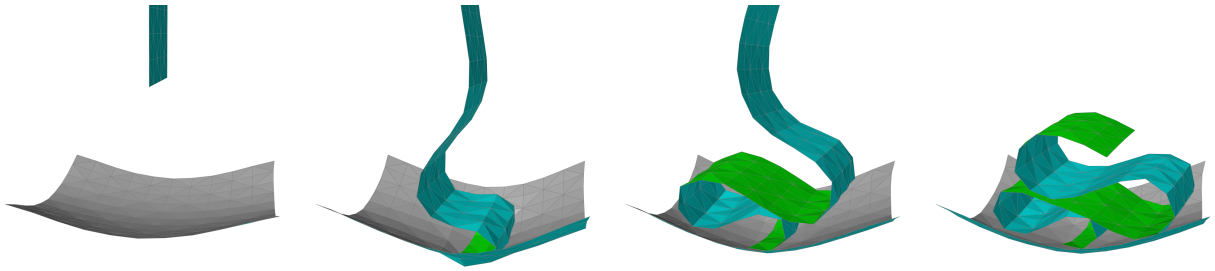


Figure 2.6: Collision Handling Demonstration. A flexible solid-shell sheet is dropped onto another sheet that has its four corners pinned in-place. The collision handler prevents the dropped sheet from self-intersecting nor penetrating through the other sheet.

When growth is prolonged, the model can potentially intersect with itself, justifying the need for collision handling. Figure 2.6 provides a simple visual demonstration of collision handling. A straight-forward approach is to deploy a discrete collision detection scheme with repulsion forces. In the growth simulations done by Rosenkrantz and Louis-Rosenberg [37], nearby vertices are detected whenever their individual invisible ellipsoid-shaped collision bodies intersected, which are repelled apart using spring forces. In another implementation, nearby contact between mesh faces and edges are explicitly detected and repelled using normal forces [46].

Alternatively, continuous collision detection (CCD) with constraint-specified impulses can be used to avoid collisions [34]. The advantage of CCD is that the time-space path of the individual vertices, edges, and triangles are accounted for, allowing collisions to be detected precisely and robustly. When collisions are detected, they are resolved using impulses that are specified by geometric constraints. The impulse approach allows energy and momentum to be conserved and does not require stiffness parameter tuning [43].

Collision impulses can be solved using iterative methods, such as the Gauss-Seidel or Jacobi method, or by using direct methods. Direct methods, particularly LCP solvers, can simultaneously account for all the given geometric constraints to guarantee that the impulses do not violate each others' constraints [33]. LCP solvers are also compatible with GPU-based implementations to accelerate collision handling [24].

While CCD can detect collisions exactly as they occur, the downside is that collisions become more difficult to

resolve in certain cases, such as the case of tangled and pinched meshes [33]. The reasoning is that, unlike its discrete counterpart, CCD does not enforce a minimum distance between nearby pairs, making it more susceptible to creating new secondary collisions when the impulses are applied. To increase the robustness of collision handling, previous work has combined discrete and continuous collision detection to offset each of their disadvantages [3].

2.10 Summary

Table 2.1: Distinguishing our growth framework from recent related works in the physics-based growth literature. The abbreviated terms are as follow: Mass-Spring (**S**), FEM Membranes or Volumes (**MV**), FEM Solid-Shells (**SS**), FEM Thin-Shells (**TS**), Rest Length (**L**), Multiplicative Offset (**M**), Plastic Embedding (**E**), Reference Metric Tensor (**T**), Length-Aware (**Length**), Strain-Aware (**Strain**), Discrete Differential Geometry (**DDG**), Discrete Detection (**D**), Continuous Detection (**C**), Spring or Normal Forces (**F**), Impulses (**Imp**), Open-Source (**OS**).

	Constitutive				**Plasticity**				**Adaptive Remesh**		**Diffusion**		**Collision Handling**				
Framework	**S**	**MV**	**SS**	**TS**	**L**	**M**	**E**	**T**	**Length**	**Strain**	**Yes**	**DDG**	**D**	**C**	**F**	**Imp**	**OS**
Kennaway et al. [22]		✗					✗		✗		✗						✗[2]
Vetter [46]				✗		✗							✗		✗		
Rosenkrantz and Louis-Rosenberg [37]	✗				✗				✗		✗	✗	✗		✗		
Chen et al. [7]				✗				✗			✗						
Kierzkowski et al. [23]		✗					✗		✗		✗	✗					
Gingras and Kry [13]				✗			✗				✗	✗					
Zheng et al. [51]			✗			✗											
Our Growth Framework			✗				✗		✗	✗	✗	✗	✗	✗		✗	✗[3]

We summarize the recent related works and our growth framework in Table 2.1. Our open-source growth framework specializes in growing solid-shell elements that are equipped with plastic embedding and a strain-aware remesher, contributing the ability for solid-shells to handle large and intricate plastic deformations. We further extend the new solid-shell growth approach by providing support for modelling morphogen diffusion and collision handling, which altogether govern and influence the growth process. Morphogen diffusion is modelled efficiently by leveraging on discrete differential geometry. Collision handling is made robust using a combination of discrete and continuous collision detection where collisions are resolved using impulses from the LCP solver.

[3] http://cmpdartsvr3.cmp.uea.ac.uk/wiki/BanghamLab/index.php/GFtbox
[3] https://github.com/dannyhx/artisynth_core/tree/shell_growth

CHAPTER 3

METHOD

Algorithm 1: Growth Method

for each time step from $k = 0$ **to** n **do**
    Remesh (3.8)
    Diffuse morphogen (3.4)
    Transform fraction of morphogen into plastic strain (3.5)
    Calculate plastic forces using plastic strain (3.6)
    Advance rest configuration using plastic forces (3.7)
    Advance world configuration using elastic forces [1]
    Perform collision handling (3.9)
end

3.1 High-Level Overview

The framework methodology is outlined in Algorithm 1 with corresponding sections highlighted in red. Growth is simulated by deforming the rest configuration of the solid-shells wherever morphogen is present. The user first specifies the initial locations of the morphogen on a FEM node basis, either through scripting or by using the provided user-interface tool. The morphogen can vary in concentration, *type*, and *subtype*. The morphogen types differ in terms of growth direction, while morphogen subtypes differ by whether plastic stretching or plastic bending is induced. Morphogen is simulated to diffuse from node to node by solving the heat equation. After a single diffusion time step, a fraction of the morphogen is absorbed to compute the plastic strain. The magnitude and direction of the plastic strain are dependent on the concentration and type of the absorbed morphogen. Plastic bending is induced by localizing the plastic strain along the bottom virtual surface of the solid-shells, as opposed to all surfaces for plastic stretching.

From the plastic strain, plastic forces are computed. These forces are derived from the solid-shell formulation and the linear material model. The material can be substituted with another 3D material (e.g. Neo-Hookean and Mooney-Rivlin) if desired. The plastic forces are then used to deform the model's rest configuration using an implicit time integration scheme. The portion of the plastic strain that is not embedded/represented in the Euclidean rest configuration is carried over to the next time step and is regarded as the residual strain. We follow Kennaway et al. [22] in that external forces are absent during growth, allowing the world configuration to simply mirror the rest configuration. But

[1] If external forces are absent, then the world configuration does not need to be solved and can mirror the rest configuration.

if external forces are desired, elastic forces can be derived by either using the same 3D material that is used to compute the plastic forces or by using a fast linear corotational material model [30]. The elastic forces are passed to the same implicit time integration scheme to solve the world configuration.

Collision handling is carried out to detect and repel apart features that are nearby or colliding. Nearby features are detected using discrete collision detection and then repelled apart using impulses that are computed according to geometric constraints. Features that are colliding are detected and repelled in the same matter but using continuous collision detection instead. Because the world and rest configurations occupy the same space in our growth simulations, collision handling is performed on the rest configuration.

Adaptive mesh refinement is performed by bisecting, flipping, and collapsing edges. The amount of refinement or coarsening is based the edge lengths, curvatures, velocities, and plastic strain of the rest configuration. Elastic strain is also accounted as needed.

To ensure reproducibility of the methods and experiments, the source code of the growth framework is made public. The growing solid-shells, morphogen diffusion procedure, adaptive remesher, and collision handler are all implemented and integrated together from the ground-up in the common Java programming language. The experiments can be reproduced by running the preconfigured scripts that are included with the growth framework.

3.2 Finite Element Method

The growth framework is built upon the FEM, which is a numerical method for solving partial differential equations. Many natural phenomena are described using partial differential equations, and therefore can be simulated using the FEM. This section provides a general understanding of the FEM in the context of simulating elastic deformations using simple volumetric elements, serving as a foundation for discussing more specific topics, including solid-shell elements and growth. Simulating elastic deformations using the FEM consists of 9 steps which are elaborated in the subsequent sections:

1. Divide deformable object into discrete elements. (3.2.1)
2. Distribute integration points (i.e. sampling points) across each element. (3.2.2)
3. Calculate the deformation gradient, strain, and stress at each integration point. (3.2.4, 3.2.5, 3.2.6)
4. Solve for the elastic forces by quadrature. (3.2.7)
5. Solve for the stiffness matrices by quadrature. (3.2.8)
6. Assemble the forces and stiffnesses into global form. (3.2.9)
7. Solve for the velocities to advance the deformed object. (3.2.10)
8. Repeat steps 3 to 7 as needed.

3.2.1 Discretization

The first step is to subdivide the problem domain into discrete *elements*. The elements are represented using simple geometric shapes. For instance, a deformable object can be subdivided into tetrahedral elements. Each element has a set of *nodes*, which typically correspond to the vertices of the shape, and are shared between adjacent elements. The nodes themselves hold data variables that are relevant to the problem. For example, a heat flow simulation would keep track of a temperature variable at each node.

Each shape has its own set of *shape functions*, also known as basis functions, which describe the domain space within the element. Shape functions can interpolate the known nodal data values to an arbitrary point within the element. Assuming that the shape functions are linear, the interpolation is formulated as

$$V = \sum_{i}^{r} \mathrm{N}_i(\boldsymbol{\xi})\, V_i$$

where r is the number of nodes of the element, V_i is the data value associated with node i, N_i is the shape function of node i, and V is the interpolated data value at the given arbitrary point, which is specified using local coordinates $\boldsymbol{\xi} = (\xi_1, \xi_2, \xi_3)$. See Figure 3.1 which lay out the local coordinate system of a tetrahedral element. The shape function themselves, which for a tetrahedral element, are defined as

$$\mathrm{N}_1(\boldsymbol{\xi}) = 1 - \xi_1 - \xi_2 - \xi_3 \qquad \mathrm{N}_2(\boldsymbol{\xi}) = \xi_1 \qquad \mathrm{N}_3(\boldsymbol{\xi}) = \xi_2 \qquad \mathrm{N}_4(\boldsymbol{\xi}) = \xi_3.$$

Regardless of the geometric shape used, its set of shape functions have a property of summing to 1 and that the i-th shape function will evaluate to 1 with others evaluated to 0 if the input local coordinates $\boldsymbol{\xi}$ corresponds to the local coordinates of node i.

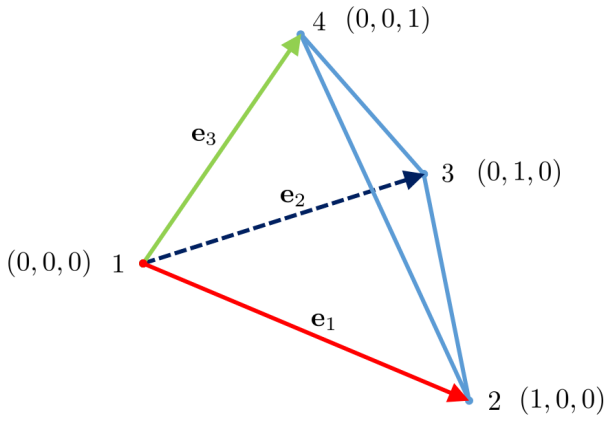


Figure 3.1: Tetrahedral Element. The local coordinates (ξ_1, ξ_2, ξ_3) is provided for each node i. The axes of the local coordinate system are individually labeled as $\mathbf{e}_1$, $\mathbf{e}_2$, and $\mathbf{e}_3$.

3.2.2 Integration Points

Each element has a set of integration points that are distributed within the element. The integration points are analogous to the quadrature points used in the Guassian quadrature scheme. These integration points allow difficult expressions

that require integrating over the element volume to be approximated by quadrature. For a tetrahedral element, it typically only has one integration point:

$$\begin{aligned} \boldsymbol{\xi}_k &= (\xi_{1,k}, \xi_{2,k}, \xi_{3,k}) & w_k &= 1 \\ &= (0.5, 0.5, 0.5) \end{aligned}$$

where $\boldsymbol{\xi}_k$ and w_k are the local coordinates and *weight* of the single integration point k. The weight is the fraction of the element volume that the integration point is representing.

3.2.3 Jacobian Matrix

Nodal data values can be described as a gradient. Gradients are measured at integration points and are represented as a 3×3 *Jacobian matrix*. Assuming that each data value is a 3D quantity (e.g. 3D position), the Jacobian matrix is defined as:

$$\mathbf{J} = \frac{\partial \mathbf{x}}{\partial \boldsymbol{\xi}} = \begin{bmatrix} \frac{\partial x}{\partial \xi_1} & \frac{\partial y}{\partial \xi_1} & \frac{\partial z}{\partial \xi_1} \\ \frac{\partial x}{\partial \xi_2} & \frac{\partial y}{\partial \xi_2} & \frac{\partial z}{\partial \xi_2} \\ \frac{\partial x}{\partial \xi_3} & \frac{\partial y}{\partial \xi_3} & \frac{\partial z}{\partial \xi_3} \end{bmatrix}.$$

The Jacobian matrix describes the change in the nodal data value $\mathbf{x} = (x, y, z)$ with respect to the change in local coordinates at a given integration point. In other words, the matrix describes how the nodal data value changes as you infinitesimally traverse through an element, starting from the given integration point. In the context of deformation simulation, the Jacobian matrix is calculated using

$$\mathbf{J} = \sum_i^r \left(\mathbf{x}_i \otimes \frac{\partial \mathrm{N}_i(\boldsymbol{\xi}_k)}{\partial \boldsymbol{\xi}} \right) \tag{3.1}$$

where r is the number of nodes of the element, $\otimes$ is the outer product operator, N_i is the i-th shape function, and $\mathbf{x}_i$ is the 3D position of node i in *global coordinates*. Node positions that are specified in global coordinates refer to the *global coordinate system*. Unlike the local coordinate system, which is uniquely defined for each element, the global coordinate system is shared across all elements. The axes of the global coordinate system is always fixed and commonly corresponds to the 3D orthogonal Euclidean coordinate system.

3.2.4 Deformation Gradient

The elastic forces are dependent on the deformation that the object is experiencing. The *deformation gradient* offers a measure of deformation and it is calculated at each integration point of the object:

$$\mathbf{F} = \begin{bmatrix} \frac{\partial x}{\partial X} & \frac{\partial y}{\partial X} & \frac{\partial z}{\partial X} \\ \frac{\partial x}{\partial Y} & \frac{\partial y}{\partial Y} & \frac{\partial z}{\partial Y} \\ \frac{\partial x}{\partial Z} & \frac{\partial y}{\partial Z} & \frac{\partial z}{\partial Z} \end{bmatrix}.$$

The deformation gradient describes, in gradient form, the deformed object relative to its *rest configuration*. The rest configuration of an object is the shape that the object takes when subjected to zero forces (i.e. no deformations).

By convention, any coordinate that is with respect to the object in its rest configuration is capitalized. For example, in the context of deformation simulation, $\mathbf{x}_i$ and $\mathbf{X}_i$ denote the position of node i in its *world configuration* and rest configuration respectively. The world configuration of an object is simply the current shape of the object, which differs from the rest configuration when deformations are present. Coordinates that refer to the objects in their rest configuration reside in the *reference-space* while coordinates that refer to the objects in their world configuration reside in the separate *world-space*.

Calculating the deformation gradient is straight forward using the Jacobian matrix:

$$\mathbf{F} = \frac{\partial \mathbf{x}}{\partial \mathbf{X}} = \frac{\partial \mathbf{x}}{\partial \boldsymbol{\xi}} \frac{\partial \boldsymbol{\xi}}{\partial \mathbf{X}} = \mathbf{J}\mathbf{J}_0^{-1} \tag{3.2}$$

where Jacobian matrix $\mathbf{J}$ is calculated using equation (3.1) and the rest Jacobian matrix $\mathbf{J}_0$ is computed in the same matter but using the node positions in reference-space instead of world-space. When the world configuration is equal to the rest configuration (i.e. no deformation), the deformation gradient $\mathbf{F}$ is simply an identity matrix.

3.2.5 Strain

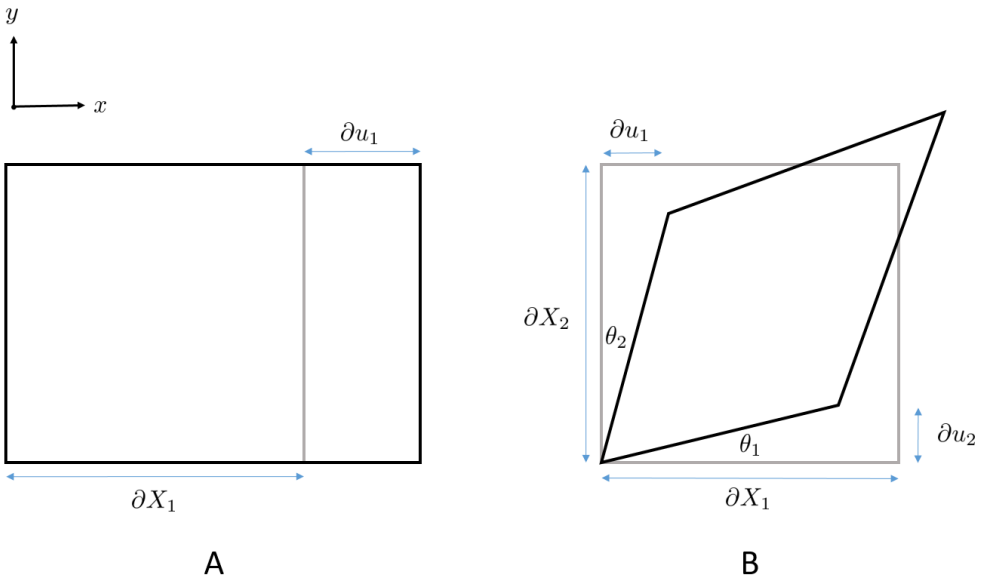


Figure 3.2: Measuring Infinitesimal Strain. (A) depicts a cube from a 2D view with an undeformed length ∂X_1 being stretched along the x-axis by ∂u_1. (B) shows the same cube being sheared instead. The angles θ_1 and θ_2 are used to quantify the amount of shearing experienced by the cube relative to its undeformed rest configuration.

While the deformation gradient represents a measurement of deformation, it can be refined into a displacement gradient tensor. The tensor measures, in gradient form, the displacement between the world-space and reference-space coordinates of the object. There are various models of the displacement gradient tensor.

For small deformations, the displacement gradient tensor is calculated as an *infinitesimal strain tensor*. This strain tensor, also known as the *Cauchy strain*, makes the assumption that deformations are small enough such that strain

can be approximated geometrically in terms of length differences. It is useful to derive the Cauchy strain in order to understand its definition and reveal how it can be computed directly from the deformation gradient [17].

The Cauchy strain $\boldsymbol{\varepsilon}$ is a symmetric matrix with 3 diagonal stretching components and 3 unique off-diagonal shearing components. Let $\mathbf{u}$ denote a 3D displacement vector and let $\mathbf{X}$ denote a 3D position in reference-space. Both $\mathbf{u}$ and $\mathbf{X}$ are in global coordinate space and thus share the common Euclidean coordinate system. Suppose we are given an infinitesimal cube subjected to stretching deformation, as shown in Figure 3.2.A. The amount of stretching along the x-axis that is experienced by the cube relative to its undeformed length across the x-axis is written as:

$$\varepsilon_{11} = \frac{\partial u_1}{\partial X_1}.$$

Similar definitions can be arrived for the y-axis and z-axis:

$$\varepsilon_{22} = \frac{\partial u_2}{\partial X_2} \qquad\qquad \varepsilon_{33} = \frac{\partial u_3}{\partial X_3}.$$

Now suppose that the infinitesimal cube is subjected to shearing deformation, as shown in Figure 3.2.B. The angular deformation that is experienced by the sheared cube along the x-y plane is defined by angles formed between the undeformed and sheared cube:

$$\gamma_{12} = \theta_1 + \theta_2.$$

For small angles, the angles are approximately equal to their tangent, which in turn, can be defined in terms of lengths using trigonometry:

$$\theta_1 \approx \tan\theta_1 = \frac{\partial u_2}{\partial X_1} \qquad\qquad \theta_2 \approx \tan\theta_2 = \frac{\partial u_1}{\partial X_2}.$$

Therefore, we can rewrite the angular deformation γ_{12} in terms of lengths:

$$\gamma_{12} = \theta_1 + \theta_2 = \frac{\partial u_2}{\partial X_1} + \frac{\partial u_1}{\partial X_2}.$$

By definition, the Cauchy strain shearing component ε_{12} is equivalent to one-half of its corresponding angular deformation γ_{12}:

$$\varepsilon_{12} = \frac{1}{2}\gamma_{12} = \frac{1}{2}\left(\frac{\partial u_2}{\partial X_1} + \frac{\partial u_1}{\partial X_2}\right).$$

We can repeat this definition for the other shearing components of the Cauchy strain:

$$\varepsilon_{13} = \frac{1}{2}\left(\frac{\partial u_1}{\partial X_3} + \frac{\partial u_3}{\partial X_1}\right) \qquad \varepsilon_{31} = \frac{1}{2}\left(\frac{\partial u_3}{\partial X_1} + \frac{\partial u_1}{\partial X_3}\right) \qquad \varepsilon_{23} = \frac{1}{2}\left(\frac{\partial u_2}{\partial X_3} + \frac{\partial u_3}{\partial X_2}\right) \qquad \varepsilon_{32} = \frac{1}{2}\left(\frac{\partial u_3}{\partial X_2} + \frac{\partial u_2}{\partial X_3}\right).$$

Altogether, the 3×3 Cauchy strain $\boldsymbol{\varepsilon}$ can be compactly written as

$$\boldsymbol{\varepsilon} = \frac{1}{2}(\nabla\mathbf{u} + \nabla\mathbf{u}^{\mathsf{T}}) \tag{3.3}$$

where

$$\varepsilon_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial X_j} + \frac{\partial u_j}{\partial X_i}\right).$$

The deformation gradient can be thought as, in gradient form, the displacement from the rest configuration. This notion is expressed as

$$\mathbf{F} = \nabla \mathbf{u} + \mathbf{I}$$

which can be rearranged:

$$\nabla \mathbf{u} = \mathbf{F} - \mathbf{I}. \tag{3.4}$$

Using equations (3.3) and (3.4), the Cauchy strain can be computed from the deformation gradient:

$$\begin{aligned} \varepsilon &= \frac{1}{2}(\nabla \mathbf{u} + \nabla \mathbf{u}^{\intercal}) \\ &= \frac{1}{2}(\mathbf{F} - \mathbf{I} + (\mathbf{F} - \mathbf{I})^{\intercal}) \\ &= \frac{1}{2}(\mathbf{F} - \mathbf{I} + \mathbf{F}^{\intercal} - \mathbf{I}) \\ &= \frac{1}{2}(\mathbf{F} + \mathbf{F}^{\intercal}) - \mathbf{I}. \end{aligned} \tag{3.5}$$

3.2.6 Stress

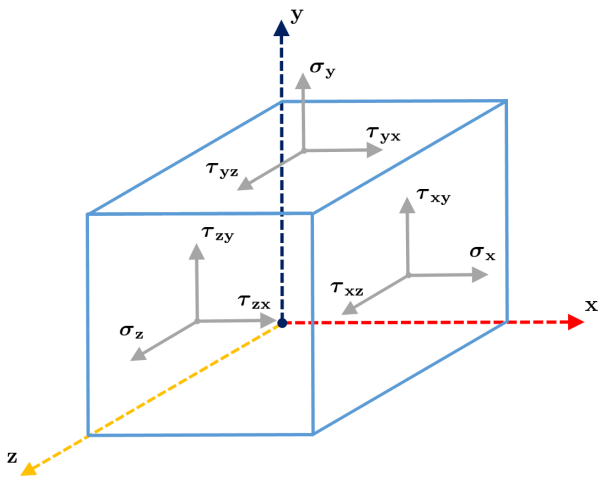


Figure 3.3: Stress Forces on an Infinitesimal Body. The normal stress forces are labelled σ_x, σ_y, and σ_z. The shear stress forces perpendicular to σ_x are labelled τ_{xy} and τ_{xz}. Those perpendicular to σ_y are τ_{yx} and τ_{yz}, and those perpendicular to σ_z are τ_{zx} and τ_{zy}

The Cauchy strain tensor can be transformed into the *Cauchy stress tensor*, which describes the elastic force gradient at a deformed infinitesimal body:

$$\boldsymbol{\sigma} = \frac{\partial \mathbf{f}}{\partial \mathbf{X}} = \begin{bmatrix} \sigma_x & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \sigma_y & \tau_{yz} \\ \tau_{zx} & \tau_{xy} & \sigma_z \end{bmatrix}.$$

Each of the 9 elements in the stress tensor corresponds to a particular force direction. See Figure 3.3 which layout the force directions. The diagonal elements are the orthogonal normal stress forces while the off-diagonal elements

represent the shear forces. Much like the Cauchy strain tensor, the Cauchy stress tensor is a symmetric matrix. Only 3 of the shear forces ($\tau_{xy}, \tau_{yz}, \tau_{zx}$) need to be accounted: $\tau_{yx} = \tau_{xy}, \tau_{zy} = \tau_{yz}$, and $\tau_{zy} = \tau_{xz}$.

The stress tensor incorporates the properties of the material that the object is made of. Material properties can include a stiffness parameter (e.g. Young's modulus E), a ratio between the expansion that occurs perpendicularly to the direction of compression (e.g. Poisson's ratio ν), and other parameters that depend on the material model used. The simplest material model is the linear material model, which assumes that stretching and compression of the object obey Hooke's law.

Assuming the linear material model and using Einstein notation, the stress tensor is calculated as

$$\sigma_{ij} = C_{ab}\,\varepsilon_{kl}$$

where $a = 2i + k$, $b = 2j + l$, and $\mathbf{C}$ is the *material elasticity tensor* that linearly maps the strain to stress while accounting for the material properties, particularly the Young's modulus E and Poisson's ratio ν:

$$\mathbf{C} = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1-\nu & \nu & 0 & 0 & 0 & 0 \\ \nu & 1-\nu & \nu & 0 & 0 & 0 \\ \nu & \nu & 1-\nu & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1-2\nu}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1-2\nu}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1-2\nu}{2} \end{bmatrix}.$$

3.2.7 Elastic Force

The stress tensor $\boldsymbol{\sigma}$ can be integrated over the element volume to yield the element's contribution towards the force of a given node i. This integration is formulated as

$$\mathbf{f}_i^e = \int_v \mathbf{B}_i^\intercal \boldsymbol{\sigma}\, \mathrm{dv}$$

where v is the volume of element e, and $\mathbf{B}_i$ is an extrapolation matrix that provides the mapping (in gradient form) between node i and any arbitrary point within the volume. Evaluating the integral analytically is difficult and impractical. Therefore, it is approximated using the quadrature approach by evaluating the integrand over the q discrete integration points of element e:

$$\mathbf{f}_i^e \approx \sum_k^q \mathbf{B}_{i,k}^\intercal \boldsymbol{\sigma}_k\, \mathrm{dv}_k. \tag{3.6}$$

In this equation, the extrapolation matrix $\mathbf{B}_{i,k}$ is constructed using node i's shape function derivatives, which are evaluated using the local coordinates of integration point k:

$$\mathbf{B}_i = \begin{bmatrix} \frac{\partial \mathrm{N}_i}{\partial \xi_1} & 0 & 0 \\ 0 & \frac{\partial \mathrm{N}_i}{\partial \xi_2} & 0 \\ 0 & 0 & \frac{\partial \mathrm{N}_i}{\partial \xi_3} \\ \frac{\partial \mathrm{N}_i}{\partial \xi_2} & \frac{\partial \mathrm{N}_i}{\partial \xi_1} & 0 \\ 0 & \frac{\partial \mathrm{N}_i}{\partial \xi_3} & \frac{\partial \mathrm{N}_i}{\partial \xi_2} \\ \frac{\partial \mathrm{N}_i}{\partial \xi_3} & 0 & \frac{\partial \mathrm{N}_i}{\partial \xi_3} \end{bmatrix}.$$

The sparsity and arrangement of the extrapolation matrix, as shown above, is intended in order to multiply the components of the symmetrical stress tensor with their corresponding extrapolation matrix component. For example, both $\boldsymbol{\sigma}_{\mathbf{x}}$ and $\boldsymbol{\tau}_{\mathbf{yx}}$ are with respect to the change in local coordinate ξ_1 and therefore should be be multiplied with the $\mathbf{B}_{i,k}^T$ components containing ξ_1. The symmetrical stress tensor in equation (3.6) is inputted as a column 6-vector:

$$\boldsymbol{\sigma}_k = \begin{bmatrix} \sigma_{x,k} \\ \sigma_{y,k} \\ \sigma_{z,k} \\ \tau_{xy,k} \\ \tau_{yx,k} \\ \tau_{xz,k} \end{bmatrix}.$$

The dv_k factor denotes the volume associated with integration point k and is calculated as

$$\mathrm{dv}_k = \det(\mathbf{J}_k)\, w_k.$$

3.2.8 Stiffness Matrix

A *stiffness matrix* $\mathbf{K}$ can be created in order to map the nodal forces into nodal displacements, while taking in account of the material stiffness. The stiffness matrix can be understood using a simple example [29]. Suppose an element has 4 nodes and each node has 3 degrees of freedom (i.e. x, y, z), which totals to 12 degrees of freedom for the element. The stiffness matrix associated with the element is consequently constructed as a 12×12 matrix. $\mathbf{K}_{ij}$ is the force exerted on the i-th degree of freedom when all the other degrees of freedom are held zero except for the j-th degree of freedom, which is allowed to be displaced by one unit.

For consistency with how the elastic forces are indexed, we index $\mathbf{K}$ on a nodal basis. Therefore in the subsequent sections, $\mathbf{K}_{ij}$ will refer to the 3×3 matrix that describes the force and displacement relationship between node i and node j. When referring to a given element e, we calculate the stiffness matrix as follows:

$$\begin{aligned} \mathbf{K}_{ij}^e &= \int_v \mathbf{B}_i^T \mathbf{C} \mathbf{B}_j \, \mathrm{dv} \\ &\approx \sum_k^q \mathbf{B}_{i,k}^T \mathbf{C} \mathbf{B}_{j,k} \, \mathrm{dv}_k \end{aligned} \tag{3.7}$$

where node i, node j, and the set of the integration points q are belonging to element e; and $\mathbf{B}_{i,k}$, to re-iterate, is the nodal extrapolation matrix $\mathbf{B}_i$ that is evaluated using the local coordinates of integration point k.

3.2.9 Assembly

The nodal forces and stiffness matrices, which were evaluated by integrating across the elements individually, can be assembled into a *global force vector* $\mathbf{f}'$ and *global stiffness matrix* $\mathbf{K}'$ in order to solve for the nodal velocities all at once. Each node is assigned a *global node index* that is unique amongst the other nodes in the FEM system. For example, the i-th node of element e can be assigned a global node index set to $e \cdot i$. Using global nodal indices, the global force vector $\mathbf{f}'$ is defined as an array of 3D nodal forces:

$$\mathbf{f}'_i = \sum_e^E \mathbf{f}_i^e$$

where i is the global node index, E is the set of elements adjacent to node i, and $\mathbf{f}_i^e$, which was defined earlier in equation (3.6), is the 3D force of node i that is contributed by element e.

Alongside with the global force vector, the global stiffness matrix $\mathbf{K}'$ is assembled as a sparse block matrix:

$$\mathbf{K}'_{ij} = \sum_e^E \mathbf{K}_{ij}^e$$

where i and j are global node indices, E is the set of elements that contain both node i and j, and $\mathbf{K}_{ij}^e$ is the 3×3 stiffness matrix that is contributed by element e and defined in equation (3.7). If E is an empty set (i.e. there does not exists an element containing node i and j), then $\mathbf{K}'_{ij}$ is simply set to a 3×3 zero matrix.

3.2.10 Solve

The global force vector can undergo a separate integration procedure to retrieve the nodal velocities that advance the deformed object into the next time step. The integration is handled by the implicit integrator (i.e. backward Euler method) provided by ArtiSynth [26]. The implicit integrator uses Newton's 2nd law of motion as a starting premise:

$$\mathbf{M}\dot{\mathbf{v}} = \mathbf{f} \tag{3.8}$$

where $\mathbf{M}$ is a diagonal matrix of nodal masses, $\dot{\mathbf{v}}$ is a vector of nodal accelerations, and $\mathbf{f}$ corresponds to the global force vector.

We use equation (3.8) to advance the world-space nodal positions $\mathbf{x}$ and velocities $\mathbf{v}$ forward in time. For stability reasons, the backward Euler scheme is used to update $\mathbf{v}$ and $\mathbf{x}$ at each time step k:

$$\begin{aligned} \mathbf{M}\mathbf{v}^{k+1} &= \mathbf{M}\mathbf{v}^k + h\mathbf{f}^{k+1} \\ \mathbf{x}^{k+1} &= \mathbf{x}^k + h\mathbf{v}^{k+1} \end{aligned} \tag{3.9}$$

where h is the integration time step.

Because $\mathbf{f}^{k+1}$ is not known, we approximate it with a Taylor expansion:

$$\mathbf{f}^{k+1} \approx \mathbf{f}^k + \frac{\partial \mathbf{f}}{\partial \mathbf{v}}\Delta\mathbf{v} + \frac{\partial \mathbf{f}}{\partial \mathbf{x}}\Delta\mathbf{x}$$

where the derivatives $\partial\mathbf{f}/\partial\mathbf{v}$ and $\partial\mathbf{f}/\partial\mathbf{x}$ correspond to the optional damping matrix and global stiffness matrix of the FEM system. In the context of equation (3.9), we have

$$\Delta\mathbf{v} \equiv \mathbf{u}^{k+1} - \mathbf{v}^k \qquad\qquad \Delta\mathbf{x} \equiv \mathbf{x}^{k+1} - \mathbf{x}^k = h\mathbf{v}^{k+1}.$$

Putting this together yields a sparse linear system that is often referred as the FEM system:

$$\left(\mathbf{M} - h\frac{\partial\mathbf{f}}{\partial\mathbf{v}} - h^2\frac{\partial\mathbf{f}}{\partial\mathbf{x}}\right)\mathbf{v}^{k+1} = \mathbf{M}\mathbf{v}^k - h\frac{\partial\mathbf{f}}{\partial\mathbf{v}}\mathbf{v}^k + h\mathbf{f}^k \tag{3.10}$$

which is solved for $\mathbf{v}^{k+1}$ at each time step using a direct sparse linear solver. $\mathbf{v}^{k+1}$ is then used to compute $\mathbf{x}^{k+1}$ according to equation (3.9).

3.3 Solid-Shell Element

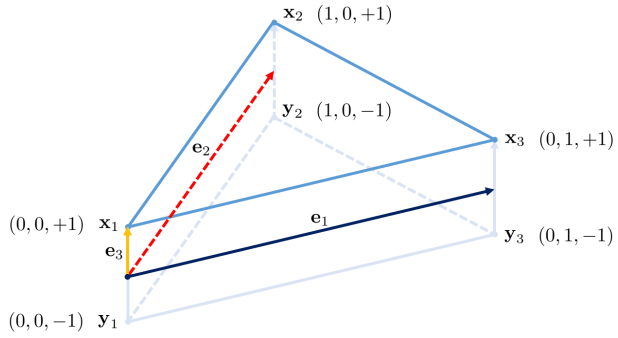


Figure 3.4: Schematic Diagram of a Triangular Solid-Shell using a Front and Back Node Implementation. The element is comprised of three front nodes ($\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$) and three virtual back nodes ($\mathbf{y}_1$, $\mathbf{y}_2$, $\mathbf{y}_3$). The virtual back node position can alternatively be specified as $\mathbf{y}_a = \mathbf{x}_a - \mathbf{d}_a$ where $\mathbf{d}_a$ is the director vector that extends from the back node $\mathbf{y}_a$ to the front node $\mathbf{x}_a$. The axes of the local coordinates are labelled as e_1, e_2, and e_3. The local coordinates (ξ_1, ξ_2, ξ_3) of each node is provided.

The surface of biological tissue can be comprised and modelled using triangular solid-shell elements. The solid-shell is a specialization of the volumetric wedge element where the thickness is linearized, and that stretching and bending are respectively represented by the translation and rotation of the director vectors. There are two possible implementations for the triangular solid-shell element. The first implementation uses 3 nodes that reside on the solid-shell's mid-surface where each node specifies the midpoint of its associated director. This implementation is previously shown in Figure 2.1.B. The second implementation uses 3 front surface nodes and 3 virtual back surface nodes where each front-back node pair represents a single director, as shown in Figure 3.4. The front node and back node of a given director can be imagined as encoding the director's position and orientation respectively. The 3 back nodes are virtual in the sense that they are not visible nor exposed to external forces. We use the second implementation of the solid-shell to handle plasticity more easily and to accommodate potential volumetric element attachments [27].

Notions of integration points, deformation gradient, strain, and stress in volumetric elements still apply to solid-shell elements. Solid-shells differ in how the deformation gradient, force, and stiffness are calculated due to its

linearized thickness and director-based design. In addition, compared to the simple tetrahedral element example, a solid-shell has 3 integration points distributed along each layer: top surface, virtual mid-surface, and virtual bottom surface. The elastic forces of the front and back nodes are obtained by numerically integrating the stress tensors across the 9 integration points. As well, the stiffness between an adjacent node pair is obtained by numerically integrating the material elasticity tensor. The exact formulation of the deformation gradient, force, and stiffness for solid-shells follow closely that of FEBio [27, Section 4.2].

Similar to thin-shells, a solid-shell model can be treated as a surface mesh (i.e. 2D manifold). The vertices, edges, and faces of the surface mesh correspond to the front nodes, front edges, and front triangles of the solid-shells. Virtual back nodes can still be accessed by referring to their corresponding front nodes. Altogether, this 2D abstraction simplifies the diffusion, remeshing, and collision detection process.

3.3.1 Construction

Let $\boldsymbol{\xi} = (\xi_1, \xi_2, \xi_3)$ denote the local coordinates of a point within a solid-shell element. The space that the local coordinates reside are shown in Figure 3.4. Coordinate ξ_3 is defined such that the front nodes and back nodes correspond to $\xi_3 = 1$ and $\xi_3 = -1$ respectively. The *2D shape functions* of the solid-shell element are defined as

$$\mathrm{N}_1(\xi) = 1 - \xi_1 - \xi_2 \qquad \mathrm{N}_2(\xi) = \xi_1 \qquad \mathrm{N}_3(\xi) = \xi_2.$$

These 2D shape functions ignore the local thickness coordinate ξ_3. Using these 2D shape functions and ξ_3 independently, the local coordinates (ξ_1, ξ_2, ξ_3) can be mapped into global coordinates $\mathbf{x}(\xi_1, \xi_2, \xi_3)$ when given the global position of each node:

$$\begin{aligned}\mathbf{x}(\xi_1, \xi_2, \xi_3) &= \sum_a^3 \mathrm{N}_a(\boldsymbol{\xi}) \left(\frac{1+\xi_3}{2}\mathbf{x}_a + \frac{1-\xi_3}{2}\mathbf{y}_a \right) \\ &= \sum_a^3 \mathrm{N}_a(\boldsymbol{\xi}) \left(\mathbf{x}_a - \frac{1+\xi_3}{2}\mathbf{d}_a \right)\end{aligned}$$

where $\mathbf{x}_a$ and $\mathbf{y}_a$ refers to the global position of the a-th front node and a-th back node respectively, $\mathbf{d}_a$ is the a-th director vector which stems between its front and back node (i.e. $\mathbf{d}_a = \mathbf{x}_a - \mathbf{y}_a$), and N_a is the a-th 2D shape function.

The local coordinates of the solid-shell element's integration points can be found in Appendix A.1. The solid-shell element has 9 integration points that are distributed across its volume. The extra integration points improve the approximation of the quadrature procedure while allowing flexibly in localizing the strain and stress in the element.

For solid-shell elements, the Jacobian matrix is composed of three column vectors:

$$\mathbf{g}_1 = \frac{\partial \mathbf{x}}{\partial \xi_1} \qquad \mathbf{g}_2 = \frac{\partial \mathbf{x}}{\partial \xi_2} \qquad \mathbf{g}_3 = \frac{\partial \mathbf{x}}{\partial \xi_3}. \tag{3.11}$$

These three vectors can be regarded as *covariant basis vectors*. Each covariant basis vector has a *contravariant basis vector* counterpart, which is written with a superscript instead: $\mathbf{g}^k$. The contravariant basis vectors is best understood as the vectors that comprise the inverse of the Jacobian matrix:

$$\mathbf{g}^1 = \frac{\partial \boldsymbol{\xi}}{\partial \mathbf{x}_1} \qquad \mathbf{g}^2 = \frac{\partial \boldsymbol{\xi}}{\partial \mathbf{x}_2} \qquad \mathbf{g}^3 = \frac{\partial \boldsymbol{\xi}}{\partial \mathbf{x}_3}$$

The formula for the covariant basis vectors can be found in the Appendix A.2, which also describes the covariant and contravariant basis vectors in more detail.

In terms of these basis vectors, the 3×3 deformation gradient is constructed as

$$\mathbf{F} = \frac{\partial \mathbf{x}}{\partial \mathbf{X}} = \frac{\partial \mathbf{x}}{\partial \boldsymbol{\xi}} \frac{\partial \boldsymbol{\xi}}{\partial \mathbf{X}} = \mathbf{J}\mathbf{J}_0^{-1} = \begin{bmatrix} \mathbf{g}^1 & \mathbf{g}^2 & \mathbf{g}^3 \end{bmatrix} \begin{bmatrix} \mathbf{G}_1 & \mathbf{G}_2 & \mathbf{G}_3 \end{bmatrix}.$$

The deformation gradient can then be used to compute the strain and stress tensors in the same matter as for volumetric elements. See Sections 3.2.5 and 3.2.6.

3.3.2 Elastic Force and Stiffness Matrix

For the elastic force and stiffness matrix, the solid-shell element has its own unique formulation. Let a be a node from a solid-shell element's set of front nodes $(1,2,3)$ and back nodes $(4,5,6)$; $a \in \{1,2,3,4,5,6\}$. Also let $\mathrm{s}(a)$ denote the index of node a relative to the face it belongs to:

$$\mathrm{s}(a) = \begin{cases} a & \text{if } a \in \{1,2,3\} \\ a-3 & \text{otherwise} \end{cases}. \tag{3.12}$$

The elastic force for node a, calculated over the q integration points of the element, is

$$\mathbf{f}_a \approx \sum_k^q \left(\boldsymbol{\sigma}_k \cdot \mathrm{grad} \left(\frac{1 + \mathrm{face}(a)\,\xi_{3,k}}{2} \mathrm{N}_{\mathrm{s}(a)} \right) \right) \mathrm{dv}_k \tag{3.13}$$

where each integration point k has its own set of local coordinates $(\xi_{1,k}, \xi_{2,k}, \xi_{3,k})$; $\boldsymbol{\sigma}_k$ is the 3×3 stress tensor computed at integration point k; and $\mathrm{face}(i)$ is 1 or -1, depending if node a belongs to the front or back face respectively. The formula for the gradient factor can be found in the Appendix A.3.

Along with the elastic shell forces, the stiffness matrix can be computed. Let a and b refer to two nodes (identical or different) for a given element. The 3×3 stiffness matrix between nodes a and b is calculated over the q integration points of the element using

$$\mathbf{K}_{ab} \approx \sum_k^q \left(\mathrm{grad} \left(\frac{1 + \mathrm{face}(a)\,\xi_{3,k}}{2} \mathrm{N}_{s(a)} \right) \cdot \mathbf{C} \cdot \mathrm{grad} \left(\frac{1 + \mathrm{face}(b)\,\xi_{3,k}}{2} \mathrm{N}_{s(b)} \right) \right) \mathrm{dv}_k \tag{3.14}$$

where $\mathbf{C}$ is the material elasticity tensor.

3.4 Diffusion

The chemicals that initiate growth can be simulated to diffuse across a surface mesh by solving the heat equation:

$$\dot{u} = \Delta u(x,t)$$

where Δ is the Laplace operator and $u(x,t)$ is the concentration of an arbitrary substance at position x and time t. Solving this heat flow expression over some period of time would yield the change in concentration at each fixed position x. Section 2.8 from earlier provides a high-level overview of the solve process.

The mass matrix $\mathbf{A}$ and discrete cotangent Laplacian matrix $\mathbf{L}$ that are required to solve the heat equation are simple to calculate. Suppose that a triangular mesh containing $|V|$ vertices is given, representing the front surface of the solid-shell model. The mass matrix $\mathbf{A}$ is a $|V| \times |V|$ diagonal matrix where the i-th diagonal entry contains one-third of the summed area of triangles that are incident to vertex i. The cotangent Laplacian matrix $\mathbf{L}$ is a $|V| \times |V|$ matrix and is constructed as follows:

$$(\mathbf{L})_{ij} \approx \begin{cases} \frac{1}{2}[\cot\alpha + \cot\beta] & \text{if edge } i\text{-}j \text{ exists} \\ -\sum_n (\mathbf{L})_{in} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \tag{3.15}$$

where $\sum_n$ is the summation over each neighboring vertex n that shares an edge with vertex i; and α and β are the angles that are on opposite sides of edge i-j, as shown in Figure 3.5.

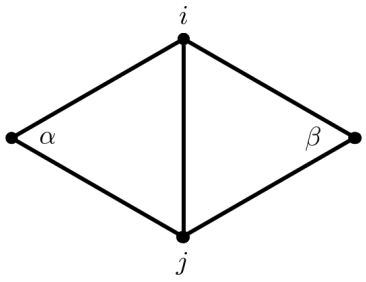


Figure 3.5: Angles between an Edge for Constructing the Cotangent Laplacian Matrix. The edge is comprised of vertices i and j, with angles α and β on opposite sides of the edge.

When given the initial concentration at each vertex i, their final concentrations after a small time step h are determined by solving for $\mathbf{u}_1$ in the linear system [39]:

$$(\mathbf{A} - h\,\mathrm{D}\,\mathbf{L})\,\mathbf{u}_1 = \mathbf{A}\mathbf{u}_0$$

where D is the diffusion coefficient and $\mathbf{u}_0$ is the $|V|$-sized vector containing the initial concentration at each vertex. Solving for $\mathbf{u}_1$ can be done using any direct sparse linear solver.

3.5 Growth Tensor

As morphogen diffuse throughout the front nodes, each front node will convert a fraction of its morphogen into a growth tensor. The growth tensor itself is a specification of the amount of growth to occur and is formulated as a function of morphogen concentration [22]. The growth tensor is formulated in such a way to support anisotropic growth by accepting multiple types of morphogen where each type corresponds to a particular growth direction. We extend this definition where each morphogen type is either of stretching subtype or bending subtype.

Assuming stretching subtype for now, the 3 types of morphogen that independently diffuse throughout the model are PAR, PER, and NOR. As these morphogens diffuse from node to node, at every time step, each node subtracts a

fixed percentage of its morphogen to be used for growth. The amount of morphogen subtracted are recorded on an element basis: each element has a 3×3 *element growth tensor* $\mathbf{E}$ where $\mathbf{E}_{ij}$ refers to the amount of morphogen of type j absorbed by the element's i-th front node.

The element growth tensor $\mathbf{E}$ is rotated to align with the element's 3×3 growth frame. The growth frame is defined as a rotation matrix:

$$\mathbf{Z} = [\mathbf{d}_{PAR}, \mathbf{d}_{PER}, \mathbf{d}_{NOR}]$$

where $\mathbf{d}_{PAR}$, $\mathbf{d}_{PER}$, and $\mathbf{d}_{NOR}$ are orthogonal unit vectors that represent the growth directions of the morphogen types. $\mathbf{d}_{PAR}$ and $\mathbf{d}_{PER}$ lie in the plane of the element while $\mathbf{d}_{NOR}$ is parallel to its normal. When growing with solid-shell elements, NOR morphogen is disabled unless the element thickness needs to gradually expand over time. With growth frame $\mathbf{Z}$, its respective element growth tensor $\mathbf{E}$ is rotated into $\mathbf{E}'$:

$$\mathbf{E}' = \mathbf{Z}\mathbf{E}\mathbf{Z}^\intercal.$$

Each element has an *integration growth tensor* $\mathbf{G}$, which is derived from the rotated element growth tensor $\mathbf{E}'$. The intention is to transform the rotated element growth tensor such that the number of rows corresponds to the number of integration points instead of the number of front nodes. In other words, the amount of morphogen absorbed at each integration point is to be inferred from the element's front nodes. This transformation can be done using another extrapolation matrix. Specifically, we use a $3 \times q$ matrix $\mathbf{M}$ where $\mathbf{M}_{ni}$ is the solid-shell's n-th 2D shape function that is evaluated using the local coordinates of the i-th integration point. Recall that these 2D shape functions are invariant to depth (i.e. 3rd local coordinate); integration points that have the same 1st and 2nd local coordinates will receive the same extrapolated morphogen absorbed.

This extrapolation matrix will only change when the morphogen is of bending subtype. The only difference is that when bending morphogen is being extrapolated, the extrapolation matrix $\mathbf{M}$ is set to zero except for the first 3 columns. In effect, this only allows the 3 integration points that reside along the virtual bottom surface to receive the extrapolated morphogen. That way, only the virtual bottom nodes, which handle rotation dynamics, will subsequently be pushed apart, causing upward bending to occur. It is worth noting that negative concentrations can be introduced to shrink the virtual bottom surface instead, which is particularly useful to induce downward bending.

Regardless how the extrapolation matrix $\mathbf{M}$ is set, it is multiplied with the rotated element growth tensor $\mathbf{E}'$ to yield the integration growth tensor $\mathbf{G}$:

$$\mathbf{G}^\intercal = \mathbf{E}'^\intercal \mathbf{M}.$$

The $q \times 3$ integration growth tensor $\mathbf{G}$ specifies the amount and direction of growth to occur at each integration point of the element. Specifically, the i-th row of the integration growth tensor is regarded as the 3 diagonal entries of the 3×3 *plastic strain* tensor $\boldsymbol{\varepsilon}_{i,\text{plastic}}$ with off-diagonal entries set to zero. In the next section, a method is described to convert the plastic strain into forces that permanently deform the model.

3.6 Plastic Forces

In order to induce plastic deformations, the *plastic forces*, which are forces associated with the plastic strain, need to be derived. The Cauchy stress tensor is first computed using the plastic strain:

$$\boldsymbol{\sigma}_{k,\text{plastic}} = \mathbf{C} \cdot \boldsymbol{\varepsilon}_{k,\text{plastic}}$$

where $\mathbf{C}$ is the linear material elasticity tensor. The plastic forces are then formulated exactly as the elastic forces of the solid-shell using equation (3.13) except that the stress tensor $\boldsymbol{\sigma}$ is substituted with $\boldsymbol{\sigma}_{k,\text{plastic}}$. When the plastic forces have been computed, if desired, they can be summed with external plastic forces to allow arbitrary plastic forces.

3.7 Plastic Embedding

The key idea to plastic embedding is that the plastic forces are applied to the rest configuration of the model [49]. The plastic forces are substituted into $\mathbf{f}^k$ of equation (3.10) to yield the corresponding velocities $\mathbf{v}^{k+1}$. The solved velocities $\mathbf{v}^{k+1}$ are then used to displace the rest configuration from time step k to $k+1$:

$$\mathbf{X}^{k+1} = \mathbf{X}^k + h\mathbf{v}^{k+1}$$

where h is the integration time step and $\mathbf{X}$ is the node positions of the rest configuration. By advancing the rest configuration, the object's stress-free shape becomes modified permanently, thus simulating plastic deformation.

When the rest configuration advances at each time step, the plastic strain tensor at each integration point decrements accordingly to reflect the remaining amount of plastic deformation to occur. Using small residual strain assumptions, the decrement procedure is formulated as

$$\boldsymbol{\varepsilon}^{k+1}_{i,\text{plastic}} = \hat{\boldsymbol{\varepsilon}}^{k}_{i,\text{plastic}} - (\hat{\mathbf{R}}_i - \mathbf{I}).$$

The hat symbol $\hat{}$ indicates that the symmetrical factor of the associated tensor should be used (see Appendix A.4 regarding decomposing a tensor into a symmetrical and rotational factor). In other words, we discard the rotational factor since they are unimportant when dealing with plasticity [21]. $\boldsymbol{\varepsilon}^{k}_{i,\text{plastic}}$ is the plastic strain tensor at the current time step k, specifying the expected amount of plastic deformation to occur at integration point i. $\mathbf{R}_i$ is the reference-space *incremental deformation gradient* [45], which is the deformation gradient of the updated rest configuration at time step $k+1$ with respect to the previous time step k, computed at integration point i:

$$\mathbf{R} = \frac{\partial \mathbf{X}^{k+1}}{\partial \mathbf{X}^k}.$$

$\boldsymbol{\varepsilon}^{k+1}_{k,\text{plastic}}$ is the remaining (i.e. residual) plastic strain that will be carried over to the next time step $k+1$.

Afterwards, the masses of the individual solid-shell elements are updated to sync with the grown rest configuration. The masses are directly related to the density and rest volume of the solid-shells.

We emphasize that the plastic embedding procedure does not require distinguishing between stretching and bending. This is due to the design of the solid-shell where stretching and bending can both commonly be described in terms of nodal displacement strain.

3.8 Adaptive Remeshing

Periodically during growth simulation, the model is remeshed to refine curved, dynamic, and compressed surfaces with enough resolution to permit surface buckling formation and retain geometric detail. At the same time, flat, idle, and relaxed surfaces are coarsen to save computation time. The implementation follows Narain et al. [32] and has been adapted for the growing solid-shells. Algorithm 2 provides an outline of the implementation.

Algorithm 2: Adaptive Remeshing Procedure

```
   /* Mesh Analysis */
 1 for each element e do
 2 |   M_crv ← Curvature metric of e (3.8.1.1)
 3 |   M_vel ← Velocity gradient metric of e (3.8.1.2)
 4 |   M_elastic ← Elastic strain metric of e (3.8.1.3)
 5 |   M_plastic ← Plastic strain metric of e (3.8.1.3)
 6 |   M̂_e ← Weighted sum of M_crv, M_vel, M_elastic, and M_plastic (3.8.1.5)
 7 |   Clamp M̂_e to respect the specified minimum and maximum edge lengths (3.8.1.6)
 8 end
 9 M_i ← Sizing field of vertex i by averaging M̂_e of adjacent faces (3.8.1.7)
10 Calculate the size of each edge i-j using M_i and M_j (3.8.1.8);
   /* Mesh Modification (3.8.2) */
11 while Exists an edge e with size greater than 1 do
12 |   Bisect edge e
13 |   while Exists a flippable edge f do
14 |   |   Flip edge f
15 |   end
16 end
17 while Exists a collapsable edge e do
18 |   Collapse edge e
19 |   while Exist a flippable edge f do
20 |   |   Flip edge f
21 |   end
22 end
   /* Mesh Post-Processing */
23 Resample the plastic strain (3.8.3)
```

3.8.1 Mesh Analysis

In the beginning of each remesh iteration, the *sizing field* of each mesh face is measured. The sizing field captures 4 metrics that judge the amount of refinement needed at the face. The 4 metrics are curvature $\mathbf{M}_{\text{crv}}$, velocity gradient $\mathbf{M}_{\text{vel}}$, elastic strain $\mathbf{M}_{\text{elastic}}$, and plastic strain $\mathbf{M}_{\text{plastic}}$.

3.8.1.1 Curvature Metric

The curvature metric $\mathbf{M}_{\text{crv}}$ is formulated to estimate the curvature between vertices i and j of a given face. The curvature is defined using the difference in vertex normals, which can be factored to involve a gradient:

$$\begin{aligned}\left\|\mathbf{n}_i - \mathbf{n}_j\right\|^2 &= \left\|\nabla\mathbf{n}\cdot\mathbf{u}_{ij}\right\|^2 \\ &= (\nabla\mathbf{n}\cdot\mathbf{u}_{ij})^\intercal(\nabla\mathbf{n}\cdot\mathbf{u}_{ij}) \\ &= \mathbf{u}_{ij}^\intercal(\nabla\mathbf{n}^\intercal\nabla\mathbf{n})\,\mathbf{u}_{ij}\end{aligned} \tag{3.16}$$

where $\nabla\mathbf{n}$ is the face's world-space vertex normal gradient with respect to the reference-space vertex normals, and $\mathbf{u}_{ij}$ is the displacement vector between vertices i and j in the *2D local reference-space*.

The 2D local reference-space is the space where the 3D reference-space would reside if it were projected onto the plane of the given face. The projection from 3D to 2D reference-space can be done using a basis matrix $\mathbf{U}$. Each face has its own basis matrix that is constructed as a 3×2 matrix where the 2 columns correspond to the normalized orthogonal vectors that are tangent to the face in reference-space. If $\mathbf{Q}$ is a vector quantity in 3D reference-space, such as a displacement vector, it can be projected into 2D local reference space by multiplying with the given basis matrix $\mathbf{U}$:

$$\mathbf{Q}\mathbf{U}.$$

If $\mathbf{Q}$ is a 3×3 gradient where $\mathbf{Q}=\partial\mathbf{f}/\partial\mathbf{x}$, both $\partial\mathbf{f}$ and $\partial\mathbf{x}$ can be projected into 2D local reference-space using the same basis matrix:

$$\mathbf{U}^\intercal\mathbf{Q}\mathbf{U}.$$

The reason for working in the 2D space is to simplify the mesh analysis. In particular, attributes, such as vertex normal, velocity, and strain, are only measured along the 2D plane of the face. The simplification is justified by the assumption that the thin object being modelled is always one-element thick; remeshing along the thickness is not required.

When the vertex normal gradient pair $\nabla\mathbf{n}$ from equation (3.16) is projected into the 2D local reference-space, it is regarded as the 2×2 curvature metric $\mathbf{M}_{\text{crv}}$:

$$\mathbf{M}_{\text{crv}} = \mathbf{U}^\intercal(\nabla\mathbf{n}^\intercal\nabla\mathbf{n})\,\mathbf{U}$$

The details for calculating the 3×3 $\nabla\mathbf{n}$ or any gradient across a face can be found in Appendix B.1.

3.8.1.2 Velocity Gradient Metric

The velocity difference between two vertices can be expressed and computed in a similar matter:

$$\left\|\mathbf{v}_i - \mathbf{v}_j\right\|^2 = \mathbf{u}_{ij}^\intercal(\nabla\mathbf{v}^\intercal\nabla\mathbf{v})\,\mathbf{u}_{ij}$$

where the projected gradient is taken to be the 2×2 velocity gradient metric $\mathbf{M}_{\text{vel}}$:

$$\mathbf{M}_{\text{vel}} = \mathbf{U}^\intercal(\nabla\mathbf{v}^\intercal\nabla\mathbf{v})\,\mathbf{U}.$$

$\nabla\mathbf{v}$ is the vertex velocity gradient with respect to the reference-space vertex positions.

3.8.1.3 Elastic and Plastic Strain Metric

Along with the curvature and velocity gradient metrics, the sizing field also dependent on the elastic strain metric $\mathbf{M}_{\text{elastic}}$ and plastic strain metric $\mathbf{M}_{\text{plastic}}$. Both metrics are measured on an element basis and are formulated as

$$\mathbf{M}_{\text{elastic}} = \mathbf{U}^{\intercal}\left(\mathbf{E}^{\intercal}\mathbf{E}\right)\mathbf{U} \qquad\qquad \mathbf{M}_{\text{plastic}} = \mathbf{U}^{\intercal}\left(\mathbf{P}^{\intercal}\mathbf{P}\right)\mathbf{U}$$

where

$$\mathbf{E} = \text{abs}(\text{sym}(\boldsymbol{\varepsilon}_{\text{elastic}})) \qquad\qquad \mathbf{P} = \text{abs}\left(\frac{1}{q}\sum_{i}\text{sym}\left(\boldsymbol{\varepsilon}_{i,\text{plastic}}\right)\right).$$

The elastic strain metric $\mathbf{M}_{\text{elastic}}$ is dependent on the Cauchy strain $\boldsymbol{\varepsilon}_{\text{elastic}}$, which is calculated at the *warping point*. The warping point is essentially a reserved integration point that resides in the center of the element, and is detailed in Appendix A.1.1. The Cauchy strain formulation can be found in equation (3.5).

The $\text{sym}(\cdot)$ operator, which accompanies $\boldsymbol{\varepsilon}_{\text{elastic}}$ and later $\boldsymbol{\varepsilon}_{i,\text{plastic}}$, denotes that the symmetrical factor of the given tensor should be used. See Appendix A.4 regarding decomposing a tensor into its symmetrical and rotational factors. The reason for discarding the rotational factor is to allow the elastic and plastic strain metrics to be subsequently summed together in a common rotation-free space when computing the sizing field.

The plastic strain metric $\mathbf{M}_{\text{plastic}}$ requires computing an average of the plastic strain. In particular, the symmetrical factor of the plastic strain is summed across the q integration points of the element, and then scaled by $1/q$. We use averaging instead of computing the plastic strain at the single warping point because the plastic strain tensors are already calculated at each integration point.

Lastly, the $\text{abs}(\cdot)$ operator returns the element-wise absolute of the given tensor. By taking the absolute, the measurement of strain becomes invariant to compression and stretching. In effect, buckling and refinement that follows are anticipated at areas where elastic compression or plastic stretching is present.

3.8.1.4 Omitted Metrics

We exclude two metrics from the sizing fields: buckling suppression $\mathbf{M}_{\text{buc}}$ and obstacle proximity $\mathbf{M}_{\text{obs}}$. Buckling suppression is particularly used to reduce surface buckling anticipation along the stretching strain direction when there is bending strain present in the perpendicular direction. This is useful for stiff materials, such as paper, but for soft tissue, the metric is not essential. The other omitted metric, obstacle proximity allows the mesh to anticipate collisions by preemptively refining areas that are near obstacles. This is useful for high impact velocity and fracture simulations. For growth simulation, collisions are slow enough such that the velocity metric alone can indicate needed refinement.

3.8.1.5 Sizing Field Summation

Altogether, the metrics undergo a weighted sum to evaluate the face's *sizing field*:

$$\hat{\mathbf{M}}_e = \frac{\mathbf{M}_{\text{crv}}}{\Delta n_{\max}^2} + \frac{\mathbf{M}_{\text{vel}}}{\Delta v_{\max}^2} + \frac{\mathbf{M}_{\text{elastic}}}{s_{\max}^2} + \frac{\mathbf{M}_{\text{plastic}}}{s_{\max}^2}.$$

The scalers Δn_{max}, Δv_{max}, and s_{max} are present to adjust the strength of each metric. For our simulations, we found setting the scalars to 0.2, 0.01, and 1.0 respectively to be sufficient.

3.8.1.6 Sizing Field Clamp

The eigenvalues of the sizing fields are clamped to correspond to the minimum and maximum edge lengths permitted and to constrain the minimum aspect ratio. This is performed by first factoring the sizing field using eigendecomposition to yield $\hat{\mathbf{M}} = \mathbf{Q}\hat{\mathbf{\Lambda}}\mathbf{Q}^{\intercal}$ where $\hat{\mathbf{\Lambda}} = \text{diag}(\hat{\lambda}_1, \hat{\lambda}_2)$. The eigenvalues $\hat{\lambda}_1$ and $\hat{\lambda}_2$ are clamped between $\left[l_{\text{max}}^{-2}, l_{\text{min}}^{-2}\right]$ where l_{min} and l_{max} refer to the specified minimum and maximum edge lengths respectively. Afterwards, the smaller clamped eigenvalue is set to α_{min}^2 times the larger value where α_{min} is the minimum aspect ratio targeted. We recommend setting α_{min} to 0.5, which prevents the remesher from breaking symmetry when given a symmetrical mesh.

3.8.1.7 Vertex Sizing Field

The sizing fields from the faces are then interpolated to the vertices. For a given vertex i, an area-weighted average is used to determine the sizing field contribution from each face:

$$\mathbf{M}_i = \frac{1}{\sum_f^{\text{F}_i} \text{A}_f} \sum_f^{\text{F}_i} \text{A}_f \hat{\mathbf{M}}_f$$

where F_i refers to the set of faces that are incident to vertex i while A_f and $\hat{\mathbf{M}}_f$ refer to the world-space area and sizing field of face f respectively.

3.8.1.8 Edge Size

The size of a given edge is calculated by first taking the average of its two vertices' sizing fields. The average is then transformed into a scalar value by multiplying with the 2D local reference-space displacement vector $\mathbf{u}_{ij}$:

$$s(i,j) = \sqrt{\mathbf{u}_{ij}^{\intercal} \left(\frac{\mathbf{M}_i + \mathbf{M}_j}{2} \right) \mathbf{u}_{ij}}$$

where i and j refer to the two vertices of the edge.

3.8.2 Mesh Modification

When the edge sizes have been calculated, the mesh topology can now be modified. First, the maximal independent set of edges that have a size greater than 1 are bisected. The bisect operation and along with other operations, are exemplified in Figure 3.6. The maximal independent set refers to the largest set of edges where no edge shares a common vertex with another. This set can be found greedily in linear time by looping through each edge where each edge is checked to see if it has a size greater than 1 and that none of its two vertices have been visited. If these two conditions are satisfied, the edge is added to the set. When the entire set has been found, each edge in the set is bisected. The new vertex that resides at the midpoint of each bisected edge has its attributes, namely the front and back node world-space positions, reference-space positions, world-space velocities, reference-space velocities, and morphogen

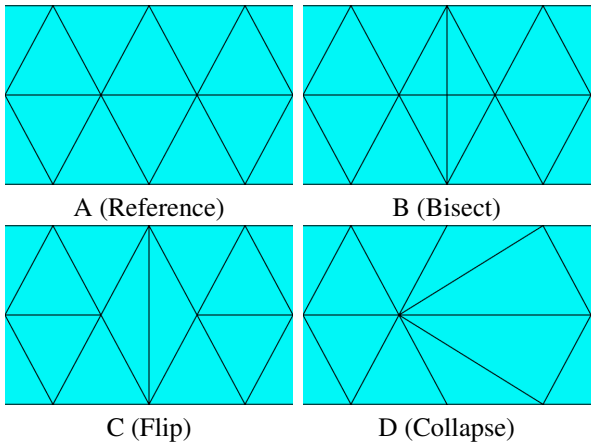


Figure 3.6: Remeshing Operations. Using the center edge in (A) as a starting reference, the effect of the three operations on the edge are shown in (B), (C), and (D) respectively. The bisect operation (B) splits the center edge while creating a new middle vertex that is connected to all four of its adjacent vertices. The flip operation (C) replaces the center edge with a new edge that is connected to the other two adjacent vertices. The collapse operation (D) collapses one of the center edge's vertices (right vertex in this case) into the other center edge vertex (left vertex in this case). If the mesh symmetry needs to be preserved, the collapse operation can be performed by collapsing the two vertices of the center edge together into a new midpoint vertex.

concentration, calculated using the average of its 2 parent vertices' attributes. When the maximal independent set of edges have been bisected, the whole edge bisecting procedure is repeated until no non-empty maximal independent set can be found.

Additionally, every time an edge is bisected, the entire mesh is checked for flippable edges and are flipped if so. Algorithmically, this process is performed by first finding the maximal independent set of flippable edges. An edge is considered flippable if the anisotropic-aware criteria holds true [31]:

$$(\mathbf{u}_{jk} \times \mathbf{u}_{ik})\,\mathbf{u}_{il}^{\mathsf{T}}\,\mathbf{M}_{\text{avg}}\,\mathbf{u}_{\text{jl}} + \mathbf{u}_{jk}^{\mathsf{T}}\,\mathbf{M}_{\text{avg}}\,\mathbf{u}_{ik}\,(\mathbf{u}_{il} \times \mathbf{u}_{jl}) < 0.$$

i and j refer to the vertices of the edge. k and l refer to the vertices of the new edge if the flip were to occur. $\mathbf{M}_{\text{avg}}$ denotes the average vertex sizing field computed from all four vertices i, j, k, and l. When all flippable edges in the maximal independent set have been flipped, the flipping procedure is repeated again until a non-empty maximal independent flippable edge set does not exist. The purpose of these edge flips are to improve the aspect ratio of the faces, similar to how Delaunay triangulation aims to maximize the minimum angles of each triangle.

After splitting and flipping the edges, edges are collapsed without introducing new edges of size greater than 1. Let F be the set of faces of the mesh. Within F, a collapsible edge is searched for. An edge is considered collapsible if it does not reside on the boundary of the mesh, will not produce an inverted or small aspect ratio face, nor produce an edge with a size exceeding $1-h$ where h is a hysteresis parameter set to 0.2 in order to avoid splitting and collapsing oscillations. If the edge is collapsible, the edge is removed without introducing any new vertex, as shown in Figure 3.6.D. However, if the mesh symmetry needs to be preserved, the edge can alternatively be removed by collapsing the two vertices of the edge into a new midpoint vertex. Once an edge is collapsed, F is updated to include the faces affected by the collapse. Similar to the bisection procedure, flippable edges in F are searched and

flipped after each collapse. The whole collapsing procedure is repeated until no edge can be collapsed nor flipped.

3.8.3 Plastic Strain Resampling

Once remeshing has finished, a custom post-processing step is executed where the plastic strain of each element is recalculated. In other words, before and after remeshing, the plastic strain across the mesh should be close to invariant in terms of its magnitude, direction, and locality. The resampling process is carried out using a nearest neighbor interpolation scheme. First, integration points of the old mesh are saved. For a given integration point of the new mesh, the 6 integration points of the old mesh that are nearest to the given integration point are searched. The search can be perform by traversing across neighboring triangles. Once the 6 points are found, their associated plastic strain matrices undergo an inverse distance weighting procedure to yield the plastic strain tensor $\boldsymbol{\varepsilon}_{\text{plastic},i}$ for the given integration point i of the new mesh:

$$\boldsymbol{\varepsilon}_{\text{plastic},i} = \frac{1}{\sum_j \text{w}(i,j)} \sum_j \text{w}(i,j)\, \boldsymbol{\varepsilon}_{\text{plastic},j}$$

$$\text{w}(i,j) = \frac{1}{\text{dist}(i,j)}$$

where j denotes one of the 6 integration points of the old mesh that are closest to integration point i of the new mesh, and $\text{dist}(i,j)$ is the distance between integration point i and j in world-space. Because plastic strain is minimized due to plastic embedding, numerical errors associated with repeated resampling are minimized as well.

3.9 Collision Handling

To prevent the growing models from intersecting, the simulations are coupled with a collision handling method that is largely based on the discrete and continuous collision handling scheme proposed by Bridson et al. [3], while leveraging on ArtiSynth's LCP solver to resolve detected collisions [26]. Algorithm 3 outlines the steps of the collision handling method, which are elaborated in the subsequent sections. Much like for remeshing, the solid-shell model is treated as a surface mesh where front nodes, front edges, and front surfaces are referred to as vertices, edges, and faces/triangles respectively. Collision handling can be configured to be performed entirely in the world-space or reference-space with the latter assuming that the world-space mirrors the reference-space. In all of our growth simulations, collision handling is performed in reference-space.

3.9.1 Discrete Collision Handling

In the first stage of collision handling, nearby *features*[2] are detected and repelled apart in attempt to enforce a minimum distance between nearby features. By maintaining a certain minimum distance between features, actual collisions that do occur can be resolved more quickly: the minimum distance imposed reduces the risk of secondary collisions occurring when the actual collisions are repelled apart.

[2] In context of collision handling, the term *feature* denotes either a vertex, edge, or triangle of a mesh.

Algorithm 3: Collision Handling Procedure

```
   /* Corresponding sections numbers are denoted in red */

   /* Physics time step */
1  x^k ← Node positions at time k (Free of penetrations)
2  v^{k+1} ← Solved node velocities at time k+1 (3.2.10)
3  h ← Integration time step
4  x^{k+1} ← x^k + h v^{k+1}                    // Penetrations temporarily ignored

   /* Gathered constraints at time k+1 */
5  C' ← ∅

   /* Stage 1: Repel features that are in close proximity */
6  Detect nearby features in x^k using discrete detection (3.9.1.1)
7  C ← Constraints of nearby feature pairs (3.9.1.2)
8  J ← Impulse velocities obtained from LCP solver and C (3.9.1.3)
9  x^{k+1} ← x^{k+1} + J
10 C' ← C' ∪ C

   /* Stage 2: Resolve collisions */
11 while x^{k+1} has collisions do
12 |  Detect colliding features using continuous detection between x^k and x^{k+1} (3.9.2.1)
13 |  C ← Constraints of colliding feature pairs (3.9.2.2)
14 |  foreach Impact zone Z of C do
15 |  |  J ← Impulse velocities obtained from LCP solver and Z (3.9.2.3)
16 |  |  x^{k+1} ← x^{k+1} + J
17 |  end
18 |  if 1st while loop iteration then
19 |  |  C' ← C' ∪ C
20 |  end
21 end

22 Constrain the next velocity solve (line 2) using C' (3.9.3)
```

3.9.1.1 Detecting Nearby Features

The detection of nearby features is performed on the last known penetration-free state of the mesh (i.e $\mathbf{x}^k$). The detection is divided into two phases. The first phase, referred as *broad-phase detection*, searches for pairs of features that are potentially nearby using axis-aligned bounding-boxes (AABB). These AABBs individually enclose each feature with the smallest possible rectangle prism that aligns with the coordinate axes of the world-space or reference-space. The AABBs are then used to assemble three bounding volume hierarchy (BVH) trees: one for the vertices, one for the edges, and one for the triangles. Overlapping AABB pairs can then be queried by recursively traversing two given BVH trees. It is sufficient to only search for vertex-triangle (VT) and edge-edge (EE) pairs whose AABB pairs overlap.

When generating the AABBs, the margins of the rectangles are adjusted to correspond to the minimum distance m that the features should be kept apart. Particularly, the width, length, and height of each AABB should be increased by $m/2$. The minimum distance itself is left as a user-defined parameter, typically corresponding to a multiple of the solid-shell thickness.

Once the potentially nearby feature pairs (i.e. whose AABBs overlap) have been found, they are passed to the *narrow-phase detection* phase to confirm the feature pairs that are actually within a distance of m apart. Given a VT pair, their distance apart is measured from the vertex to the closest point of the triangle. For an EE pair, their distance apart is measured between the two closest points of the two edges. An efficient implementation for calculating the two closest edge points can be found in [34, Appendix B.3] which does not require an iterative search.

3.9.1.2 Generating Constraints From Nearby Features

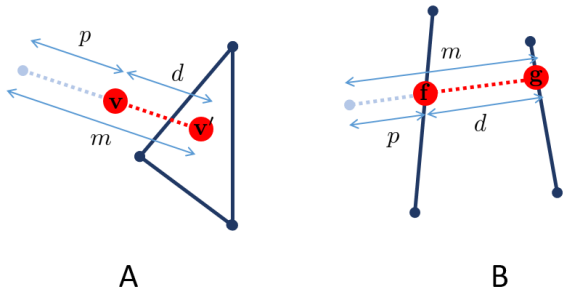


Figure 3.7: Penetration Distance. (A) shows a nearby vertex-triangle pair scenario. $\mathbf{v}$ is the vertex position and $\mathbf{v}'$ is the vertex projected onto the triangle plane that is closest to $\mathbf{v}$. d is the distance between $\mathbf{v}$ and $\mathbf{v}'$. m is the minimum distance that vertex $\mathbf{v}$ should be kept away from $\mathbf{v}'$. p is the distance that $\mathbf{v}$ is currently penetrating into the minimum distance m. (B) shows a nearby edge-edge pair scenario. $\mathbf{f}$ and $\mathbf{g}$ are the two closest points between the edges where $\mathbf{f}$ belongs to the first edge of the edge pair. d is the distance between $\mathbf{f}$ and $\mathbf{g}$. m is the minimum distance that $\mathbf{f}$ should be kept apart from $\mathbf{g}$. p is the distance that $\mathbf{f}$ is currently penetrating into the minimum distance m.

Each nearby feature has a corresponding unilateral constraint that is created. A unilateral constraint consists of a penetration distance p, normal vector $\mathbf{N}$, and weights of the nodes involved:

$$w_n \mathbf{N} \geq p.$$

The penetration distance p is the depth that the feature pairs are penetrating into the minimum distance m threshold. Figure 3.7 provides a visual and explanation for calculating p for a given nearby VT or EE pair. The minimum distance m itself is left as a heuristic parameter [3].

The normal vector $\mathbf{N}$ of a constraint indicates the direction/plane that the first feature of the nearby feature pair should be pushed towards to maintain their minimum distance apart. For a VT pair, the normal vector is set to the normal of the triangle and negated if necessary such that the normal vector is facing the same plane that the vertex should be pushed towards. Determining whether or not to negate the normal vector can be done using a dot product test:

$$\mathbf{N} \cdot (\mathbf{v} - \mathbf{v}') < 0$$

where $\mathbf{v}$ is the vertex position and $\mathbf{v}'$ is the vertex projected onto the triangle plane that is closest to $\mathbf{v}$. In the EE pair case, the normalized cross product of two edge vectors is used as the constraint's normal vector, which is similarly negated if necessary such that the normal vector faces the same plane that the first edge of the edge pair should be pushed towards. This normal vector should be negated if

$$\mathbf{N} \cdot (\mathbf{f} - \mathbf{g}) < 0$$

where $\mathbf{f}$ and $\mathbf{g}$ are the closest points of the first and second edge respectively.

There is a boundary case for the normal vector of an EE pair constraint that requires attention. If $\mathbf{f}$ or $\mathbf{g}$ resides on the very end of its edge, then the normal vector of the EE pair constraint is computed as the normalized $\mathbf{f} - \mathbf{g}$ vector without any needed negation. This is required for cases where the cross product approach would actually prevent a nearby edge from passing by a boundary edge.

The weight of a vertex indicates the fraction of the constraint's impulse that is to be applied to the vertex. For a VT pair, the vertex $\mathbf{v}$ is assigned a weight of $w_0 = 1.0$. The weights of the 3 triangle vertices (w_1, w_2, w_3) are equal to the negated barycentric coordinates of $\mathbf{v}'$ where $\mathbf{v}'$ is $\mathbf{v}$ projected onto the triangle plane and closest to $\mathbf{v}$. The reasoning for using negative weights for the triangle vertices is to scale the triangle vertex impulses such that they push in the opposite direction of the impulse that is applied to vertex $\mathbf{v}$. For EE pairs, the vertex weights are based on how close the vertices are to the two closest edge points:

$$w_0 = \frac{\text{dist}(\mathbf{P}_0, \mathbf{f})}{\text{dist}(\mathbf{P}_0, \mathbf{P}_1)} \qquad w_1 = 1 - w_0$$

$$w_2 = -\frac{\text{dist}(\mathbf{P}_2, \mathbf{g})}{\text{dist}(\mathbf{P}_2, \mathbf{P}_3)} \qquad w_3 = -1 - w_2.$$

$\mathbf{P}_n$ and w_n refer to the position and assigned weight of vertex n. n ranges from 0 to 3 where $n = 0$ and $n = 1$ refer to the vertices of the first edge of the EE pair while $n = 2$ and $n = 3$ refer to the vertices of the other edge. $\mathbf{f}$ and $\mathbf{g}$ refer to the two points of the two edges that are a minimum distance apart. The operator $\text{dist}(\cdot)$ simply denotes the distance between two given points. Similar to the VT constraint, the weights of the opposite edge, namely w_2 and w_3, end up becoming negative in order to repel the second edge in the opposite direction of the first edge's impulse.

Once the constraints have been created, they are represented on a node basis. Each front node that correspond to a vertex of a given nearby feature pair has a copy of the penetration magnitude, normal vector, and vertex's assigned

weight. These per-node constraints are then duplicated for the back nodes. This allows the front and back nodes to be subjected to the same impulse. If the front nodes were only affected by the impulses, plastic embedding would cause impulse-induced displacements of the front nodes to accumulate in the reference-space, causing the director vectors in the reference-space to unnaturally elongate from the back nodes.

3.9.1.3 Creating Impulses using Constraints

The task of obtaining the impulse velocities, subjected to unilateral constraints, can be seen as a mixed LCP [8, 12]. The mixed LCP is essentially a linear problem (e.g. solve for $\mathbf{x}$ in $\mathbf{Ax} = \mathbf{b}$) that accounts for bilateral and unilateral constraints. For determining the impulses, the LCP is presented as

$$\begin{bmatrix} \mathbf{M} & -\mathbf{B}^\intercal & -\mathbf{U}^\intercal \\ \mathbf{B} & \mathbf{0} & \mathbf{0} \\ \mathbf{U} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{v} \\ \lambda \\ \mathbf{z} \end{bmatrix} + \begin{bmatrix} -\mathbf{f} \\ -\mathbf{b} \\ -\mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{s} \end{bmatrix} \tag{3.17}$$
$$0 \leq \mathbf{z} \perp \mathbf{s} \geq 0.$$

Laying out the notations first, $\mathbf{M}$ is the mass block matrix. $\mathbf{B}$ and $\mathbf{U}$ are the bilateral and unilateral constraint matrices respectively, capturing the left-hand side (LHS) portion of each constraint equation. $\mathbf{v}$ is the nodal impulse velocities to solve for, while λ and $\mathbf{z}$ are the constraint impulses that are solved along the way. $\mathbf{f}$ is the vector of nodal forces. $\mathbf{b}$ and $\mathbf{p}$ are the bilateral and unilateral constraint vectors respectively, capturing the right-hand side (RHS) portion of each constraint equation. $\mathbf{s}$ is a slack variable to transform the inequality equations into equality equations. The expression involving the $\perp$ symbol indicates the complementarity constraint that must be met, and is shorthand for

$$\mathbf{z} \cdot \mathbf{s} = 0,\ \mathbf{z} \geq 0,\ \mathbf{s} \geq 0.$$

The contents of the mass and constraint variables are structured on a nodal basis. Assuming a system with n nodes, $\mathbf{M}$ is structured as a $3n \times 3n$ diagonal mass matrix. The unilateral constraint matrix $\mathbf{U}$ is sized $c \times 3n$ where c is the number unilateral constraints. $\mathbf{U}$ captures the weights w and normals N of the LHS portion of each unilateral constraint. The unknown impulse velocities to be solved are delegated to the $3n$-sized vector $\mathbf{v}$. For the RHS of the unilateral constraints, the unilateral constraint vector $\mathbf{p}$ of size c holds the penetration magnitudes. Because no bilateral constraints are involved, its constraint matrix $\mathbf{B}$, constraint vector $\mathbf{b}$, and placeholder vector λ can all be assumed to be zero-sized. In addition, the force vector $\mathbf{f}$ is zero-sized.

Solving for $\mathbf{v}$ is handled by ArtiSynth, which provides a mixed LCP solver. Details on the mixed LCP solver are provided in ArtiSynth's documentation [26]. When the impulse velocities $\mathbf{v}$ are solved, they are added to the current node positions $\mathbf{x}^{k+1}$, essentially repelling the nearby features apart.

3.9.2 Continuous Collision Handling

When detecting and repelling nearby features, we assume that the impulses, which are calculated based on the mesh at time step k, would always correct the mesh at time step $k+1$ into a penetration-free state. This assumption is only

reasonable when the mesh geometry does not significantly differ between time step k and $k+1$. However, in cases where the growth model is highly strained, the mesh geometry between time step k and $k+1$ can greatly differ. For these harder cases, continuous collision detection (CCD) is deployed where the mesh geometry at both time steps are accounted in order to formulate constraints that adequately undo the penetrations.

3.9.2.1 Detecting Collided Features

The CCD implementation follows closely to the works of Owens et al. [34], whom used AABBs and a root-finding method to respectively search and confirm both EE and VT collisions that occurred between $\mathbf{x}^k$ and $\mathbf{x}^{k+1}$. Parallel to detecting nearby features, the CCD implementation is divided into two phases: *broad-phase detection* and *narrow-phase detection*.

In broad-phase detection, potential collisions are searched in a similar matter to how potential nearby features were searched. However, a key difference is that each AABB now encloses the space-time path that a feature traversed between time step k and $k+1$, assuming that vertex displacement is linear across each time step. For CCD, the margins of the AABBs do not have to be extended except by a small error tolerance.

In narrow-phase detection, the potential VT and EE collisions are examined closely to confirm whether or not they have actually collided. Particularly, for each potential collision, the two features are tested for coplanarity and intersection. If both coplanarity and intersection hold to be true at a given time between time step k and $k+1$, then the two features would have collided along their space-time paths. For a given feature pair, finding the time(s) that they become coplanar requires solving for the roots of a cubic polynomial equation. Afterwards, the feature pair can be checked for intersection at their times of coplanarity in constant time. Refer to Section 4.2.2 in Owens et al. [34] for details regarding implementing the coplanarity and intersections tests.

3.9.2.2 Generating Constraints from Detected Collisions

The constraints of the detected collisions can be generated using the same process that was done for nearby features but with a few minor differences. The primary difference is that the impulses specified from the constraints are calculated to undo the penetration of collided features instead of enforcing the minimum distance m between feature pairs.

Generating a constraint for a given VT collision is straight forward. The penetration magnitude p is formulated as the distance from the vertex to the point of the triangle plane that is closest to the vertex. Both vertex and triangle are assumed to be at time $k+1$. The constraint's normal vector $\mathbf{N}$ is set to the normal of the triangle at time $k+1$. For CCD, determining whenever or not the normal vector has to be negated can be done using a double dot product test. Specifically, the normal vector should be negated if

$$\mathbf{N} \cdot (\mathbf{v}^{k+1} - \mathbf{f}^{k+1}) > \mathbf{N} \cdot (\mathbf{v}^k - \mathbf{f}^k)$$

where $\mathbf{N}$ is the normal vector being tested, $\mathbf{v}^k$ is the vertex position at time k, and $\mathbf{f}^k$ is the point on the triangle plane at time k that is closest to $\mathbf{v}^k$. Lastly, the 4 vertex weights of the VT collision constraint are calculated in the same way as in the nearby case except that the barycentric coordinates, before their negation, is calculated from the collision point.

Along with the VT constraints, a constraint is generated for each EE collision. For a given EE constraint, its penetration magnitude p is the distance between the closest points of the two edges at time $k+1$. When calculating the normal vector, the same procedure that was used for the nearby EE case can be done with the two edges at the exact time of their collision. Unlike the CCD VT case, the CCD EE case does not require a double dot product test; the same dot product test from the nearby EE pair case be used. Lastly, when assigning weights to the 4 vertices of the EE constraint, the collision point of the two edges is used in place of both btf and $\mathbf{g}$ for calculating the vertex weights.

3.9.2.3 Creating Impulses from Detected Collisions

The constraints are binned by *impact zone* [15]. An impact zone is a set of constraints whose vertices overlap. For example, a VT constraint and EE constraint would be binned into the same impact zone if one of the vertices in the VT pair can also be found in the EE pair. If a constraint does not share a common vertex with any constraint, it is placed in its own impact zone.

The impact zones are resolved one at a time using the same LCP solver. Particularly, for a given impact zone, its constraints are inputted into the LCP solver to solve for the appropriate impulses that satisfy the constraints. The steps in solving for the impulses velocities using constraints and the LCP solver is the same as in the nearby case. The solved impulse velocities are then added to the node positions of the impact zone to eliminate its penetrations.

After each impact zone has been dealt with, new collisions can potentially occur. Therefore, the entire procedure of detecting collisions, generating constraints, and resolving impact zones is repeated until all collisions have been eliminated.

3.9.3 Velocity Correction

Once the mesh becomes free of penetrations, the impact velocities of resolved nearby and collision pairs need to be updated. In contrast, the impulses earlier only corrected the positions of the nodes; the nodal velocities have yet to be updated. Velocity correction is done by constraining the subsequent FEM system solve with the constraints of the nearby feature pairs and constraints of the primary (i.e. non-secondary) collision pairs. The constrained FEM system essentially becomes another LCP that is to be solved:

$$\begin{bmatrix} \hat{\mathbf{M}} & -\mathbf{B}^{k\mathsf{T}} & -\mathbf{U}^{k\mathsf{T}} \\ \mathbf{B^k} & \mathbf{0} & \mathbf{0} \\ \mathbf{U^k} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{v}^{k+1} \\ \lambda \\ \mathbf{z} \end{bmatrix} + \begin{bmatrix} -\mathbf{M}\mathbf{u}^k - h\hat{\mathbf{f}}^k \\ -\mathbf{b}^k \\ -\mathbf{p}^k \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{s} \end{bmatrix} \tag{3.18}$$
$$0 \leq \mathbf{z} \perp \mathbf{s} \geq 0.$$

This equation is essentially the regular FEM system (equation (3.10)) combined with the mixed LCP equation (equation (3.17)), forming the constrained FEM system. The terms $\hat{\mathbf{M}}$ and $\hat{\mathbf{f}}^k$ encapsulate the existing terms from the original FEM system and are expanded as

$$\hat{\mathbf{M}} = \mathbf{M} - h\frac{\partial \mathbf{f}^k}{\partial \mathbf{v}} - h^2\frac{\partial \mathbf{f}^k}{\partial \mathbf{x}} \qquad\qquad \hat{\mathbf{f}} = \mathbf{f}^k - h\frac{\partial \mathbf{f}^k}{\partial \mathbf{v}}\mathbf{v}^k.$$

The constraint terms from the mixed LCP equation, namely $\mathbf{B}$, $\mathbf{U}$, $\mathbf{b}$, and $\mathbf{p}$, are appended with a k superscript to indicate that the terms refer to the constraints that were computed immediately prior (i.e. from time step k).

The constrained FEM system is solved in the same matter as solving the mixed LCP equation, yielding the updated nodal velocities $\mathbf{v}^{k+1}$ that respect the specified constraints. Consequently, unless the nodal strain forces $\mathbf{f}^k$ are overwhelmingly opposing, the directions of the impact velocities of the nearby and collision pairs become reflected.

Although not used in our growth simulations, friction is supported by the box friction model [26]. The friction model is applied as a post-hoc correction to $\mathbf{v}^{k+1}$ using a simplified version of the constrained FEM system. In the system, $\hat{\mathbf{M}}$ is replaced with $\mathbf{M}$, and includes extra constraints that are tangential to the directions of the contact points.

3.9.4 Accounting for Remeshing

Between the end of position correction and start of velocity correction, the growth procedure is carried out, including remeshing. The constraints that are to be passed to the constrained FEM system (equation (3.17)) need to be interpolated whenever the mesh topology changes. Because the constraints are represented on a nodal basis, interpolation is straight forward: new nodes are given an even average of its parent nodes' constraints, and the constraints of deleted nodes are discarded.

3.9.5 Culling Redundant Nearby and Collided Features

The computation time of the mixed LCP solver can be improved by reducing the number of constraints in the system. One way to reduce the number of constraints is to cull out redundant nearby or collided feature pairs that were detected. In the CCD case, we use the following rules:

- Given a detected VT pair, it can be ignored if any of the connected edges of the vertex is part of a detected EE pair that has an earlier collision time.
- Given a detected EE pair, it can be ignored if any of its vertices is involved in a detected VT pair that has an earlier collision time.

In the nearby case, the same rules can be applied by substituting the collision time with the penetration distance.

CHAPTER 4

RESULTS AND DISCUSSION

In this Chapter, we showcase a number of simulated growth experiments that investigates the capabilities of the new solid-shell growth approach. The first set of experiments intends to verify the surface buckling capabilities of the framework by growing basic shapes. The second set of experiments showcases the growth of delicate wrinkles, large ripple cascades, macroscopic deformations, and large bending deformations. Furthermore, both adaptive remeshing and plastic embedding undergo ablation experiments to contrast between their inclusion and absence in order to observe their benefits. We also demonstrate the simulation of residual stress that accompany growth. Lastly, two models are grown to showcase large plastic deformations involving collisions.

4.1 Basic Shapes

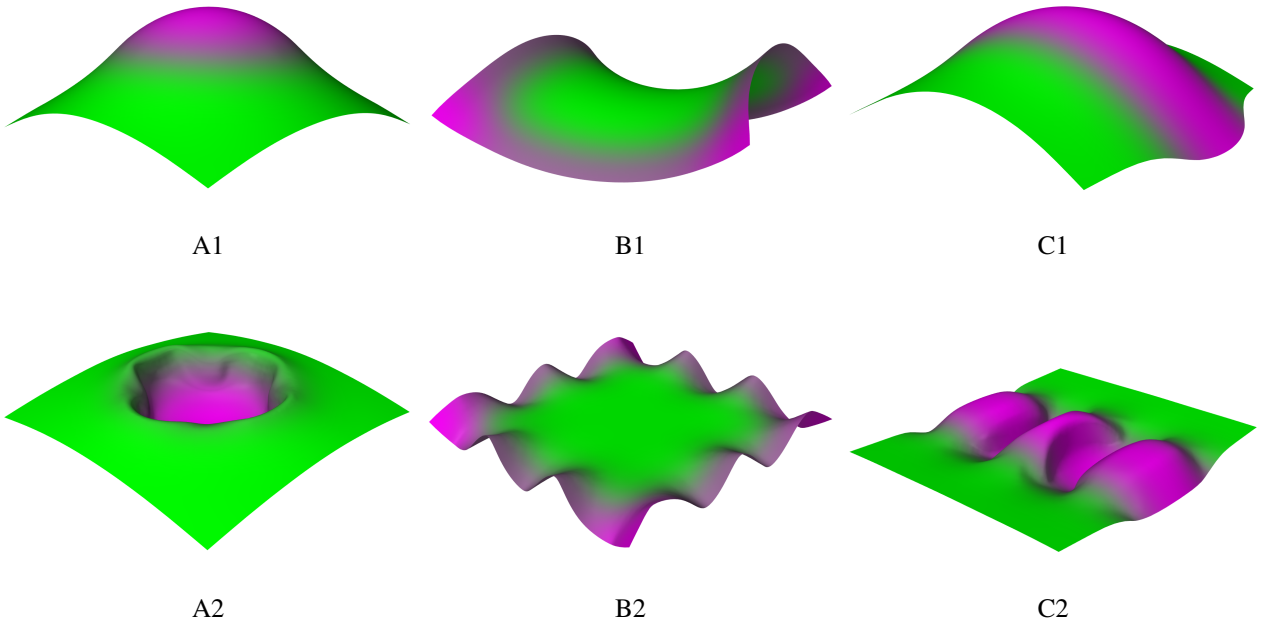


Figure 4.1: Basic Grown Shapes. The bulge (A1), saddle (B1), and arc ridge (C1) are produced by respectively placing morphogen (denoted in purple) in the center, boundary, and middle strip of the sheet. If the bending stiffness of the sheet is lowered, the steep crater (A2), ripples (B2), and grooves (C2) are respectively produced instead.

Before growing elaborate models, we simulate the growth of a set of basic shapes to verify that the buckling patterns are consistent with previously published simulation results. These basic shapes are shown in Figure 4.1. The

variety in shapes are attributed by the different combinations of bending stiffness and morphogen placement used.

The shapes in the top row of Figure 4.1 were grown using a relatively higher bending stiffness material than those in the bottom row. Material with high bending stiffness has the tendency of producing shapes and patterns with lower curvature. A prime example is the bulge shape (A1), which can be produced by applying morphogen in the center of a sheet with such material. The bulge shape produced is comparable with existing experiments [29, Fig. 7.3] [22, Fig. 3]. When morphogen is applied on the sheet perimeter, a comparable saddle (B1) is produced [29, Fig. 6.3a]. The arc ridge (C1), although an uncommon shape to compare, can be seen as a variation of the bulge experiment (A1) where the bulge is instead elongated along the strip of morphogen, forming a ridge.

When the bending stiffness is lowered, the shapes and patterns exhibit higher curvature. In other words, the energy associated with bending becomes lower, permitting bends to occur more freely in the minimal energy shape. This can be observed in the ripples (B2) and grooves (C2) shapes, which are consistent with previous experiments [29, Fig. 6.3b and 6.6]. The crater shape (A2), although another uncommon shape to compare, has subtleties that are consistent with the effects of lower bending energy: the bends along its ridge are steeper and small faint buckled protrusions appear along its sides.

4.2 Ripple Cascade Demo

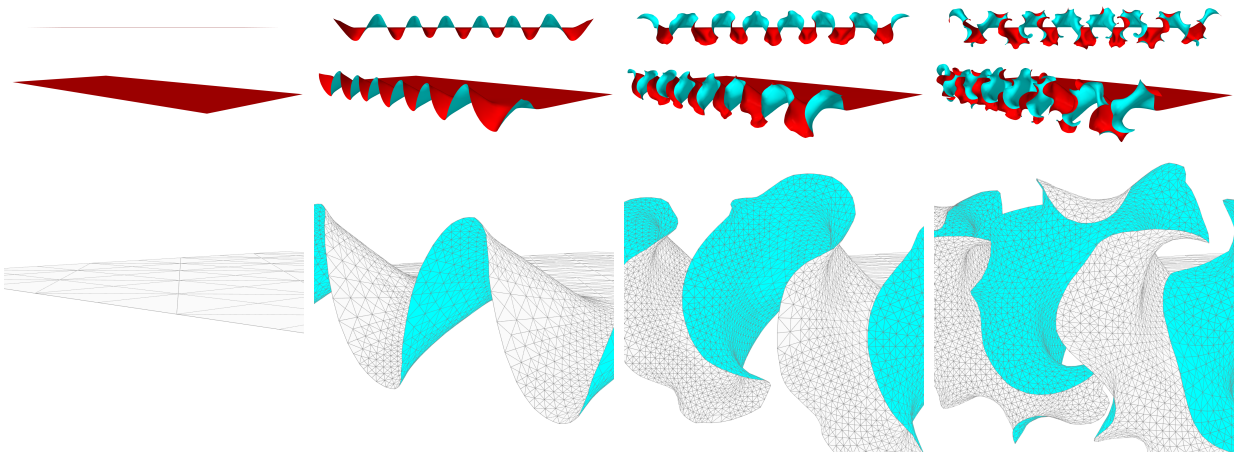


Figure 4.2: Time-Lapse Growth of a Ripple Cascade. The front view (top row), oblique view (middle row), and a close-up view of the shape's mesh topology (bottom row) are shown. The ripple cascade is generated by applying diffusible morphogen along the front side of the sheet.

Growing tissue can develop fractal patterns along their edges. The fractal patterns that appear in the kale leaf (*Brassica oleracea*) are a notable example and have been modelled using rule-based growth approaches, namely L-Systems [36]. Fractal patterns can also be reproduced using physics-based approaches. When a flat sheet composed of thin-shells is subjected to anisotropic plastic strain along one of its edge, the edge can buckle into a cascade of ripples [47, Fig. 11].

We can perform the same ripple cascade experiment using our augmented solid-shells. The augmented solid-shells particularly builds upon the original experiment by introducing plastic embedding and adaptive remeshing, whereas the original experiment relied upon the multiplicative decomposition approach and use of pre-refined meshes. Figure 4.2 provides a time-lapse of the ripple cascade experiment using augmented solid-shells. In this experiment, anisotropic morphogen is applied along one side of the solid-shell sheet. The sheet has a length-thickness ratio of 2000:1, Young's Modulus of 10^6, and started with 256 elements. When enough strain accumulates, buckling occurs to form the initial set of ripples (B). As morphogen continues to be supplied, (C) highlights the early formation of secondary ripples that occur on the initial set of ripples, which later become more prominent in (D).

One of the significant features of our ripple cascade experiment compared to the original experiment is the formation of larger ripples. The larger ripples are attributed to plastic embedding and the adaptive remesher. Plastic embedding actively transfers the plastic strain into the rest configuration, essentially minimizing the plastic strain in order to prevent large and unstable plastic forces. Alongside, the adaptive remesher helps ensure that the aspect ratio of each element remains in balance during growth, reducing the risk of ill-conditioned elements. The balanced elements are particularly noticeable in the bottom row of Figure 4.2.D despite prolonged growth.

The inclusion of adaptive remeshing has an additional advantage of allowing the user to perform growth experiments without needing to preemptively predict the location and magnitude of mesh refinement needed. The original experiment required the use of *a posteriori* refined meshes [47, Fig.10], which can be time-consuming to construct. With adaptive remeshing, highly-detailed patterns can still emerge from a coarse mesh that automatically refines itself where needed.

4.3 Delicate Wrinkle Pattern Demo

There is recent research on simulating the formation of delicate wrinkle patterns on thin tissue. Simulating these patterns requires elements that can handle steep bending deformations and prolonged growth. Consequently, thin-shells with deformable rest configurations have become a common choice. Particularly, hinge-based thin-shells coupled with plastic embedding can be used to experimentally generate such patterns [13]. Another work used non-Euclidean plates to generate delicate wrinkle patterns by gradually scaling the reference metric tensor of each triangle [28].

The formation of delicate wrinkles can also be achieved using solid-shells with plastic embedding. In contrast to existing approaches for creating wrinkles, we include a remeshing component in our augmented solid-shells. The remesher can adaptively refine the mesh where wrinkling is anticipated. The experiment is shown in Figure 4.3, which displays the time-lapse growth of a pattern that resembles a reaction-diffusion pattern. The pattern was generated by applying diffusible morphogen (from $t = 0$ to $t = 2$) in the center of the sheet. The bending stiffness of the sheet was set to a lower extreme (length-thickness ratio of 10000:1 and Young's Modulus of 100) to accommodate the formation of delicate wrinkles. In Figure 4.3, the sheet (A) is initially flat with a uniform mesh resolution of only 200 elements. After 1 second (B), the sheet remains flat but the remesher anticipates buckling by preemptively increasing the mesh resolution in the center. After 2 seconds (C), buckling occurs, forming the initial wrinkles. The mesh resolution near

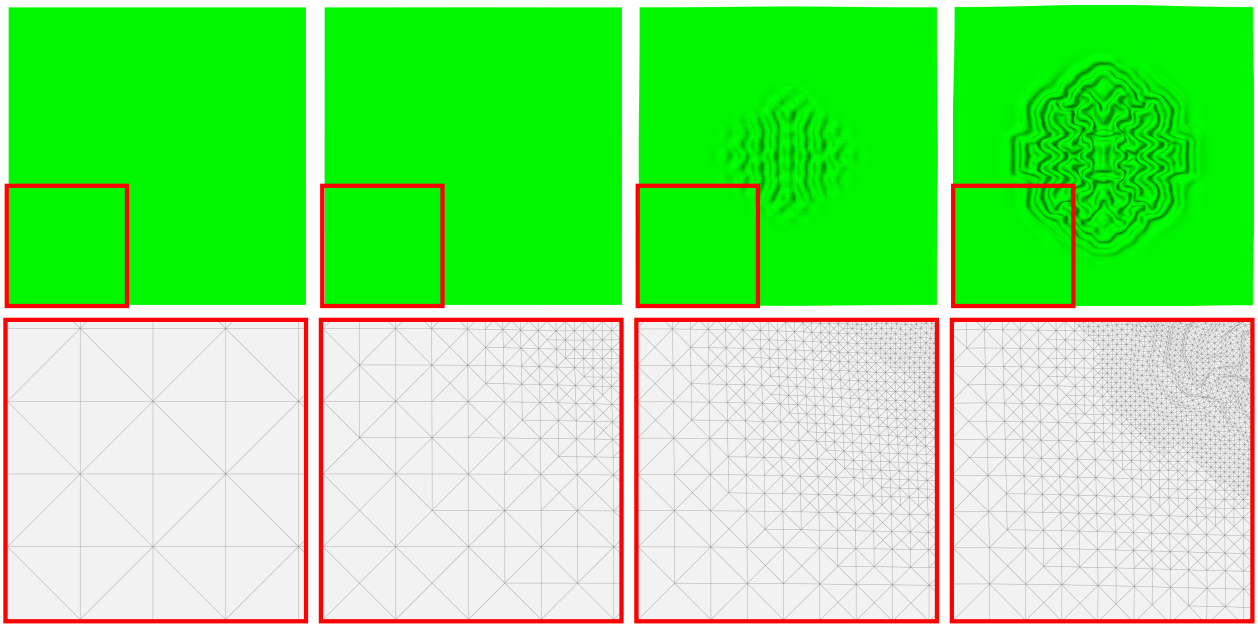


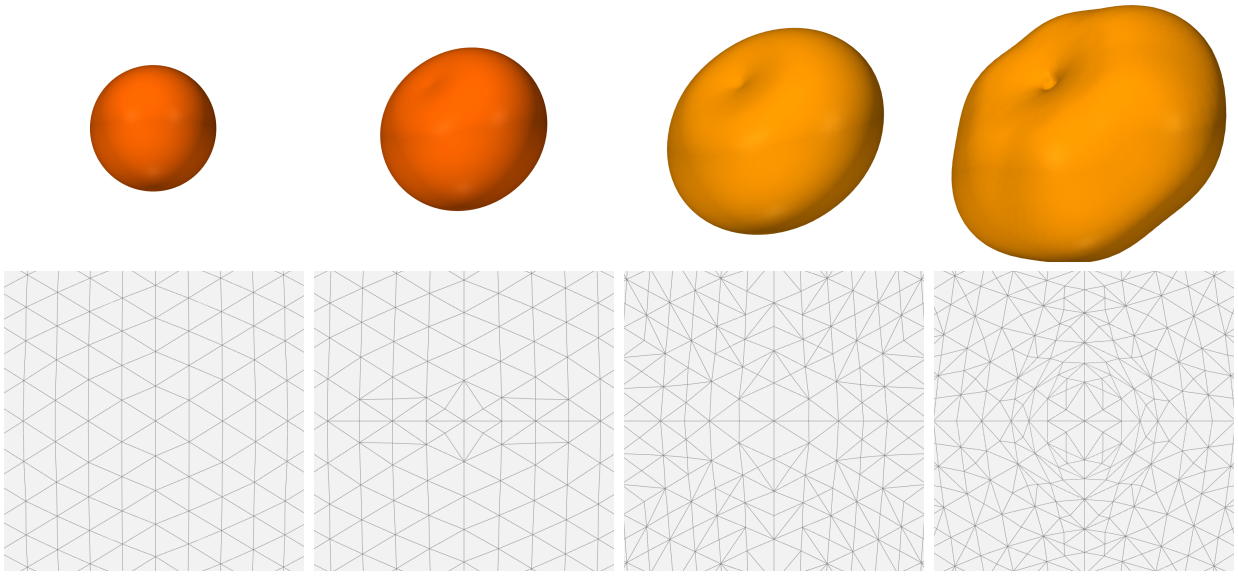
Figure 4.3: Time-Lapse Growth of a Delicate Wrinkle Pattern. Each rendered image (top row) is accompanied with a close-up view of its mesh topology (bottom row). The pattern is generated by supplying diffusible morphogen to the center of a very thin and highly compliant sheet.

the center further increases in resolution to respect the curvature metric and increasing strain. At 3 seconds (D), the mesh resolution adapts again to accommodate the more propagated and detailed wrinkle pattern. At the same time, the boundary of the mesh is hardly refined, as shown in the bottom-left of (D). Overall, this experiment showcases the ability of the augmented solid-shells in handling a growth scenario that would normally be simulated using thin-shells with deformable rest configurations.

4.4 Fruit-like Shape Demo

Beyond delicate wrinkles and ripple cascades, mechanical stresses can contribute to the formation of features at macroscopic scale. In particular, the overall shapes found in many fruits and vegetables have been suggested to be the result of stress-induced buckling [50]. This suggestion can be supported using simulation, as shown in Figure 4.4 where a fruit-like model is grown by applying morphogen uniformly across a starting regular icosahedron sphere composed of solid-shells. The Young's Modulus is set to 10^6, and unlike the previous demos, the solid-shells are set to be very thick (thickness to sphere radius ratio of 1:5) to model the fleshy interior that resides behind a fruit's surface.

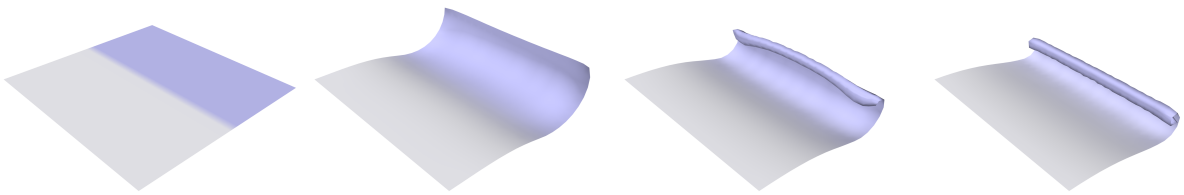
Across the growth time-lapse in Figure 4.4, the starting sphere develops two polar navels, forming a biconcave ellipsoid. This resulting morphology is consistent with the fact that biconcave ellipsoids have a lower energy shape than the sphere [4]. The sphere shape would be retained if only the stretching energy is accounted [7]. When grown further, bulges form to complement the biconcave ellipsoid. Overall, the experiment showcases the augmented solid-shell's ability in modelling large plastic deformations for tissue with considerable thickness.



A (t=0) A (t=13) A (t=15) B (t=20)

Figure 4.4: Time-Lapse growth of a Fruit-like Shape. Each rendered image (top row) is accompanied with a close-up view of the fruit's top navel (bottom row), highlighting the symmetry of the mesh topology even after growth and remeshing. The fruit-like shape is grown by applying a constant morphogen concentration uniformly across the surface of a regular icosahedron.

4.5 Curling Sheet Demo



A (t=0) B (t=5) C (t=10) D (t=15)

Figure 4.5: Time-Lapse Simulation of Plastic Bending. Morphogen is placed on one-half of the sheet to anisotropically shrink and expand the front and virtual back surface respectively of the solid-shells, producing an exaggerated curling effect.

There are existing approaches on simulating plastic bending. A recent approach is to use an extension of non-Euclidean plates that supports adjustable rest curvatures. These extended non-Euclidean plates are capable of simulating the curling of paper [7]. For solid-shells, a bilayer of solid-shells can be used where the bottom layer is only subjected to stretching growth [51]. By growing only the bottom layer, authors were able to make a flat star-like structure curl upwards into a sphere-like structure.

In this experiment, we demonstrate large plastic bending using a single layer of solid-shells by leveraging on our bending morphogen. Figure 4.5 provides a time-lapse of the experiment. The starting square sheet (A) has a length-thickness ratio of 100:1 and a Young's modulus of 10^5. On one-half of the sheet, anisotropic (type PER) bending

morphogen is placed. To produce a more exaggerated curling effect, the same half of the sheet has morphogen that induces ansisotropic (type PER) shrinking. The initial effect of the morphogen is shown in (B) where one side of the sheet begins curling. As morphogen continues to be absorbed, (C) and (D) shows the curling curvature becoming more pronounced, causing the sheet to roll up into itself. To prevent the curling sheet from intersecting itself, collision handling is enabled in the simulation.

While thin-shells are known for handling plastic bending, this experiments reveals that solid-shells can also be used. In comparison to the curling paper experiments done using non-Euclidean plates [7], we demonstrate curling curvature of similar magnitude with the addition of adaptive remeshing and collision handling working alongside. At the same time, the experiment demonstrates that it is possible to simulate plastic bending using a single layer of solid-shells.

4.6 Remesher and Plastic Embedding Ablation

Ablation tests are performed to observe the effects of plastic embedding and adaptive remeshing on the formation of detailed shapes. In each ablation test, morphogen is applied to one side of the sheet, identical to how the ripple cascade experiment was setup. However, the ablation tests attempt to simulate growth with plastic embedding and remeshing both disabled, or only one of the two enabled, or both enabled. Tests without plastic embedding fallback to the multiplicative decomposition approach to model plasticity. The four ablation tests are shown in Figure 4.6, which showcases the resulting world-space shape (red), reference-space shape (cyan), and reference-space topology (grey) for each test.

In the first ablation test (A), both plastic embedding and remeshing are disabled. The sheet in world-space buckles into a single arc while its reference-space shape and reference-space topology remains flat and unchanged. When plastic embedding is enabled (B), multiple arcs are formed instead in both the world-space and reference-space shapes. If remeshing is enabled instead of plastic embedding (C), the bends occur at an even greater frequency. Both (B) and (C) reflect the importance of evolving the reference-space shape and maintaining a high resolution mesh, which can otherwise prevent the formation of smaller and more detailed shapes. When both plastic embedding and remeshing are enabled (D), the advantages of both are combined, allowing bends to occur within bends, forming the ripple cascade.

4.7 Tissue Incision and Residual Stress Demo

Large growth deformations often have a side effect of producing residual stress. This accompanying stress is due to the fact that the grown rest configuration cannot be represented in Euclidean space. The existence of residual stresses has been speculated to be relevant in biological processes. Particularly for artery tissue, residual stress is a suggested mechanism that optimizes the tissue's load-bearing shape [41].

In this experiment, we demonstrate residual stress by showcasing its build-up and release, as shown in Figure 4.7. The cylinder model (A) is first grown in order to accumulate residual stress (B). The grown model is then vertically incised in half (C). By incising the model, the area along the incision becomes less physically constrained and therefore

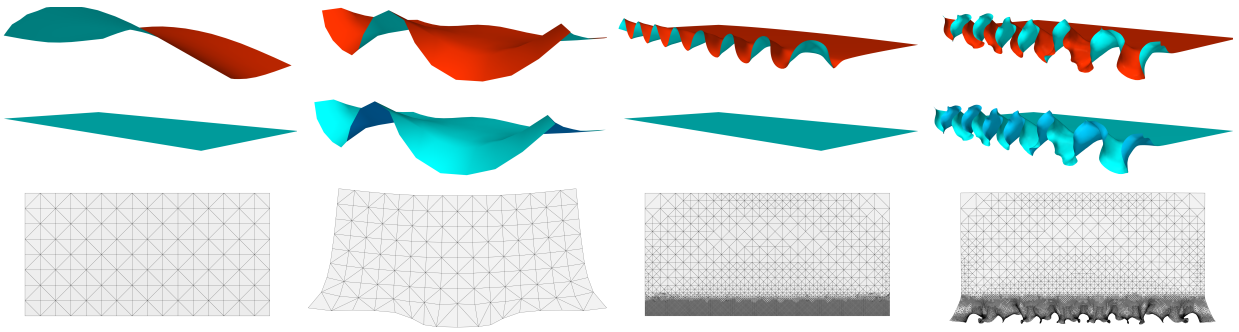


Figure 4.6: Ablation of Plastic Embedding and Remeshing. Experiments include: (A) neither plastic embedding nor remeshing, (B) plastic embedding alone, (C) remeshing alone, and (D) both plastic embedding and remeshing present. For each experiment, the world-space shape (red), reference-space shape (cyan), and reference-space topology (grey) of the sheet is shown. Growth is induced by applying morphogen along one side of the sheet.

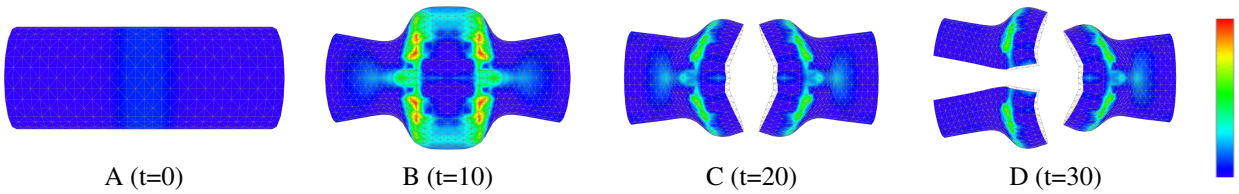


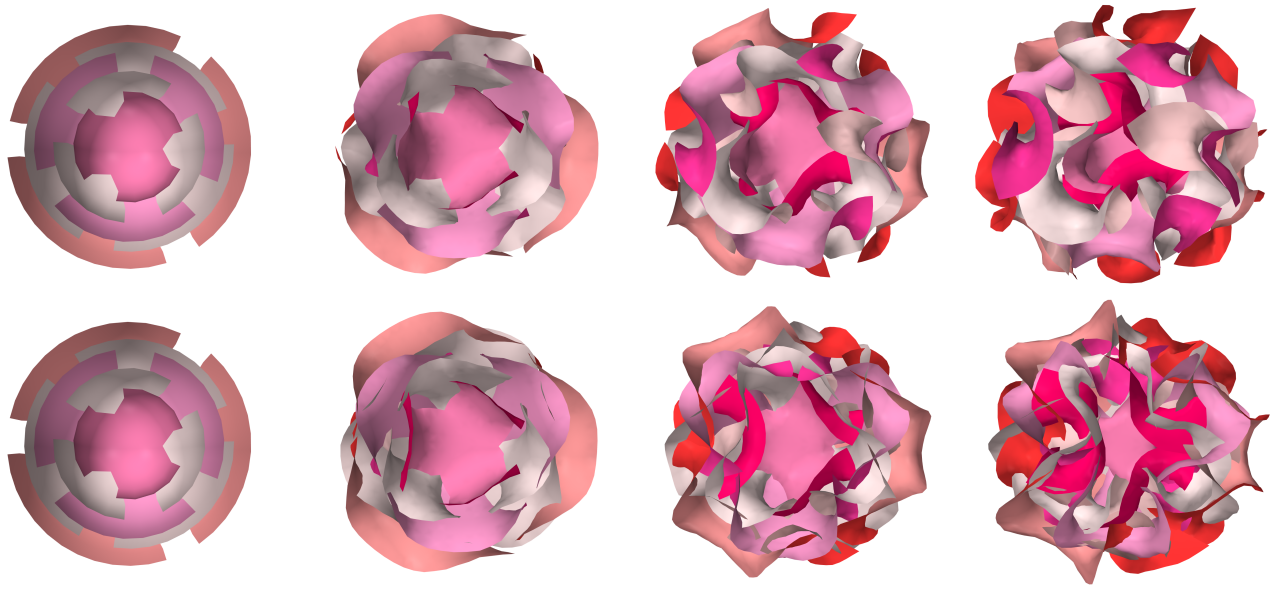
Figure 4.7: Build-Up and Release of Residual Stress. (A) depicts the starting cylinder model that is subjected to growth for 10 seconds (B). (C) is the model after being vertically cut in half and allowed to relax for 10 seconds. (D) shows the left half of the model after being horizontally cut in half and relaxed for 10 seconds. The colormap denotes the magnitude of residual stress from low (blue) to high (red).

permitted to expand into a lower energy shape. The release of residual stress is evident by the new shape (C) that the model takes on after incision. When the second incision is performed (D), the model again takes on a new shape. The figure also provides a color-mapping of the residual stress, which dissipates after each incision. Altogether, the experiment highlights that the growing solid-shells can capture residual stress dynamics that accompany large plastic deformations.

4.8 Ripple Bouquet: Collision Handling Demo 1

We departure from existing FEM growth scenarios in the literature to now simulate large plastic deformations with collision handling. In this experiment, a collision stress test is setup where multiple layers of ripples are grown and forced to collide with each other. The time-lapse growth of this model is shown in the top row of Figure 4.8. The model was set with a Young's modulus of 10^7, and radius to thickness ratio of 2172:1. The model was grown by applying diffusible morphogen along the boundary of each layer. After 8 seconds of growth, the starting model (top A) transforms into (top B), which has small simple ripples developed. When the secondary ripples emerge in (top C), collisions begin to occur. Eventually, the model uptakes a form where the developed ripples exists in a penetration-free state (top D).

The bottom row of Figure 4.8 showcases the same growth time-lapse but with collision handling disabled. Without



A (t=0) B (t=8) C (t=17) D (t=25)

Figure 4.8: Time-Lapse Growth of a Ripple Bouquet. The model uses five layers of thin sheets. Growth is initiated by applying diffusible morphogen along the boundary of each layer. Rippling deformations become large enough for collisions to occur. The bottom row shows the same time-lapse but with collision handling disabled.

collision handling, penetrations become severe artifacts during simulation (bottom C and D). Aside from resolving penetrations, the addition of collision handling influences the morphology that the model is permitted to take. This is most noticeable when comparing between the final penetration-free model (top D) with its final penetration-permitted form (bottom D). Overall, this experiment highlights the importance of collision handling on morphology development for large plastic deformations.

4.9 Confined Growth: Collision Handling Demo 2

When a growth model is confined to grow within a fixed volume, the model can deform into the matching shape of the fixed volume. A simulation of this scenario is depicted of Figure 4.9. The starting model is composed of 8 leaflets that are distributed circularly in two layers (top A). The bottom innermost nodes are permanently pinned in-place throughout the simulation. Growth is enabled by supplying the outermost nodes with a constant morphogen concentration that diffuses throughout. Young's modulus was set to 10^5 with a sphere radius to leaflet thickness ratio of 2000:1.

When growth begins, the leaflets expand outwards towards the wall of the sphere. Eventually, the leaflets collide against the sphere, causing growth to be deflected along the inner surface of the sphere (top B-C). With enough time, the growth model engulfs the available space and altogether becomes spherical-like itself (top D). The bottom row of Figure 4.9 shows the scenario where collision handling between leaflets is disabled. Artifacts appear due to the penetration between leaflets. Because the inner leaflets no longer press the outer leaflets against the sphere surface, the final model (bottom D) appears less uniform in curvature.

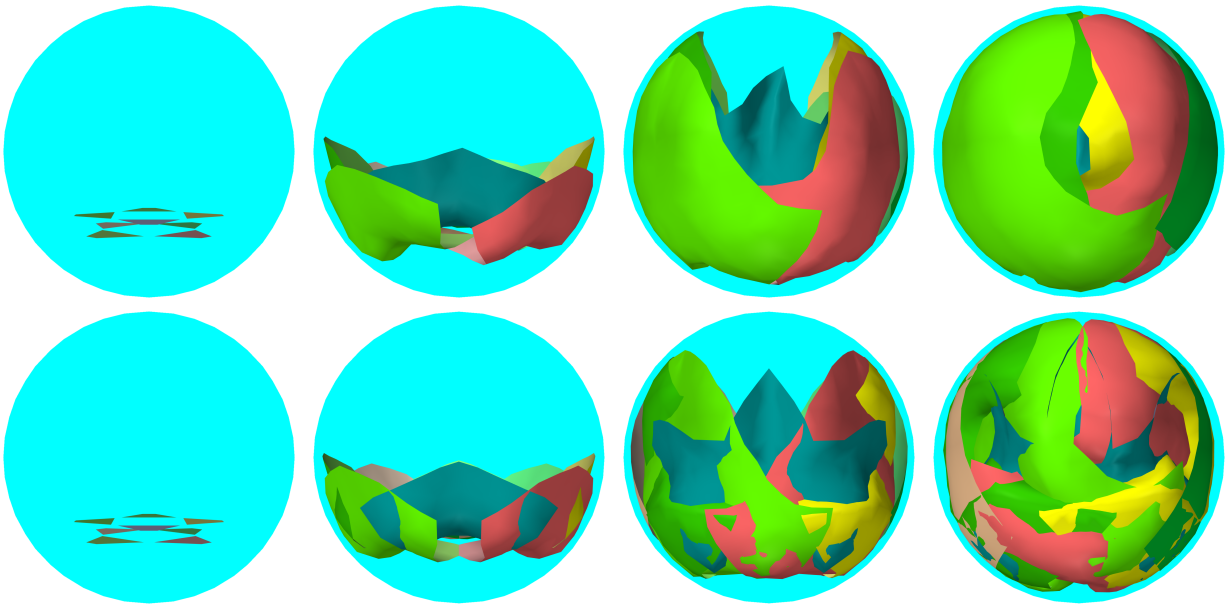


Figure 4.9: Time-Lapse Growth of Thin Leaflets within a Confining Sphere. Collision handling between individual leaflets enabled is shown (top row) and contrasted against collision handling disabled (bottom row).

Confined growth can arise during tissue development. Particularly in plants, there are studies suggesting that volume constraints can influence the possible shapes of leaves and flower petals [9, 42]. This demo aims to demonstrates the potential of using augmented solid-shells towards studying the effects of confinement on growth within a simulated environment.

4.10 Performance

The performance timings of the growth simulations are summarized in Table 4.1. Excluded from the results are the basic shape (Figure 4.1), ablation (Figure 4.6), and residual stress (Figure 4.7) demos, which did not stress the machine as much relative to the other demos. The simulations were all computed and rendered on an Intel i7-7700K and 16GB RAM machine.

The breakdown of the performance timings in the table reveals collision handling as being the most expensive component to simulate. Slowdown due to collision handling is most notable in the final performance snapshot of the Ripple Bouquet and Confined Growth experiments. Much of the computational complexity of collision handling is attributed to the contact points typically becoming persistent once they form due to plasticity. Persistent contact points cause an accumulation of unilateral constraints that the LCP solver must account for. Consequently, the collision resolution process can slow down substantially, which was also observed in another work that simulated crumbled paper using plasticity and unilateral constraints [32, Table 1]. In general, collision handling is a known bottleneck in physics-based simulation [3, 15]. We provide suggestions in Section 5.1 to address and alleviate the added computational complexity of the collision handler.

Table 4.1: Performance Timings of Growth Simulations. Each table row provides a snapshot of the performance metrics at a specific frame for a given experiment. The column headers are described as follows. "Frame" refers to the frame of when the performance snapshot was taken. The rendered image that corresponds to the frame is referenced under "Figure". The experiment that was being conducted when the snapshot was taken is under "Experiment". "# Ele" and "# Con" are respectively the number of elements and collision constraints present when the snapshot was taken. The other metrics report the computational time required (in real-life seconds) to carry out a given simulation component in order to generate the single frame. These other metrics and their abbreviations are as follow: diffusion (DIFF), remeshing (MESH), stiffness matrix assembly and force calculation (MTXF), (constrained) FEM system solve (SOLV), collision detection (DECT), collision resolution (RESO), miscellaneous (MISC), and the sum of these metrics (=). Miscellaneous refers to every other metric not mentioned, including rendering. Remeshing is performed every 25 frames; timings reported under MESH are averaged per frame for the last remesh since the given frame. The last column "Elapsed" reports the total time required to simulate the experiment from start to until the performance snapshot was taken.

Experiment	Figure	Frame	# Ele	# Con	DECT	RESO	DIFF	MESH	MTXF	SOLV	MISC	=	Elapsed
Curling Sheet	4.5.B	500	1k	-	0.00	-	0.00	0.00	0.02	0.02	0.00	0.18	1 min
	4.5.C	1000	1k	-	0.14	-	0.00	0.07	0.02	0.02	0.04	0.22	3 min
	4.5.D	1500	1k	368	0.16	0.02	0.00	0.01	0.02	0.40	1.54	2.15	7 min
Delicate Wrinkle	4.3.B	100	4k	-	-	-	0.00	0.92	0.05	0.05	0.09	1.11	14 sec
	4.3.C	200	14k	-	-	-	0.01	1.65	0.16	0.18	0.32	2.32	2 min
	4.3.D	300	30k	-	-	-	0.07	1.55	0.55	0.62	0.81	3.60	8 min
Fruit	4.4.B	666	10k	-	-	-	0.02	0.19	0.18	0.19	0.25	0.83	5 min
	4.4.C	1333	23k	-	-	-	0.05	0.30	0.43	0.48	0.26	1.52	20 min
	4.4.D	2000	32k	-	-	-	0.08	0.67	0.60	0.76	0.81	2.92	46 min
Ripple Cascade	4.2.B	666	18k	-	-	-	0.03	0.21	0.30	0.34	0.44	1.32	10 min
	4.2.C	1333	36k	-	-	-	0.09	0.52	0.67	0.72	0.99	2.99	36 min
	4.2.D	2000	51k	-	-	-	0.15	1.16	0.89	1.06	0.49	3.75	80 min
Ripple Bouquet	4.8.B	833	6k	308	2.56	0.06	0.00	0.03	0.11	1.01	0.19	3.96	0.7 hrs
	4.8.C	1666	8k	340	4.04	0.06	0.00	0.05	0.14	1.40	0.24	5.93	1.8 hrs
	4.8.D	2500	11k	552	6.06	0.19	0.02	0.07	0.19	3.29	0.22	10.04	3.6 hrs
Confined Growth	4.9.B	2500	2k	92	0.24	0.00	0.00	0.00	0.00	0.11	0.11	0.46	0.1 hrs
	4.9.C	3500	5k	1080	1.26	0.55	0.01	0.02	0.01	3.23	0.55	5.63	0.6 hrs
	4.9.D	4000	8k	3276	3.29	19.39	0.01	0.03	0.11	34.71	0.19	57.73	3.7 hrs

CHAPTER 5

CONCLUSION

Solid-shells have not previously been used for simulating large growth deformations. In response, we have developed a new solid-shell growth approach to investigate the potential of simulating large growth using solid-shells. The new solid-shell growth approach entails augmenting solid-shells with plastic embedding and strain-aware adaptive remeshing. Solid-shells offer shell-like decoupling of stretching and bending to handle large bends, while accounting for thickness compression, shearing, and compatibly with existing 3D material laws. Both stretching and bending are captured by the director vectors, allowing both dynamics to share a common plastic procedure, namely plastic embedding. Plastic embedding enables large plastic deformations to occur in solid-shells by deforming the stress-free rest configuration according to the growth/plastic strain. Before deformations emerge or become large, the strain-aware adaptive remesher automatically refines the mesh where needed to maintain a stable and high quality mesh topology. Alongside, the large and intricate plastic deformations are governed and influenced by efficient morphogen diffusion, and a discrete and continuous constraint-based collision handler.

The construction of the growth framework has resulted in a number of novel contributions. These include (1) enabling large plastic deformations in solid-shells via plastic embedding, (2) introducing strain-aware adaptive remeshing to growth simulation, (3) extending the growth tensor description proposed by Kennaway et al. [22] to support plastic bending, and (4) introducing morphogen diffusion and collision handling to growing solid-shells. Altogether, these individual contributions are used to support the primary contribution of enabling and accommodating large plastic deformations in growing solid-shells.

The growth experiments demonstrate a wide range of growth scenarios that the framework can simulate. The augmented solid-shells are capable of handling large fractal patterns, fine delicate wrinkles, and large plastic bending in thin tissue. The solid-shells are also adapt in handling the growth of thick tissue, as shown in the formation of a fruit-like shape. In contrast to similar experiments in the literature, our experiments incorporate strain-aware adaptive remeshing to permit detailed growth starting from a coarse mesh. We also showcase novel experiments of simulating large plastic deformations with collision handling of growing FEM tissue. Altogether, the qualitative experiments conducted demonstrates that the augmented solid-shell are a viable alternative to handle large and intricate plastic deformations of tissue compared to thin-shells.

5.1 Limitations and Future Work

The framework carries limitations that accompany finite element simulations, particularly regarding the element inversion risk and collision handling performance. Element inversion can occur whenever plastic embedding cannot keep up with the build-up of plastic strain, which is often due to excessive morphogen being applied within a short amount of time. Consequently, it is necessary to grow at a slow enough rate such that the rest configuration is given enough iterations to converge into a shape that represents the plastic strain [13]. Otherwise, the plastic strain can accumulate to produce volatile forces.

To address the collision handling performance, we plan to investigate using a technique in which contact constraints are treated as bilateral (i.e. equality) constraints during each simulation step, with separation applied as required between steps, which effectively distributes the LCP solve across the entire simulation to improve simulation efficiency [26]. The collision handling performance could also be improved by parallelizing the LCP solve on the GPU [24].

The thesis is a qualitative investigation of growing solid-shells. Consequently, the experiment results are limited to visual comparisons to similarly simulated growth models in the literature. Providing a comprehensive quantitative comparison of the growing solid-shells to other implemented growth models remains as a future work, as source code for other growth frameworks are largely unavailable (see "Open Source" column in Table 2.1).

5.2 Concluding Remarks

The simulation of growth is relatively more challenging compared simulating elasticity alone. Much of the challenge is attributed to needing to accommodate large and intricate plastic deformations while integrating factors that govern growth, including morphogen diffusion and collision handling. We hope that the growth framework with its new solid-shell growth approach presented allows researchers to readily conduct growth experiments and further advance developments in growth modelling. To assist the developments, the source code of the growth framework presented in this thesis is made publicly available. The framework can particularly be applied and extended to quantitative studies and applications including plant morphology research and appearance modelling.

BIBLIOGRAPHY

[1] D Ambrosi, Gerard A Ateshian, Ellen M Arruda, SC Cowin, J Dumais, A Goriely, Gerhard A Holzapfel, Jay D Humphrey, R Kemkemer, Ellen Kuhl, et al. Perspectives on biological growth and remodeling. *Journal of the Mechanics and Physics of Solids*, 59(4):863–883, 2011.

[2] P Betsch, F Gruttmann, and E Stein. A 4-node finite shell element for the implementation of general hyperelastic 3d-elasticity at finite strains. *Computer Methods in Applied Mechanics and Engineering*, 130(1-2):57–79, 1996.

[3] Robert Bridson, Ronald Fedkiw, and John Anderson. Robust treatment of collisions, contact and friction for cloth animation. In *ACM Transactions on Graphics (ToG)*, volume 21, pages 594–603. ACM, 2002.

[4] Peter B Canham. The minimum energy of bending as a possible explanation of the biconcave shape of the human red blood cell. *Journal of theoretical biology*, 26(1):61–81, 1970.

[5] DR Carter, PR Blenman, and GS Beaupre. Correlations between mechanical stress history and tissue differentiation in initial fracture healing. *Journal of Orthopaedic Research*, 6(5):736–748, 1988.

[6] Tian-Gen Chang, Shuoqi Chang, Qing-Feng Song, Shahnaz Perveen, and Xin-Guang Zhu. Systems models, phenomics and genomics: three pillars for developing high-yielding photosynthetically efficient crops. *in silico Plants*, 1(1):diy003, 2019.

[7] Hsiao-Yu Chen, Arnav Sastry, Wim M van Rees, and Etienne Vouga. Physical simulation of environmentally induced thin shell deformation. *ACM Transactions on Graphics (TOG)*, 37(4):146, 2018.

[8] Richard W Cottle, Jong-Shi Pang, and Richard E Stone. *The linear complementarity problem*, volume 60. SIAM, 1992.

[9] Etienne Couturier, Sylvain Courrech Du Pont, and Stéphane Douady. The filling law: a general framework for leaf folding and its consequences on leaf shape diversity. *Journal of theoretical biology*, 289:47–64, 2011.

[10] Shoubing Dong, Yannan Yan, Liqun Tang, Junping Meng, and Yi Jiang. Simulation of 3d tumor cell growth using nonlinear finite element method. *Computer Methods in Biomechanics and Biomedical Engineering*, 19(8): 807–818, 2016.

[11] Efi Efrati, Eran Sharon, and Raz Kupferman. Elastic theory of unconstrained non-euclidean plates. *Journal of the Mechanics and Physics of Solids*, 57(4):762–775, 2009.

[12] Andreas Enzenhöfer, Sheldon Andrews, Marek Teichmann, and Jozsef Kövecses. Comparison of mixed linear complementarity problem solvers for multibody simulations with contact. In *VRIPHYS*, pages 11–20, 2018.

[13] Charles Gingras and Paul G Kry. Procedural modelling with reaction diffusion and growth of thin shells. In *Proceedings of the 45th Graphics Interface Conference on Proceedings of Graphics Interface 2019*, pages 1–7. Canadian Human-Computer Communications Society, 2019.

[14] Alain Goriely and Martine Ben Amar. On the definition and modeling of incremental, cumulative, and continuous growth laws in morphoelasticity. *Biomechanics and Modeling in Mechanobiology*, 6(5):289–296, 2007.

[15] David Harmon, Etienne Vouga, Rasmus Tamstorf, and Eitan Grinspun. Robust treatment of simultaneous collisions. *ACM Transactions on Graphics (TOG)*, 2008.

[16] Jan Henkel, Maria A Woodruff, Devakara R Epari, Roland Steck, Vaida Glatt, Ian C Dickinson, Peter FM Choong, Michael A Schuetz, and Dietmar W Hutmacher. Bone regeneration based on tissue engineering conceptions - a 21st century perspective. *Bone research*, 1:216, 2013.

[17] Scott J. Hollister. Course Notes for Introduction to Biosolid Mechanics. `http://www.umich.edu/~bme332/ch3strain/bme332straindef.htm`.

[18] Jeffrey A Hubbell. Biomaterials in tissue engineering. *Bio/technology*, 13(6):565–576, 1995.

[19] Dietmar W Hutmacher. Scaffolds in tissue engineering bone and cartilage. *Biomaterials*, 21(24):2529–2543, 2000.

[20] Donald E Ingber. Mechanical control of tissue morphogenesis during embryological development. *International Journal of Developmental Biology*, 50(2-3):255–266, 2003.

[21] Geoffrey Irving, Joseph Teran, and Ronald Fedkiw. Invertible finite elements for robust simulation of large deformation. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 131–140, 2004.

[22] Richard Kennaway, Enrico Coen, Amelia Green, and Andrew Bangham. Generation of diverse biological forms through combinatorial interactions between tissue polarity and growth. *PLoS computational biology*, 7 (6):e1002071, 2011.

[23] Daniel Kierzkowski, Adam Runions, Francesco Vuolo, Sören Strauss, Rena Lymbouridou, Anne-Lise Routier-Kierzkowska, David Wilson-Sánchez, Hannah Jenke, Carla Galinha, Gabriella Mosca, et al. A growth-based framework for leaf shape development and diversity. *Cell*, 177(6):1405–1418, 2019.

[24] Peter Kipfer. LCP algorithms for collision detection using CUDA. *GPU Gems*, 3:723–740, 2007.

[25] Thomas Lecuit and Pierre-Francois Lenne. Cell surface mechanics and the control of cell shape, tissue patterns and morphogenesis. *Nature reviews Molecular cell biology*, 8(8):633–644, 2007.

[26] JE Lloyd, I Stavness, and S Fels. Artisynth: A fast interactive biomechanical modeling toolkit combining multibody and finite element simulation. In *Soft Tissue Biomech Modeling for Computer Assisted Surgery*, pages 355–394. Springer, 2012.

[27] Steve Maas, Dave Rawlins, J Weiss, and G Ateshian. FEBio Theory Manual Version 2, 2014.

[28] DA Matoz-Fernandez, Fordyce A Davidson, Nicola R Stanley-Wall, and Rastko Sknepnek. Wrinkle patterns in active viscoelastic thin sheets. *Physical Review Research*, 2(1):013165, 2020.

[29] Mark Jeffrey Matthews. Physically based simulation of growing surfaces. 2002.

[30] Matthias Müller and Markus Gross. Interactive virtual materials. In *Proceedings of Graphics Interface 2004*. Canadian Human-Computer Communications Society, 2004.

[31] Rahul Narain, Armin Samii, and James F O'Brien. Adaptive anisotropic remeshing for cloth simulation. *ACM transactions on graphics (TOG)*, 31(6):152, 2012.

[32] Rahul Narain, Tobias Pfaff, and James F O'Brien. Folding and crumpling adaptive sheets. *ACM Transactions on Graphics (TOG)*, 32(4):51, 2013.

[33] Miguel A Otaduy, Rasmus Tamstorf, Denis Steinemann, and Markus Gross. Implicit contact handling for deformable objects. In *Computer Graphics Forum*, volume 28, pages 559–568. Wiley Online Library, 2009.

[34] Andrew Owens, Mikolaj Cieslak, Jeremy Hart, Regine Classen-Bockhoff, and Przemyslaw Prusinkiewicz. Modeling dense inflorescences. *ACM Transactions on Graphics (TOG)*, 35(4):136, 2016.

[35] Gregory C Phillips. In vitro morphogenesis in plants-recent advances. *In Vitro Cellular & Developmental Biology-Plant*, 40(4):342–345, 2004.

[36] Przemyslaw Prusinkiewicz and Pierre Barbier de Reuille. Constraints of space in plant development. *Journal of experimental botany*, 61(8):2117–2129, 2010.

[37] Jessica Rosenkrantz and Jesse Louis-Rosenberg. Nervous systems. (2018). `https://n-e-r-v-o-u-s.com`, 2018.

[38] Dominik Schmidt and Katrin Kahlen. Positional variation rather than salt stress dominates changes in 3d leaf shape patterns in cucumber canopies. *in silico Plants*, 1(1):diz011, 2019.

[39] Justin Solomon, Keegan Crane, and Etienne Vouga. Laplace-beltrami: The swiss army knife of geometry processing. In *Symposium on Geometry Processing graduate school (Cardiff, UK, 2014)*, volume 2, 2014.

[40] Yasuhiko Tabata. Tissue regeneration based on growth factor release. *Tissue engineering*, 9(4, Supplement 1): 5–15, 2003.

[41] Larry A Taber and Jay D Humphrey. Stress-modulated growth, residual stress, and vascular heterogeneity. *Journal of biomechanical engineering*, 123(6):528–535, 2001.

[42] Seiji Takeda, Akira Iwasaki, Noritaka Matsumoto, Tomohiro Uemura, Kiyoshi Tatematsu, and Kiyotaka Okada. Physical interaction of floral organs controls petal morphogenesis in arabidopsis. *Plant physiology*, 161(3): 1242–1250, 2013.

[43] XH Tang, A Paluszny, and RW Zimmerman. Energy conservative property of impulse-based methods for collision resolution. *International Journal for Numerical Methods in Engineering*, 95(6):529–540, 2013.

[44] Alan Mathison Turing. The chemical basis of morphogenesis. *Bulletin of mathematical biology*, 52(1-2):153–197, 1990.

[45] RA Fontes Valente, RJ Alves De Sousa, and RM Natal Jorge. An enhanced strain 3d element for large deformation elastoplastic thin-shell applications. *Computational Mechanics*, 34(1):38–52, 2004.

[46] Roman Vetter. *Growth, interaction and packing of thin objects*. PhD thesis, ETH Zurich, 2015.

[47] Roman Vetter, Norbert Stoop, Thomas Jenni, Falk K Wittel, and Hans J Herrmann. Subdivision shell elements with anisotropic growth. *International journal for numerical methods in engineering*, 95(9):791–810, 2013.

[48] Tomonari Watanabe, Jim S Hanan, Peter M Room, Toshihiro Hasegawa, Hiroshi Nakagawa, and Wataru Takahashi. Rice morphogenesis and plant architecture: measurement, specification and the reconstruction of structural development by 3d architectural modelling. *Annals of botany*, 95(7):1131–1143, 2005.

[49] Martin Wicke, Daniel Ritchie, Bryan M Klingner, Sebastian Burke, Jonathan R Shewchuk, and James F O'Brien. Dynamic local remeshing for elastoplastic simulation. In *ACM Transactions on graphics (TOG)*, volume 29, page 49. ACM, 2010.

[50] Jie Yin, Zexian Cao, Chaorong Li, Izhak Sheinman, and Xi Chen. Stress-driven buckling patterns in spheroidal core/shell structures. *Proceedings of the National Academy of Sciences*, 105(49):19132–19135, 2008.

[51] Yonggang Zheng, Jianhua Wang, Hongfei Ye, Yin Liu, and Hongwu Zhang. A solid-shell based finite element model for thin-walled soft structures with a growing mass. *International Journal of Solids and Structures*, 163: 87–101, 2019.

[52] Alexander M Zöllner, Maria A Holland, Kord S Honda, Arun K Gosain, and Ellen Kuhl. Growth on demand: reviewing the mechanobiology of stretched skin. *Journal of the mechanical behavior of biomedical materials*, 28:495–509, 2013.

Appendices

CHAPTER A
SUPPLEMENTARY MATERIAL FOR SOLID-SHELL ELEMENT

A.1 Integration Points

Each triangular solid-shell element has a set of 9 integration points. The integration points are distributed such that the bottom, middle, and top portion of the element are assigned 3 integration points each. The local coordinates (ξ_1, ξ_2, ξ_3) and weight w for each of the 9 integration points are

$$\begin{array}{llllll} \xi_1 = (a,b,-b) & w_1 = ac_1 & \xi_4 = (a,a,0) & w_4 = ac_2 & \xi_7 = (a,a,b) & w_7 = ac_1 \\ \xi_2 = (b,a,-b) & w_2 = ac_1 & \xi_5 = (b,a,0) & w_5 = ac_2 & \xi_8 = (b,a,b) & w_8 = ac_1 \\ \xi_3 = (a,b,-b) & w_3 = ac_1 & \xi_6 = (a,b,0) & w_6 = ac_2 & \xi_9 = (a,b,b) & w_9 = ac_1 \end{array}$$

where

$$a = \frac{1}{6} \qquad b = \frac{2}{3} \qquad c_1 = \frac{5}{9} \qquad c_2 = \frac{8}{9}.$$

A.1.1 Warping Point

We define a special integration point, separate from the 9 integration points of the triangular solid-shell element. This special integration point is referred as the *warping point* and resides in the center of the element volume. For the triangular solid-shell element, the local coordinates of the warping point is

$$\boldsymbol{\xi} = \left(\frac{1}{2}, \frac{1}{2}, 0\right). \tag{A.1}$$

The warping point is only used for the purpose of measuring quantities, which are normally evaluated at every integration point, that is best representative of the entire element. For example, the deformation gradient $\mathbf{F}$ that is best representative of a given element can be measured by evaluating the deformation gradient formula (equation (3.2)) using the local coordinates of the element's warping point.

A.2 Covariant and Contravariant Basis Vectors

For a triangular solid-shell element, the Jacobian matrix is constructed from column vectors $(\mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3)$:

$$\begin{aligned} \mathbf{g}_\alpha = \frac{\partial \mathbf{x}}{\partial \xi_\alpha} &= \sum_a^3 \frac{\partial \mathrm{N}_a(\xi_1,\xi_2)}{\partial \xi_\alpha}\left(\frac{1+\xi_3}{2}\mathbf{x}_a + \frac{1-\xi_3}{2}\mathbf{y}_a\right) \\ &= \sum_a^3 \frac{\partial \mathrm{N}_a(\xi_1,\xi_2)}{\partial \xi_\alpha}\left(\mathbf{x}_a - \frac{1-\xi_3}{2}\mathbf{d}_a\right) \end{aligned} \qquad \begin{aligned} \mathbf{g}_3 = \frac{\partial \mathbf{x}}{\partial \xi_3} &= \sum_a^3 \frac{1}{2}\mathrm{N}_a(\xi_1,\xi_2)\,(\mathbf{x}_a - \mathbf{y}_a) \\ &= \sum_a^3 \frac{1}{2}\mathrm{N}_a(\xi_1,\xi_2)\,\mathbf{d}_a \end{aligned} \tag{A.2}$$

where $\alpha \in \{1,2\}$ and other variables being defined earlier in section 3.3.1. These vectors are referred as covariant basis vectors as they can *co-vary* with the changes in local coordinate bases. For example, if the element's local coordinate bases were to be modified by multiplying the bases by some matrix, the covariant basis vectors would need to be multiplied by the same matrix.

The column vectors that comprise the inverse Jacobian matrix can be regarded as the contravariant basis vectors and are denoted as $(\mathbf{g}^1, \mathbf{g}^2, \mathbf{g}^3)$. Compared to covariant vectors, these contravariant vectors *contra-vary* with the changes of the element's local coordinate bases, meaning that if the local coordinate bases were to be transformed by some matrix, the contravariant vectors would need to be multiplied by the inverse of the matrix.

The primary reason for constructing the Jacobian in terms of basis vectors is that it allows for the gradient formulation to vary depending on the direction taken across the local coordinates. In particular, the gradient between the top and bottom face of the solid-shell element at a given (ξ_1, ξ_2) is set to be constant by making the 3rd covariant basis vector independent of the local thickness coordinate ξ_3. In effect, this simplifies the solid-shell with a linearized thickness.

A.3 Gradient Factors

In the elastic shell force and stiffness matrix equations, the gradient factor that is found

$$\text{grad}\left(\frac{1+\text{face}(b)\,\xi_{3,k}}{2}\,\text{N}_{s(a)}\right)$$

can be expanded as

$$\frac{1}{2}\left((1+\text{face}(a)\,\xi_{3,k})\,\text{grad}(\text{N}_{\text{s}(a)}) + \text{face}(a)\,\text{N}_{\text{s}(a)}(\xi_{1,k},\xi_{2,k})\,\mathbf{g}^3\right).$$

Recall that face(i) is 1 or -1, depending if node a belongs to the front or back face respectively; $\text{N}_{s(a)}$ is the $s(a)$-th 2D shape function where $s(a)$ is defined in equation (3.12); and $\mathbf{g}^3$ is the 3rd contravariant basis vector of the element, which is calculated using the element's node positions in world-space and the local coordinates of integration point i. The grad$(N_{\text{s}(a)})$ factor expands into

$$\frac{\partial \text{N}_{\text{s}(a)}(\xi_{1,k},\xi_{2,k})}{\partial \xi_1}\mathbf{g}^1 + \frac{\partial \text{N}_{\text{s}(a)}(\xi_{1,k},\xi_{2,k})}{\partial \xi_2}\mathbf{g}^2$$

where each term is a derivative of shape function $\text{N}_{\text{s}(a)}$ that is evaluated using the first two local coordinates of integration point k and subsequently multiplied by its respective contravariant basis vector.

A.4 Polar Decomposition

The deformation gradient (or strain tensor) can be decomposed into a symmetrical and rotational factor. Given a tensor $\mathbf{M}$ to be decomposed, the *right polar decomposition* of the tensor is defined as

$$\mathbf{M} = \mathbf{RP}$$

where **R** and **P** are the rotational and symmetrical factors respectively, and can be computed using existing matrix libraries. Conceptually, using the deformation gradient as an example, its rotational factor describes how the deformable body should be rotated in-place before the symmetrical factor stretches/compresses the body.

CHAPTER B
SUPPLEMENTARY MATERIAL FOR ADAPTIVE REMESHING

B.1 Element-Based Gradient

The analysis stage of the adaptive remesher operates on gradients that are defined on an element basis rather than an integration point basis. Suppose the element-based gradient of an arbitrary quantity $\mathbf{x}$ of a solid-shell element is formulated as

$$\nabla \mathbf{x} = \mathbf{J}\mathbf{J}_0^{-1}.$$

The 3×3 Jacobian $\mathbf{J}$ is computed using equations (3.11) and (A.2) by substituting $\boldsymbol{\xi}$ with the local coordinates of the element's warping point and by substituting both $\mathbf{x}_a$ and $\mathbf{y}_a$ with the arbitrary quantity that corresponds to the a-th front node and a-th back node respectively. For instance, $\mathbf{x}_a$ and $\mathbf{y}_a$ can be set to the velocity of the a-th front node and a-th back node respectively if one were to compute the velocity gradient. Both $\mathbf{x}_a$ and $\mathbf{y}_a$ can be set to the same quantity, such as the vertex normal of the a-th front node, if the gradient is intended to be invariant across the front and back nodes.

The $\mathbf{J}_0$ is the rest Jacobian, which is calculated in the same matter as $\mathbf{J}$, but using quantities that the gradient is respect to. For example, $\mathbf{J}_0$ would use reference-space node positions during its calculation in order to compute a gradient that is respect to the reference-space node positions.