STEPHEN THOMAS GOODE

# DEVELOPMENT OF A SPINAL CORD INJURY MODEL USING THE MATERIAL POINT METHOD

# DEVELOPMENT OF A SPINAL CORD INJURY MODEL USING THE MATERIAL POINT METHOD

STEPHEN THOMAS GOODE
INSTITUTE OF MEDICAL AND BIOLOGICAL ENGINEERING
SCHOOL OF MECHANICAL ENGINEERING
UNIVERSITY OF LEEDS


Submitted in accordance with the requirements for the degree of

*Doctor of Philosophy*

September 2016

# DECLARATION

The candidate confirms that the work submitted is his own, except where work which has formed part of jointly-authored publications has been included. The contribution of the candidate and the other authors to this work has been explicitly indicated overleaf. The candidate confirms that appropriate credit has been given within the thesis where reference has been made to the work of others.



*September 2016*

Stephen Thomas Goode

# ACKNOWLEDGMENTS

My thanks to:

- my supervision team, Dr Jon Summers, Prof Richard Hall, and Prof Joanne Tipper for their support and guidance throughout this study
- the Leeds Spine Group and the Leeds Virtual Group for their advice and support
- Ms Jodie Jones for her kind support and artistic talent
- Dr Mark Spillman for his moral and immoral support
- the IT Support team at the University of Leeds for their technical prowess
- everyone at the Institute of Medical and Biological Engineering for creating a wonderful working environment
- everyone else who has helped and hindered me along the way, it's been quite a journey

This research was funded by the EPSRC Doctoral Training Centre in Tissue Engineering and Regenerative Medicine, a collaboration between the Universities of Leeds, Sheffield and York. Grant number EP/500513/1.

# ABSTRACT

Spinal cord injury (SCI) is characterised by permanent loss of motor and sensory function. The primary damage from the initial mechanical insult is exacerbated by the secondary patho-physiological cascade. Research into neuroprotective interventions to preserve tissue and reduce the damage caused by the secondary injury is hampered, in part, due to a lack of understanding of the link between the biomechanics of the primary traumatic injury and the subsequent evolution of the secondary injury. Hence, there is a need to better understand the biomechanics of SCI, the distinct injury patterns produced, and how these affect the evolution of the secondary cascade.

Computational models using finite element methods (FEM) have been established as a useful tool for investigating SCI biomechanics. These may be used to obtain data that is difficult or impossible to capture through in vivo and in vitro experiments, in particular; stress and strain fields within the neural tissue. However, the complexity of these models is limited by difficulties. These include: problems coping with large deformations over short periods of time due to mesh tangling, difficulties in incorporating the fluid structure interactions, and scalability issues when attempting to make use of high performance computing facilities, utilising large numbers of processors.

This work has involved the creation of a computational spinal cord injury using the Material Point Method (MPM) and MPMICE (MPM for Implicit, Continuous Fluid, Eulerian), alternative computational methods that overcome these limitations. The model incorporates the neural spinal cord tissue, the dura mater, and the cerebrospinal fluid. This model has been validated against equivalent experimental and FEM results. MPM/MPMICE was found to be a viable alternative to FEM for modelling SCI computationally, with the potential to enable more complex and anatomically detailed models through the utilisation of increased parallel computation.

# CONTENTS

# LIST OF FIGURES

## LIST OF TABLES

# 1

# INTRODUCTION AND LITERATURE REVIEW

## 1.1 INCIDENCE OF SPINAL CORD INJURY

The annual incidence of traumatic spinal cord injury (SCI) ranges from 11.5 to 53.4 per million population for developed countries [1]. In the South of England the recorded incidence of spinal trauma with risk of SCI is approximately 16 per million population per year (approximately 480 people per year based on the current population), with actual spinal damage occurring in 10-12 people per million per year (approximately 300-360) [2]. The primary cause of this type of injury is road traffic accidents, followed by occupational and sporting activities, then by acts of violence (Table 1). Young males comprise the majority of the victims, the male to female ratio is typically around 4:1, two thirds of victims are below the age of 30 [1]. While this type of injury affects a relatively small number of people, the effect is on patients is catastrophic, resulting in permanent loss of motor and sensory function [3]. SCI patients go on to suffer from secondary medical complications, most commonly; pressure ulcers, autonomic dysreflexia, pneumonia and atelectasis [4]. The average lifetime medical expenses incurred by an individual suffering from SCI ranges from between US$500,000 and US$2,000,000, dependent on the severity of the injury and subsequent disability [5]. Note that these figures were published in 1997, the present costs are likely to be higher.

Traumatic spinal injuries often result in damage to the spinal cord and a complete or partial neurological deficit. Vertebral dislocation is the most widely observed form of spinal injury, with a prevalence

| Cause of Injury | Incidence (%) |
|---|---|
| Traffic accidents (motor vehicle, bicycle, pedestrian) | 40 – 50 |
| Work | 10 – 25 |
| Sports and recreation | 10 – 25 |
| Falls | 20 |
| Violence | 10 – 25 |

Table 1: Etiology of Adult SCI [1]. These are general figures, causes of SCI are affected by underlying cultural issues and vary substantially between countries.

of approximately 45% [6, 7], where the spinal cord is sheared between adjacent vertebral segments. Vertebral burst fractures, where bone fragments are forced into the spinal canal, result in initial axial contusion to the cord, followed by chronic cord compression. This type of injury is prevalent in between 30-48% of spinal injury cases in developed countries [7]. Flexion distraction or extension distraction of vertebrae stretches the spinal cord, subjecting it to excessive tensile stress. The cord may also be subjected to axial tension resulting from distraction injuries. Cord tethering, where the cord is pulled after becoming tethered to an immobile part of the lower spine, can contribute to SCI [6]. Transection of the cord may also occur, although catastrophic, complete transection occurs very rarely [8]. Of these injuries, the burst fracture has been the subject of a majority of the studies using *in vitro* and computational techniques, as it is a well defined injury and readily recreated experimentally [9–11]. In addition SCI may also occur as the result of non-traumatic lesions that cause chronic mechanical compression. Examples include congenital and developmental disorders such as spinal bifida, rheumatologic and degenerative conditions such as spondylosis, ossification of the posterior longitudinal ligament, and tumours [12, 13].

The mechanical behaviour of the spinal cord is yet to be fully understood, particularly where traumatic loading is concerned. *In vitro*

studies have utilised human, animal, and synthetic materials that are mechanically representative of human spinal cord tissues. There is evidence showing significant correlation between the mechanics of the initial injury and the resultant neurological damage [6, 7, 14–19].

## 1.2 SPINAL ANATOMY

### 1.2.1 *Spinal Vertebral Column*

The vertebral column extends from the cranium to the apex of the coccyx. It is the main component of the axial skeleton, and supports the weight of the human body superior to the pelvis. The column is comprised of 33 spinal vertebrae and is subdivided into 5 regions: 7 cervical, 12 thoratic, 5 lumbar, 5 sacral, and 4 coccygeal. Of these, the upper 33 vertebrae (cervical, thoracic, and lumbar) are articulating whilst the lower 9 vertebrae (sacral and coccygeal) are fused. Significant motion only occurs in the upper 24 vertebrae, movement in the pre-sacral column is facilitated by semi flexible intervertebral discs, which join the articulating vertebrae. The sacrum transfers weight from the vertebral column to the pelvic girdle via the sacroiliac joints. In addition to its structural role, the vertebral column helps to protect nerves and the spinal cord [20–22].

The morphology of the spinal vertebrae varies between levels, with structural differences relating to the load supported. Vertebrae increase in size towards the bottom, reaching maximum size before the sacrum. A typical vertebrae consists of a vertebral body, vertebral arch, and seven spinal processes, Figure 1. The vertebral body (anterior) provides strength and supports the spinal cord and the weight carried by it. The vertebral arch is anchored to the posterior surface of the vertebral body by two pedicles, forming the lateral pillars, the

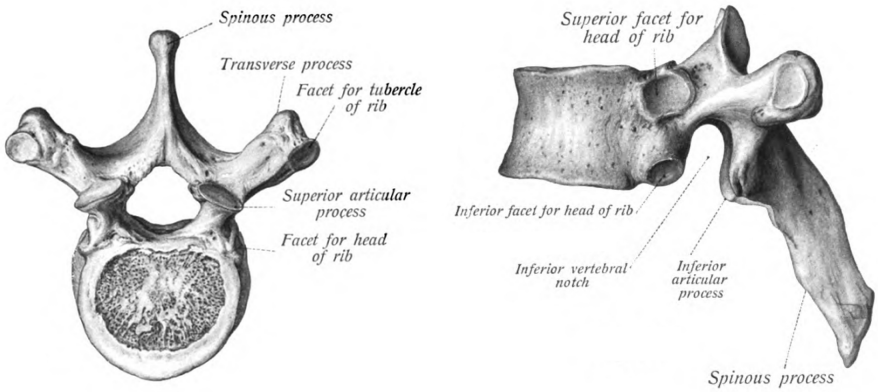



Figure 1: Illustration showing the anatomy of a human vertebra [23].

roof of the arch is formed by the left and right laminae, which fuse at the midline. The arches align to form the lateral and posterior walls of the vertebral foramen, the succession of which form the vertebral canal, containing the spinal cord, meninges, nerve roots and vessels. The epidural space, formed between the walls of the vertebral canal and the the dura mater, which envelops the spinal cord, is filled with fat and a venous plexus [20–22].

One median spinous process projects posteriorly from the vertebral arch, where the laminae meet the pedicle, these typically overlap with the inferior vertebrae. Two transverse processes extend laterally from the junctions between the two pedicles and laminae. The median and transverse processes provide attachment sites for ligaments and muscle and serve as levers for the action of the muscles against the vertebrae. Four articular processes also arise from the junction between the pedicles and the laminae, two superior and two inferior, these provide articulation sites (facets) with the adjacent vertebrae. The interlocking of the adjacent articular processes helps to keep the vertebrae aligned and determines the range of movements permitted between the neighbouring vertebrae of each region [20–22].

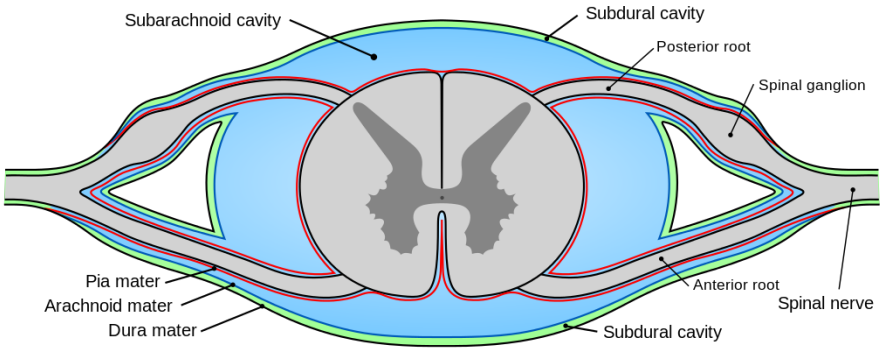



Figure 2: Diagram showing a transverse section of the medulla spinalis and its membranes [25].

### 1.2.2 *Meninges*

The meninges are a system of protective membranes that envelop the CNS, consisting of three layers: the pia mater (innermost layer), arachnoid mater, and dura mater (outermost layer), Figure 2. The pia mater is a thin layer of fibrous tissue that adheres to the surface of the spinal cord. Arachnoid mater is a cobweb-like material, interposed between the pia and dura, it attaches to the inner surface of the dura. The arachnoid trabeculae, delicate fibres of connective tissue, span the subarachnoid space and connect the adjacent arachnoid mater and pia mater. The dura is the tough, fibrous, outermost meningeal layer, that encloses the brain and spinal cord, it forms a thick tube around the cord. The subarachnoid space is filled with cerebrospinal fluid. The cord is suspended within the dural sheath by denticulate ligaments, running downwards longitudonally, arranged in pairs on either side. These ligaments are comprised of fibrous sheets of pia mater, they extend between the anterior and posterior nerve roots, attaching to the internal surface of the dural sac. The dura and the cord are anchored to the spinal column via the spinal nerves, two nerves project from each side of the cord at each level [20, 21, 24].

### 1.2.3 *Cerebrospinal fluid*

Cerebrospinal fluid (CSF) is a clear, colourless, slightly alkaline fluid with a low specific gravity. Its composition differs from that of blood plasma, CSF contains no blood cells and, by comparison, there is a very low concentration of proteins. CSF is continually secreted by the choroid plexus, a structure of ventricles in the brain, it flows into the subarachnoid space though the median and lateral apertures of the fourth ventricle, surrounding the brain and spinal cord, and is reabsorbed through granulations and specialised villi on the arachnoid membrane, which project out from the arachnoid membrane. The CSF provides buoyancy to the brain, reducing its weight and reducing the pressure on nerves and blood vessels. In addition, it has been shown to protect the brain and spinal cord during impacts [14, 26]. Further to physical protection, the CSF also provides chemical protection, regulating the extracellular environment for neurons, it exchanges solutes with the interstitial fluid and removes waste products. Ions and nutrients are transported from the blood to the CSF via specialised regions in the walls of the ventricles, the osmotic gradient draws across water along with solutes [20, 22, 24, 27].

### 1.2.4 *Blood brain barrier*

The blood brain barrier (BBB) is a functional barrier between the interstitial CNS fluid and the blood. It is not a literal barrier, rather it refers to the selective permeability of CNS capillaries, which shelter it from toxins and fluctuations in ion, hormone, and neauroactive substance levels. The CNS tissue itself effectively creates the BBB, CNS capillaries are less permeable than other capillaries. In other epitheliums the anchoring junctions between endothelial cells leave pores through

which certain molecules can pass. In the BBB tight junction formation between endothelial cells of CNS vessels is induced via paracrines from adjacent astrocytes; astrocytic foot processes surround the capillary. Selected carriers transport materials over the barrier, if a water soluble molecule is unable to be transported across by a carrier then it is unable to cross the the BBB. Small lipid soluble molecules (such as oxygen, carbon dioxide, and ethanol) are able to diffuse through lipid bilayers, and can cross the BBB. The BBB is lacking in some small regions, known as the circumventricular organs, including the posterior pituitary and the medulla oblongata vomiting centre, whose autonomic regulatory functions require a blood supply in absence of the BBB [20, 27].

### 1.2.5 *Spinal cord*

The central nervous system (CNS) comprises of the spinal cord and the brain, some classifications also include the optic, auditory, and olfactory systems. The CNS conducts and interprets signals and provides excitatory signals to the peripheral nervous system (PNS). The PNS includes the cranial nerves originating from the brain and from the spinal cord, and the sensory nerve bodies. The PNS innervates muscles and transmits signals to and from the spinal cord [28]. Housed and protected within the spinal vertebral column, the spinal cord is the primary pathway through which the brain is connected to the rest of body. The spinal cord begins as an extension of the medulla oblongata in the brain and extends longitudonally in the cranialcaudal direction. The spine is divided into sections, Figure 3, from top to bottom these are the Cervical, Thoracic, Lumbar, Sacral, and Coxal sections. Nerves project symmetrically in pairs from each side of the cord at each level (C.8, T.12, L.5, S.5, Co.1) [24], to innervate each

side of the body these subdivide into ventral and dorsal roots. Ventral roots conduct motor signals to muscles, glands, etc. from the spinal cord. Dorsal roots transmit incoming sensory information to the spinal cord [22]. The spinal cord is composed mainly of dendrites, axons, and cell bodies, organised into longitudinally oriented tracts of grey and white matter, Figure 3. The grey matter at the centre of the cord mostly consists of the cell bodies of excitatory cells, glial cells, and blood vessels. It is surrounded by white matter, which helps in insulating and protecting the spinal cord. White matter consists of axons, astrocytes, dendrocytes, and microglia. The astrocytes contribute to the blood-nerve barrier that separates the CNS from blood proteins and cells. Fascicles of axons project from the white matter, through the bone casing, to the PNS-CNS interface [22, 29]. The cross sectional area of white matter decreases caudally as there are fewer descending fibres, more having branched off at the superior levels.The cross sectional area of grey matter is indicative of the number of neurons at a spinal level, it is greatest at the levels responsible for supplying signals to and from the limbs.[20–22, 24, 27, 28].

On a cellular level, the nervous system is comprised of two main cell types, neurons and neuroglia. Neurons are the main structural elements and are comprised of the cell body (soma) and its extensions (axons and dendrites). Dendrites transmit electrical signals to the soma, while axons conduct impulses away. Neuroglia (glial cells) are support cells that aid the function of neurons, and are more abundant than neurons. In the PNS the auxiliary cells are Schwann cells, and in the CNS they are astrocytes and myelinating oligodendrocytes. Axons in the CNS are surrounded by an insulating myelin sheath, created by dense wrappings of oligodendrocytes. Myelin plays a crucial role in nerve function, it increases the propagation velocity of neural impulses in the axons, which is particularly important for long axons [24, 27, 29].

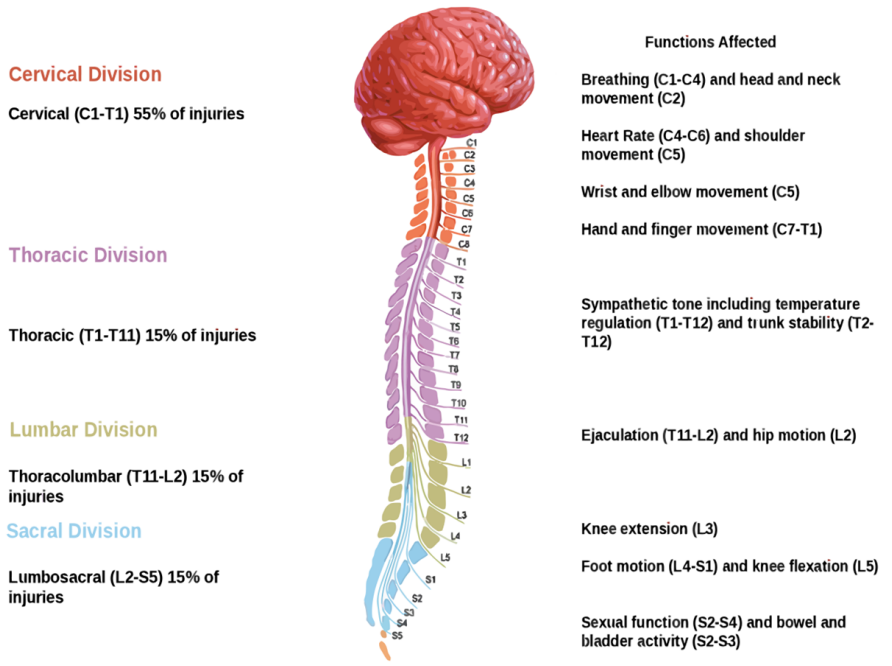
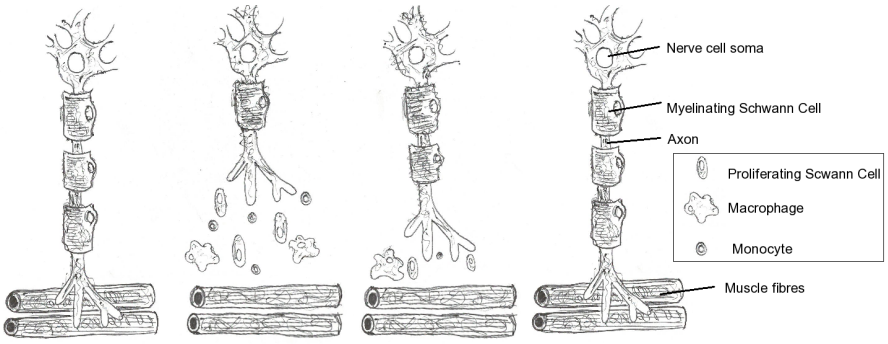


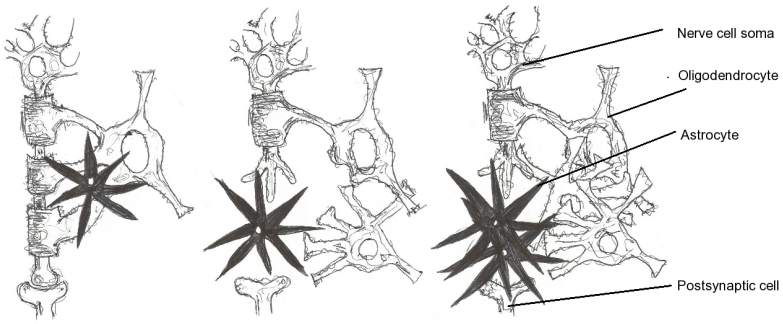

Figure 3: Anatomy of the Human Central System including the brain and spinal cord, divided into the Cervical, Thoracic, Lumbar, Sacral, and Coxal sections. Adapted from Human Anatomy Chart [30].

## 1.3 SPINAL CORD INJURY

Nerves have a poor capacity for regeneration, as neurons do not undergo mitosis; however, they are capable of forming new extensions and regenerate severed sections under certain circumstances [31]. Axon regeneration in the CNS following injury is limited, and poor at restoring function. Glycoproteins in the extracellular environment, such as myelin, inhibit axon growth. In the PNS, macrophages, Schwann cells, and monocytes all work together to promote regeneration by removing debris, and leading new axons to their synaptic targets. In the CNS macrophages infiltrate the site of the injury much more slowly, inhibited by the blood-spine barrier delaying the clean-up of myelin debris [32, 33]. In the PNS cell adhesion molecules in the distant end of the nerve are upregulated, this does not occur in the CNS and macrophage recruitment is reduced. Astrocytes proliferate following the injury, becoming reactive astrocytes which contribute




(a) PNS




(b) CNS

Figure 4: Responses to axotomy (severing of an axon) in the PNS and the CNS (spinal cord), adapted from [31, 36]. The leftmost diagrams show the uninjured nerve configurations, the rightmost diagrams show the post healing nerve configurations. Note that in the CNS the axon is unable to reach its synaptic target due to the formation of a glial scar.

to the formation of a glial scar, partially made up from myelin and cell debris, Figure 4. The neurons that survive the initial injury will attempt to regenerate axons, but these will be blocked from reaching their synaptic targets by the glial scar, which inhibits axon growth and myelination [33, 34]. The formation of cavities and cysts within the spinal cord can interrupt ascending and descending axonal tracts, blocking the transmission of neural signals [35].

### 1.3.1 *Primary and Secondary Injury*

Traumatic SCI may be divided into two phases, the primary injury triggers a secondary pathophysiological cascade. The primary injury concerns direct injury of neural elements from an initial mechanical trauma [37]. The spinal cord tissue is subjected to damaging traction and compressive forces, which may come as a result of displaced bone and disc material. The initial mechanical insult causes neuron death, axon and blood vessel damage, and micro-haemorrhaging in the grey matter. Toxins are released from disrupted cell membranes. The spinal cord experiences swelling to the limits of the spinal canal, enacting additional compressive force on the spinal tissue. Ischemia occurs when the pressure exceeds venous blood pressure, restricting bloodflow to the affected tissue. Electrolyte shifts, in combination with toxins from damaged cells, and ischemia, trigger the secondary injury [13, 38].

The secondary injury occurs over the hours and weeks following the primary injury. Acute peripheral circulatory failure, hyperperfusion, develops in the grey matter and then spreads to the surrounding white matter [39], acting to slow or stop axon propagation [13]. The toxins released during the primary injury damage more cells around the injury site. Glutamate, in particular, is highly disruptive. It plays a key role in excitotoxicity, binding in high quantities to receptors on target neurons leading to excessive neurotransmitter stimulation inducing an influx of calcium ions, killing the cells [38]. In addition to furthering neuron death, excitotoxicity has also been found to kill oligodendrocytes, leading to the demyelination of axons, disrupting their function [40, 41]. Haemorrhage and calcium ion influx have been linked to numerous degenerative processes and poor functional outcomes [6]. In addition to necrosis, studies have also suggested that

apoptosis occurs on a significant scale, compounding the damage to CNS tissue. A study in rats found evidence of spinal cord cell apoptosis following chronic mechanical cord compression, concluding that apoptosis may play a role in the pathogenesis of motor paresis resulting from the destruction of spinal cord tissue [12]. Another study in rats and monkeys found apoptotic cells from 6 hours to 3 weeks, after the initial time of injury, particularly in the white matter [42]. Oligodendrocyte apoptosis may occur as far as four spinal segments away from the initial point of injury. Chronic demyelination of axonal tracts appears to result in part from apoptosis during the secondary injury phase. CNS injury therefore appears to share some common features with degenerative diseases such as multiple sclerosis [42].

### 1.3.2 *Investigating SCI biomechanics*

Biomechanics is the application of engineering principles in the context of biological systems, commonly the human body [43]. In relation to SCI biomechanical studies have focussed on the mechanical aspects of the injury and the effects these have. Treatment strategies for SCI have largely focus on mitigating the secondary pathological cascade. There may be a short time window following the infliction of the primary injury for therapies to reduce the neurological damage resulting from the secondary injury [6, 44]. Most strategies to date focus on specific pathways of damage in grey and white matter. Axon disruption in white matter is thought to be the most significant contributor to the eventual clinical deficits [18, 45, 46]. While a wide range of neuroprotective strategies have been investigated, the clinical translation of these strategies has not significantly improved patient outcomes [44, 47, 48]. The mechanical behaviour of the spinal cord is yet to be fully understood, particularly where traumatic load-

ing is concerned [14]. Progress in SCI treatment is hindered by a lack of understanding about the biomechanics of spinal cord injuries, the distinct injury patterns produced, and how these injury patterns affect the evolution of the secondary cascade. There is, therefore, a need for further investigation into the biomechanics of traumatic SCI.

Different modes of injury have been shown to produce distinctly different injury patterns. There is evidence showing significant correlation between the initial mechanical injury and the resultant neurological damage[14]. *In vitro* cell culture experiments have found traumatic neuronal injury is influenced by the loading pattern [49]. *In vitro* studies have also utilised human, animal, and synthetic materials that are mechanically representative of human spinal cord tissues [9, 14]. *In vivo* injury models using rats have shown distinct and varied patterns of neural tissue damage for different mechanisms of injury [6, 7, 15, 16, 19]. The pathology of the secondary injury is extremely complex, and it is likely that successful treatments will require a series of interventions [48]. Bunge and Pearse recommend that after preventing secondary tissue loss, through the use of anti-inflammatory and immunomodulatory interventions, regeneration strategies should focus on the following factors: reduction of scar formation, diminishing the accumulation of proteoglycan molecules inhibitory to axonal growth; overcoming additional inhibitory molecules, including myelin, that can prevent axonal extension; stimulating damaged nerve cells to regenerate axons; providing sustenance to the nerve cells separated from their targets; facilitating and guiding axonal growth across the site of injury; enabling formation of new connections and, finally, retraining the nervous system to use the therapeutic interventions [48]. While a wide range of neuroprotective strategies have been investigated [44, 48], the clinical translation of these strategies has not significantly improved patient outcomes. Animal models do not represent the full spectrum of SCI

in humans, additional models are needed to aid the development and evaluation of new preclinical strategies [6, 50]. Computational models complement animal and *in vitro* studies, and allow the capture of data that is not easily measurable using lab based techniques. Finite element methods (FEM) of the human spine and spinal cord have been used to measure stress and strain patterns within the cord itself [14, 51, 52]. Distinctly different stress and strain patterns in the cord tissue in response to varying biomechanical injury mechanisms have been demonstrated [14, 18, 53, 54].

#### 1.3.2.1 *Cell Culture Models*

Cell culture models have been used to investigate certain aspects of the biological and mechanical mechanisms involved in traumatic neuronal injury *in vitro*, this is relevant to both SCI and traumatic brain injury. Cell culture models have been used to simplify injury mechanics in order to investigate specific cell interactions under specific physiological conditions. Geddes-Klein et al. [49] investigated the cellular response of CNS tissue to the mechanical injury, using cultured primary cortical rat neurons [49]. The neurons were cultured in 2D on an elastic substrate, the substrate was then subjected to uniaxial and biaxial stretches (of 0, 10, 30, and 50%), imparting a mechanical insult to the cells. The different loading mechanisms resulted in different responses in the secondary excitotoxic injury mechanisms, while $Ca^{2+}$ ion increases were observed in both cases, the increase was substantially larger in the case of biaxial stretch. In the case of the uniaxially stretched neurons, $Ca^{2+}$ ion transients were blocked by specific channel antagonists, whereas a significant $Ca^{2+}$ ion transient remained following biaxial stretch. The authors suggested that this increased $Ca^{2+}$ ion influx may be due to pores/tears forming in the cell membranes of the biaxially stretched neurons as these cells showed signif-

icant carboxyfluorescein uptake, a molecule that is usually unable to penetrate the cell membrane.

Cullen and LaPlaca studied the response of both 2D and 3D cell culture configurations to high rate mechanical deformation [55]. Primary cortical rat neurons were cultured in a 3D bioactive matrix, cells homogeneously dispersed throughout the matrix, and in a monolayer, sandwiched between two layers of bioactive matrix, a 3D cell shearing device was then used to apply mechanical deformation. Greater levels of cell death were observed in the 3D culture than the 2D culture. Computer simulations were used to predict the local cellular strains. In the 3D culture a heterogeneous strain field was observed, comprising of compressive, shear, and tensile strains. A simpler strain pattern was observed in the 2D culture, comprised mostly of shear strain. The results suggest that the susceptibility of neurons to mechanical loading differs between 2D and 3D culture configurations, potentially due to differences in local cellular strain.

Tissue engineering models can be used to bridge the gap between cell culture models and animal models [56]. Type-I collagen is the preferable 3D cell culture material, as it is a major component in the natural tissues and supports the survival of most cell types [57]. East et al. developed a 3D tissue model using type-I collagen hydrogels, which allows cell interactions within the gel to be studied and provides a tool for investigating astroglial scar response [58]. Phillips and Brown described a range of techniques using 3D cell seeded type-I collagen gels, in which the mechanical cues imparted to the cells can be controlled and manipulated [57]. Key parameters including alignment, density, stiffness, and strain, can be controlled, making it possible to predict and understand the role of mechanical cues on cell behaviour. These approaches may be used for exploring a wide range of cellular responses, including those relating to SCI.

#### 1.3.2.2 *In vivo Animal Models*

The most widely used contusion method for the experimental study of SCI to date is based on a technique pioneered by Allen in 1911 [59]. This involves the use of a mechanical device to drop a weight from height onto the exposed spinal cord dorsal surface of a rat, simulating a crush injury to the spinal column. The cord is exposed by performing a laminectomy of the vertebrae corresponding to the section of the cord that is to be tested. The height and drop-weight properties are controlled variables that produce a predicable force, quantifying the dynamics of the injury, allowing reproducible injuries to the spinal cord to be produced experimentally [16, 59]. However, post traumatic neurological examination of animals injured in this way have shown substantial variation in their results, which appears to be caused by variation in uncontrolled parameters in the Allen model [16]. Certain biomechanical parameters have been established as having significant influence on acute and chronic experimental SCI. These include the contact surface area of the impactor, the contact surface between the cord and the impounding mass, the degree of cord compression, stroke, and duration of impact [60, 61]. Contact velocity and the degree of cord compression are also significant due to the viscoelastic mechanical properties of the cord [6, 16, 62]. Furthermore, this method is only suited to experimentally modelling one mode of injury, resulting in contusion lesions. In order to investigate different modes of injury more sophisticated experimental methodologies were required to be developed, with greater control over these mechanical parameters.

Using a vertebral dislocation injury mechanism, Fiford et al. [16] extended the Allen method to develop a new model of SCI in rats, allowing for greater control over the biomechanical parameters. This model utilised an experimental device that causes lateral displace-

ment of one vertebra in relation to the adjacent vertebrae. This includes the rupture of surrounding ligaments, which has been shown to occur in equivalent injuries in humans [63]. Immunohistochemistry was used to quantify the extent of the resultant axonal and vascular damage. Increased axonal injury was observed in regions of high axial strain and haemorrhaging was lateralized within the grey matter [16].

Choo et al. [6] further investigated SCI mechanisms in rats using a multi-mechanism device based on a linear actuator capable of simulating contusion, fracture dislocation (anterior), and flexion distraction injuries. Haemorrhage and cellular membrane compromise were used as the indicators of primary damage, given their significant influence in initiating the secondary injury events. Results indicated distinctly different injury patterns for each of the injury mechanisms, with distinct distributions of membrane compromise along the anteroposterior axis in both grey and white matter. Contusion injuries showed local increases in membrane compromise, whereas fracture dislocation and flexion distraction injuries showed an asymmetrical distribution with increased damage at the anterior end, consistent with the direction of the mechanical insult. The extent of haemorrhage was similar between contusion and fracture dislocation injuries, no haemorrhage was observed following distraction.

Clinical studies have shown that anterior fracture dislocation injuries often produces a greater neurological deficit than lateral fracture dislocation [64]. Clarke and Bilston [15] investigated fracture dislocation injuries in the rat thoracolumbar spine comparing both the anterior-posterior and lateral loading directions. Anterior thoracolumbar injuries were produced using the system developed by Choo et al. [6], and lateral injuries were produced using the system developed by Fiford et al. [16]. Significantly different spatial distributions of axonal damage, neuron damage, and haemorrhaging were

observed using relevant histological techniques. The neuropathology of anterior fracture dislocation injuries was observed to be more severe than lateral fracture dislocation injuries. This finding is in agreement with the trends observed in clinical studies, this highlights the significance of loading direction in the progression of SCI.

#### 1.3.2.3 *Ex vivo Experimental Models*

The Allen model has been widely extended and adapted to recreate burst fractures in *ex vivo* vertebral column specimens, rather than simply impacting the cord surface directly [10, 11, 65–70]. The occlusion of the spinal canal by impeding bone fragments was considered in these studies. Panjabi et al. [68] conducted experiments using 15 fresh human cadaveric thoracolumbar spine specimens (between T11-L1), of which burst fractures were successfully created in 9 specimens. Panjabi et al. noted that different impact energies produced different fractures, end plate fractures, wedge fractures, and burst fractures were observed, with burst fractures being the most common when impact energy was 84J or higher. The weight of the impactor was incrementally increased to create impacts of increasing energy until a fracture was achieved. It is possible that this approach affected the morphology and mechanical properties of the specimen prior to the induction of a fracture, potentially confounding the results. Wilcox et al. [10, 11], employed a similar approach using calf thoracolumbar spines, the energy of the impact was varied by adjusting the magnitude of the impactor mass, this approach reliably produced a burst fracture in every specimen. Impact energy was varied between 20J and 140J. Denis type C fractures (fracture of the inferior endplate) were observed for impacts up to 60J, above this Denis type A fracture (fracture of both endplates) were observed. An earlier study by Tran et al. reported the same extent of canal occlusion resulting from the experimental fractures observed by Wilcox et al., but at lower energy

impacts [69]. The difference in results may be due to differences in equipment and variations in the age of the spine specimens [14]. Tran et al. also used calf thoracolumbar spines, a drop weight impactor was used for high energy fractures, resulting in burst fractures, and a materials testing rig was used to induce low energy fractures, resulting in compressive fractures with lower degrees of canal occlusion [69].

There are issues associated with these experimental techniques, the incremental increasing of impact energy by Panjabi et al. is not truly representative of SCI. Chang et al. [66], Tran et al. [69], and Carter et al. [65] produced the burst fractures in a single test, occlusion was measured using an indirect method, by placing a plastic tube filled with water into the canal such that fluid pressure could be correlated to canal occlusion. Wilcox et al. used a high speed video camera to record the results, while this is a direct method of measuring canal occlusion it requires the removal of the cord from the spinal canal, consequently other spinal soft tissues were excluded from the experiments. Despite these difficulties these experimental studies support the link between neurological damage and the biomechanics of the primary injury. Impact energy, velocity, and weight, will influence the nature of the fracture during injury [71]. These combined studies show that transient occlusion is greater than final canal occlusion, the final resting place of the bone fragments in the spinal canal following a burst fracture does not indicate the full extent of the injury. This supports the hypothesis of Limb et al. "that neurological damage occurs at the moment of injury when the anatomy is most distorted, and is not due to impingement in the resting positions observed afterwards" [72]. This may account for the poor correlation between canal encroachment, assessed in patients through post-trauma radiographs or tomography scans, and the overall neurological deficit in those patients [68].

#### 1.3.2.4 *The importance of cerebrospinal fluid*

A number of recent studies have elucidated that the CSF and the dura mater act to protect the spinal cord, cushioning impacts [14]. Experiments by Persson et al. investigated the biomechanical influence of CSF on the spinal cord, analysing the response to impacts from a simulated bone fragment, mimicking a burst fracture [26]. This study used a computational FEM model, this was validated against an *in vitro* transverse impact test using bovine tissue, using a method developed by Hall et al. for experimentally reproducing burst fractures [9]. A burst fracture was simulated by propelling the bone fragment, with similar magnitude and velocity to that observed in the burst fracture process, through a tube to transversely impact the surface of the bovine spinal cord. Three pellet sizes of equal mass but varying impact surface area were used, specimens were tested in three states, with dura and CSF, with dura only, and without dura. A high speed video camera was used to record the impact, and image analysis was used to determine the deformation magnitude and duration. The results were used to validate the computational model. It was found that the presence of the CSF had a significant effect on the biomechanics of SCI and the patterns of deformation within the cord. The CSF and the dura were found to reduce the compression, stresses and strains following the impact. These results have compounded the findings of earlier studies by Persson and by Jones et al., showing the importance of the incorporating CSF into biomechanical models in order to gain an accurate understanding of the deformation of the spinal cord [73–75]. Jones et al. later developed a drop weight impactor experiment using a synthetic spinal cord specimen, an inverse relationship between the thickness of the CSF layer and cord deformation was observed, these results further support the conclusions of the earlier studies, that the CSF has a protective role in SCI [76].

To date, many computational models have not incorporated the CSF, or have included it in a simplified form, in doing so these models are limited in the extent to which they can simulate human SCI. This stems from the high computational expense of modelling the fluid and fluid structure interaction, this is discussed further in sections 1.4.2 and 1.5. In *ex vivo* tissue studies the incorporation of the CSF presents an additional challenge, as the fluid is generally lost when the specimen is removed from the body. This may be overcome through the use of a suitable CSF substitute for biomechanical experiments. A saline solution (0.9% NaCl) is an acceptable pseudo-CSF [14], as both are Newtonian fluids and have a similar viscosity. While the levels of blood cells and proteins present in CSF do affect the viscosity, it is not thought that high cell and protein concentrations affect it significantly [77]. The relative viscosity of CSF to saline is 1.02 at both room (20 °C) and body (37 °C) temperatures [78]. The maximum difference in density is 0.1% [14, 79, 80].

Animal tissue has frequently been used as a substitute for human tissue in creating *in vitro* and *in vivo* models of spinal cord biomechanics. Bovine and rodent samples are commonly used. Bovine tissue in particular is very close to human in terms of anatomy and its mechanical behaviour [81–83]. The CSF layer in humans is significantly thicker, in relative terms, than that found in bovine and rodent species [26]. Having established the importance of the CSF on the biomechanics of traumatic spinal cord injury, there is a need to study the effects of the varying the thickness of the CSF layer in order to translate the results of animal models to human physiology. Different animals, with differing CSF layers, may experience different neurological deficits in response to the same injuries [26].

## 1.4 COMPUTATIONAL MODELS OF SCI

### 1.4.1 *Modelling using finite element methods*

Finite element analysis (FEA) is a widely used technique for modelling a wide range of scientific and engineering problems. The basic principle is to divide the structure of interest into a finite number of smaller sub-structures (elements), joined together at nodes. Based on assumptions and real world observations, mathematical equations of motion are then defined in order to approximate the behaviour of each element [14, 84]. The nodes are essentially points within the defined coordinate system, and are generally positioned at the corners and edges of elements. FEA typically consists of creating a mathematical formulation of a physical process and then performing numerical analysis of the mathematical model. The creation of the initial mathematical model is reliant on background knowledge of the subject area. The numerical analysis requires knowledge and assumptions about how the process works, such as the mechanical properties of spinal cord materials [84].

There are a wide range of approaches and algorithms available for creating finite element models, as well as commercial software packages to aid researchers. The general approach may be divided into three stages: pre-processing, solution, and post processing. In the pre-processing phase the geometry of the specimen, for example a subsection of human spinal cord, is divided into discrete elements. This typically means creating a mesh of elements, although it should be noted that meshless FEM techniques may also be used [84, 85]. In this stage the properties of the elements are defined mathematically and conditions are set, such as boundary conditions. Next the FEA solver will derive the matrix equations, assemble the elements and solve the

system of equations. Finally, the post-processing phase involves extracting the results (e.g. stresses, displacement) and the derivation of additional quantities such as errors and specialised measures of stress (e.g. von Mises stress) [84].

### 1.4.2 *SCI Studies Using FEM*

Numerous studies have also applied FEM in calculating internal tissue stress and strain fields and investigating various aspects of SCI. Such models have been utilised to capture data that is not obtainable through *in vivo* and *in vitro* experiments. Studies investigating the properties of spinal cord tissue have used FE models to compliment their experiments, in order to analyse internal stress and strain patterns within the sample [51, 86–88]. Obtaining this sort of data is only really possible using a computational model, as it is not feasible to implant sensors within the tissue sample itself. FE models have subsequently become an key tool in the biomechanical investigations, a number of more complex FE models have been created to investigate various facets of SCI.

#### 1.4.2.1 *3D model of the human cervical spine*

Greaves et al. developed a three-dimensional finite element model of a section of human cervical spinal cord, including 60mm of the spinal cord and three spinal vertebrae (C4-C6) [18]. This model was used to investigate three clinically relevant modes of injury: transverse contusion (consistent with a burst fracture), distraction (as caused by distraction/distortion of the spinal column), and dislocation (as would result from a fracture dislocation injury). The geometry of the vertebrae and the cord was based on transverse 1mm cryosection images. The geometry of other components, including ligaments, dura, and

attachments, was based on geometries reported in the literature. Ligaments were modelled as two-node link elements with linear shape functions, reactive to tensile load only. The tissues were assigned linear elastic properties based on values from published experiments. Spinal cord tissue has been demonstrated to exhibit linear elastic behaviour for quasi-static axial strains of up to 5% [89–91], as the axial strains of the model reached maximums of between 8–13% this was deemed an acceptable simplification within the scope of the study. Grey and white matter were modelled as a contiguous single material, on the basis that the mechanical properties of grey and white matter have yet to be fully characterised. Contact elements were created for groups of contacting elements; the dura, CSF, and spinal cord. For each simulation normal stiffness was approximated for the contact elements to account for the effect of CSF and fat in connecting spaces. The CSF was included indirectly, as a pure slip interface between the spinal cord and the dura. This simplification omits the protective effect of the CSF and the full mechanics of the fluid structure interaction. Nerve roots and rootlets were excluded from the model, as previous studies have shown that they transmit load to the cord through dura mater and denticulate ligaments rather than via the rootlets themselves [92]. To validate the model simulations were set up to match published *in vivo* experiments [6], allowing for comparison between the numerical and experimental results. *In vivo* experiments were used wherever possible, in some cases parameters were scaled up from small animal *in vivo* experiments to closer match that of humans. The Young's Modulus was used to measure the mean average response of the nodes, the authors stated that future work would include more complex material models. Von Mises strain was used as an indicator of spinal cord damage. Overall the results of the simulations demonstrated distinctly different strain fields for each mode of injury.

#### 1.4.2.2 *Models with distinct materials for grey and white matter*

Maikos et al. [93] created a three-dimensional FE model of a rat spine, including the cord, CSF, dura, and spinal vertebrae, this model was used to simulate impactor drop weight tests, based on the Multicentre Animal Spinal Cord Injury Study (MASCIS) impactor model of spinal cord contusion developed by Young [94]. Spinal cord geometry was obtained based on 4-Tesla MRI images of a rat, this rat was sacrificed and the spinal column excised for further imaging. The spinal column was imaged using 2-Tesla weighted spin echo MRI, these images were used to manually extract the boundaries between the grey and white matter. In creating the FE model the cord tissue was partitioned into elements with a more uniform geometry, independent element sets were defined for grey and white matter. Tethering structures, including ligaments, blood vessels, and nerve roots, were omitted from the model. The CSF and dura were included by expanding the outer boundary of the spinal cord by 3% and 5% respectively, based on the mean average diameter of the MRI images. Both the dura and the CSF were $50 - 80$ $\mu$m in length, and one element thick in FE model. The geometry of the vertebral column was gathered from micro-CT imaging of a rat spinal cord, a T9/T10 laminectomy was performed to match the drop weight experiments. The material properties of the cord tissue were based on results published by Fiford [17], which characterised the behaviour of rat spinal cord tissue in vivo, but only up to strains of 5%. Although the strains in this experiment exceed 5%, no other data was available at the time of the study. The data from the Fiford study was fitted with an Ogden hyperelastic strain energy density function, this function has been used previously to model both spinal cord and brain tissue, a Prony series exponential decay was used for the viscoelastic portion. Maikos et al. [95] characterised the hyperelastic-viscoelastic properties of rat dura mater in response

to uni-axial tension at low and high strain rates. Dura was modelled as a one-term Ogden hyperelastic function with the four-term Prony series. CSF was modelled with fluid like behaviour, using solid elements with a low shear-to-bulk modulus and Mooney-Rivlin hyperelastic material properties. Pia mater was excluded in this model.

The material properties were tweaked, following calibration and sensitivity analysis, to more closely resemble the condition of the drop weight experiment [93]. The model was validated and refined against a parallel impactor drop weight experimental study. Analysis of displacement measurements provided spatial and temporal displacement profiles of the mechanical response to the mechanical insult for comparison with the FE model. Overall this model compared well with the *in vivo* experimental results in terms of the injury patterns produced. A parametric sensitivity analysis was performed using the homogeneous model with a 12.5 mm drop height. It was found that altering the instantaneous shear modulus produced the largest changes in impactor trajectory and injury patterns. Changes to the quasi-static shear modulus and Prony series viscoelastic constants produced proportionally smaller changes. The model was not sensitive to changes in the $\alpha$ Ogden material parameter or to the coefficient of friction and mechanical properties of the dura and CSF. The authors suggest that with some refinement, development, and further validation it may be used to establish tissue level thresholds for SCI in the future. Parametric testing found that shear properties have a strong influence on tissue displacement, in addition to the magnitude and distribution of stress and strain. The model was restricted to a short segment of the overall spine, this was done to reduce the computational complexity, however, it was found that altering the length of the spinal segment did not significantly impact the results for this type of injury simulation [93]. The material properties of the white and grey mater are significant, the authors noted that to

achieve greater fidelity to experimental results it would be necessary to further characterise the mechanical properties of both grey and white matter, beyond the findings of Fiford [17].

Li and Dai [53] used FEM to investigate hyperextension injuries to the spinal cord in three dimensions. The geometry of their FEM model was adapted from a physical specimen. During the autopsy of a healthy human cervical spinal cord cross sectional images were captured. Image analysis was used to extract the outlines of the white and grey matter, key points on the external and internal contours of the cervical cord were also obtained. This geometry was used to build the finite element model, by feeding the data into the pre-processor of a commercial FEM software package. A number of assumptions were made to simplify the model construction, for example the spinal segment length was set to 1.5 cm (a value based on statistic data of post-mortem measurements). This length was used for all segments, assuming they are all equal in length, in reality there is some variation. Whereas Greaves et al. [18] assumed a homogeneous material, this model incorporated two materials with differing properties, the white and grey matter of the spinal cord. The mechanical properties of these materials were defined based on a review of the available literature, the elastic moduli and Poisson's ratios for white and grey matter elements were set accordingly. All materials were modelled as a linear elastic continuum, exhibiting isotropic properties. This is a simplification, as in reality the spinal cord is anisotropic and exhibits viscoelastic behaviour under deformation [96]. The model, both white and grey matter, was meshed using 10-node tetrahedral elements (quadratic tetrahedrons). A convergence test is used to verify the numerical accuracy of a finite element solution, it can give an indication of how many elements are needed in order to be trust the solution. In this example, the test was implemented by applying longitudinal load from the top of the spinal segment, with the base

constrained. The stiffness calculated by dividing the top loading force by the longitudinal displacement of the cord at the loading point. This was repeated using varying numbers of elements. The model was considered to be converged when the cervical cord stiffness of the denser model differed by less than 3%, compared to the previous model [53]. The model created by Li and Dai was used to simulate a hyperextension injury. The extension forces involved in this injury type can be divided into axial and anteroposterior distraction forces, applied to the nodes on the surface of the cord. The simulations were repeated using varying forces, comparable to those used in other published experiments. Following the simulated injury, the authors analysed the strain distributions and extracted the von Mises stress for nine relevant regions of the spinal cross sections. The model was validated against published experimental studies, the force displacement response was compared to a study by Hung et al. [97], results for axial tension injury were compared to a study by Maiman et al. [98]. These experimental studies characterised the mechnical response of cord tissue. In both cases the FE results showed the same trends in comparison to the experimental results, indicating that this FE model accurately represents the biomechanics of CNS tissue [53]. Hyperextension injury was found to apply high stress at the anterior and posterior horn in the grey matter. It was suggested that this may account for loss of hand movement ability in patients with central spinal cord injuries [53].

#### 1.4.2.3 *Model to investigate the biomechanical significance of CSF*

Persson et al. [26] used a FE model to investigate the significance of the CSF in spinal trauma models, looking specifically at contusion injuries, this model was validated against a parallel *in vitro* study using bovine tissue. Previous models have either excluded the CSF entirely, or included in a highly simplified form, this was the first model to

include the fluid structure interactions due to presence of the CSF. The FEM software used for this study (Version 8.5; ADINA R&D Inc., Watertown, MA) allowed for simultaneous fluid and solid solution, and implicit solution of the dynamic FSI [99]. Results were presented in terms of axial strain and Von Mises stress, these have been shown previously to correlate to to neurological damage [17, 88]. The experiments involved propelling a simulated bone fragment against the posterior surface of the spinal cord to simulate a contusion injury representative of a burst fracture. The spinal cord geometry for the FE model was based on measurements obtained from previous experiments [73, 74, 100]. Three versions of the model were created: the spinal cord only, the spinal cord and dura mater, and the CSF, dura, and cord together. The dura was positioned $1mm$ from the spinal cord and was assumed to be frictionless. Denticulate ligaments and nerve roots were excluded. The cord was not divided into pia mater, grey and white matter, rather it was assumed to be homogeneous for the purposes of the model. This assumption has been made in previous studies and was deemed acceptable for the purposes of a base model [17, 18, 100], use of a homogeneous model reduces computational expense. The cord was later divided into these sub-materials in order to perform a sensitivity analysis for these parameters. The cord material properties were based on a previous study into the mechanical responses of cat spinal cord tissue to static loading by Hung et al., this experiment was performed using a low strain rate ($0.002mm/sec$) [97]. Although this strain rate is much lower than would occur in typical SCI, this is the only data available on spinal cord under compression available at the current time. Previous studies have shown that the effect of strain rate on the mechanical properties is minimal, the use of this data was therefore deemed acceptable [96, 101]. Studies have also shown that there are no significant differences in the mechanical properties between species at similar strain rates [81, 83, 89].

The data was fitted to a hyperelastic Ogden model, (Table 2). A parallel study by Persson et al. [96] characterised the properties of the spinal dura mater at high strains. A constant single modulus for the dura in this model, based on the tangent modulus, was derived from this data, using only the straight section of the stress-strain curve to represent the mechanical properties. This is a simplification that has been used in a number of earlier studies and helps to reduce computational complexity. The computational model was validated against the experimental study, and overall the computation results compared well [26].

Previous experiments had assumed that deformation of the cord did not occur prior to subdural collapse, until all of the CSF had been pushed out of the way by the impacting fragment [73, 74], however, the results of this study demonstrated significantly different patterns of displacement with the inclusion of the FSI. In the computational model instantaneous cord deformation was observed, as pressure was transferred, through the incompressible fluid, from the dura to the cord. Furthermore it was found that differently sized impacting fragments could produce different displacement patterns in addition to different magnitude of deformation. Earlier studies by Jones et al. and by Persson indicated that the CSF had a protective effect [73, 74]. This FSI model confirms this, demonstrating that the CSF has a significant protective effect in contusion injuries, reducing compression, and promoting the longitudinal distribution of stress and strain along the spinal cord. This longitudinal distribution falls in line with clinical observations [102, 103] and animal experiments [6, 104]. Persson [73] observed a direct protective effect of the dura mater, in terms of reduced maximum compression, although this was not always statistically significant, the effect of the CSF being greater. The sensitivity analysis portion of this study indicated that the pia mater had a significant effect of spinal cord deformation in response to contusion in-

juries, it was found to create concentration stresses within the centre of the cord when included in the model. The division of the cord tissue into grey and white matter resulted in small variations in stresses or strains, likely due to the very similar stiffness of the two materials. The authors note that the division of the cord into grey and white matter may be important for evaluating strain fields within the cord. However, for this comparative study, concerned with cord deformation and bone fragment trajectory, these variations were not found to be significant. An implication of this study is that neurological damage may differ between species due to variations in morphology in relation to the CSF. Rodent *in vivo* models are often used to investigate SCI [6, 15–17, 93, 104–108], however, the CSF layer in rodents is significantly thinner than in humans or bovine subjects. In humans lying prone has been shown to reduce the thickness of the posterior CSF layer and increase the thickness of the anterior CSF layer [82]. Testing on animals in the prone position may also decrease the thickness of the posterior CSF layer, potentially an additional source of error in experimental SCI.

#### 1.4.2.4 *Model of a rat cervical spine with CSF modelled using SPH*

Russell et al. [54] created a three dimensional FE model of the rat cervical spine, which was used to investigate contusion and dislocation injuries. In contrast to many other models that have omitted the CSF, or included it only as an abstract and simplified motion, Russell et al. modelled the CSF behaviour using Smooth Particle Hydrodynamics (SPH) [109], which they propose as an efficient means of incorporating the fluid structure interactions. This method is discussed further in section 1.5.4. SPH functionality is available in PAM-CRASH (ESI Group, Paris, France), the software package used for this study. This method was used to incorporate the interactions between the cord, CSF, and dura in the impact simulations.

The model included four spinal vertebrae (C3-C6), selected to correspond with earlier experiments by Choo et al. [6]. Restricting the model in this way was done to reduce computational complexity, earlier experiments by Maikos et al. [93] have shown that changing the model length had only a minimal impact on the results, due to the localised nature of drop weight method for simulating contusion injuries. In the case of dislocation injuries, Choo et al. [6] demonstrated that the axial injury extends up to 3mm away from the epicentre of the injury, this falls within the length of the C3-C6 model. High resolution 7-T magnetic resonance imaging was used to scan a rat cervical spinal cord, in-plane and through-plane scans were combined to create a fused pixel image from which the geometry was extracted for the FE model. This process was semi-automated through the use of image analysis software. The dura mater could not be reliably extracted from the MRI image, therefore the dura was created by expanding the surface of the cord based on the outline of the CSF based on MRI images provided by Choo et al. [6]. The dura was assigned a thickness of 90$\mu$m. Ligaments were modelled using manually defined two-dimensional elements, similar to Greaves et al. [18]. The computational experiments were validated against previous rat experiments. Cellular permeability to fluorescein-dextran was used as an indicator of neural tissue damage, as a link has been established to neuronal pathology [110–113]. The experimental data was correlated to the maximum principal strain in the computational model. The spinal cord was assigned hyper-viscoelastic Ogden and Prony material properties based on experiments by Maikos et al. [93], and hyperelastic properties based on experiments by Fiford [17] combined with viscoelastic brain tissue properties [114], calibrated by Maikos et al. [93] to match their drop weight experiments (Table 2). Dura mater properties were assigned based on values reported by Maikos et al. [95]. Spinal ligaments were modelled in a simplified form, using non-

linear tension bar with linear elastic properties. Properties for the cross sectional areas were scaled down from the human model created by Greaves et al. [18] to fit the rat model. Maximum strains were based on values reported in the literature [115–117]. For the intervertebral discs the annulus fibrosus were included, the nucleus pulposus was omitted to reduce computational complexity. Linear elastic properties of the C4/C5 disc were based on the properties used previously by Greaves et al. Endplate connection was simulated using spot welds and calibrated to match the behaviour observed in previous rat experiments by Choo et al.

### 1.4.3 *Material properties*

The material properties used in published finite element SCI studies are listed in Table 2. From 2005 onwards many of these studies have based the material properties of the spinal cord tissue on results published by Fiford [17], adapting and scaling them as necessary. This study investigated the response of freshly excised rat spinal cord specimens to uniaxial tension, finding a non-linear viscoelastic response. Fiford tested the tissue with strain rates between 0.0002s and 0.2s, strains ranging from 2% to 5%. These strains and strain rates are below that experienced by the cord tissue in the computational models and in traumatic SCI, while the authors acknowledge this, there is currently no more data available characterising the mechanical properties of spinal cord tissue, suggesting that further experimentation is required. Models of the human spinal cord generally rely on this data obtained for the rat cord, although the tissue properties are assumed to be similar, rat tissue cannot be fully representative of human tissue. Furthermore, there is lack of data regarding the individual mechanical properties of white and grey matter, while it is generally accepted

that the difference is small, as models become more detailed this difference may become more significant. The difficulty in characterising the mechanical properties of the tissue is compounded by confounding factors, notably age, and the time between subject death and testing.

| Study | Component | Material Models | Parameters | $\nu$ | Software |
|---|---|---|---|---|---|
| Bilston [51] | Spinal Cord | Linear elastic<br>Hyperelastic (Ogden)<br>Viscoelastic | $E = 1.2MPa$<br>$G = 100kPa$,<br>$\alpha = 25$<br>$G_\infty = 100kPa$,<br>$\alpha = 25$,<br>$G_1 = 154kPa$,<br>$G_2 = 179kPa$,<br>$G_3 = 559kPa$,<br>$\tau_1 = 4.38s$,<br>$\tau_2 = 0.554s$,<br>$\tau_3 = 234.4s$ | 0.4 | N/A |
| Ichihara et al. [87] | Spinal cord<br>White matter<br>Grey matter | Viscoelastic model (Standard linear solid model with nonlinear component) | $E_e = 0.28MPa$,<br>$E_c = 0.15MPa$,<br>$C_1 = 6.1 \times 10^5$,<br>$C_2 = 1.9 \times 10^4$.<br>$E_e = 0.66MPa$,<br>$E_c = 0.36MPa$,<br>$C_1 = 2.2 \times 10^6$,<br>$C_2 = -1.6 \times 10^4$. | 0.4<br>0.4 | N/A |
| Oakland [100] | Spinal cord<br>Dura Mater | Linear elastic<br>Linear elastic | $E = 1.25MPa$<br>$E = 1.44MPa$ | 0.49<br>0.49 | LS-Dyna; Livermore Software Technology, CA, USA |
| Wilcox et al. [118] | Spinal cord<br>Dura Mater | Linear elastic<br>Anisotropic elastic | $E = 1.3MPa$<br>$E_{rr} = 142MPa$,<br>$E_{\theta\theta} = 142MPa$,<br>$E_{zz} = 0.7MPa$ | 0.35<br>N/A | LS-Dyna; Livermore Software Technology, CA, USA |
| Fiford and Bilston [119] | Spinal cord | Hyperelastic (Ogden) | $G = 176kPa$<br>$\alpha = 47$ | N/A | N/A |
| Galle et al. [86] | Spinal cord white matter | Hyperelastic (Mooney-Rivlin) | $C_{10} = 592Pa$<br>$C_{01} = 249Pa$ | N/A | COMSOL 3.2 with Matlab, COMSOL, Inc., Burlington, MA |
| Ouyang et al. [88] | Spinal cord white matter | Hyperelastic (Mooney-Rivlin) | $C_{10} = 592Pa$<br>$C_{01} = 249Pa$ | N/A | COMSOL 3.2 with Matlab, COMSOL, Inc., Burlington, MA |

| | | | | | |
|---|---|---|---|---|---|
| Greaves et al. [18] | Spinal cord<br>Dura mater | Linear elastic<br>Linear elastic | $E = 0.26MPa$<br>$E = 5MPa$ | 0.49<br>0.45 | Ansys 7.1, Ansys Inc., PA, USA |
| Maikos et al. [93] | Spinal cord | Hyperelastic (Ogden) combined with viscoelastic (two-term Prony series decay) | $G_\infty = 32kPa$, $\alpha = 4.7$ $G_1 = 99.4kPa$, $G_2 = 56.8kPa$, $\tau_1 = 8ms$, $\tau_2 = 150ms$ | 0.45 | Ansys software, Ansys Inc., PA, USA |
| | Dura mater | As above but four-term Prony series decay | $G_\infty = 1205kPa$, $\alpha = 16.2$ $G_1 = 1069kPa$, $G_2 = 416kPa$, $G_3 = 335kPa$, $G_4 = 335kP$, $\tau_1 = 9ms$, $\tau_2 = 81ms$, $\tau_3 = 0.564s$, $\tau_4 = 4.69s$ | 0.45 | |
| | CSF | Hyperelastic (Mooney-Rivlin) | $G = 134Pa$ $C_{01} = 33.5Pa$ $C_{10} = 33.5Pa$ | N/A | |
| Sparrey et al. [120] | Spinal cord white matter | Hyperelastic, Linear elastic* | $E = 0.065$, 0.09, 0.115, 0.14 and $0.165MPa$ | N/A | Not specified |
| | Spinal cord grey matter | Hyperelastic, Linear elastic* | (as for white matter) | N/A | |
| | Pia mater | Linear elastic | $E = 0.06$, 1.2, 1.8, 2.4 and $3.0MPa$ | N/A | |
| Li and Dai [53] | Spinal cord white matter | Linear elastic | $E = 0.277MPa$ | 0.4 | Ansys software, Ansys Inc., PA, USA |
| | Spinal cord grey matter | Linear elastic | $E = 0.656MPa$ | 0.4 | |
| Persson 2009 [73] and 2011 [26] | Spinal cord | Hyperelastic (Ogden) | $G = 9kPa$, $\alpha = 9$ | 0.4 | ADINA Version 8.5; ADINA R&D Inc., Watertown, MA |
| | Dura mater | Linear elastic | $E = 80MPa$ | 0.4 | |
| | CSF | Newtonian | $\eta = 0.001Pas$ | 0.49 | |

| | | | | | |
|---|---|---|---|---|---|
| Russell et al. [54] | Spinal cord | Hyperelastic (Ogden) combined with viscoelastic (two-term Prony series decay)** | $\mu = 40.04\text{kPa}$, $\alpha = 4.7$ $g_1 = 0.5282$, $g_2 = 0.3018$, $\tau_1 = 8\text{ms}$, $\tau_2 = 150\text{ms}$ | 0.45 | PAM-CRASH, ESI Group, Paris, France |
| | Dura mater | As above but four-term Prony series decay** | $\mu = 40.04\text{kPa}$, $\alpha = 4.7$ $g_1 = 0.3182$, $g_2 = 0.1238$, $g_3 = 0.0997$, $g_4 = 0.0997$, $\tau_1 = 9\text{ms}$, $\tau_2 = 81\text{ms}$, $\tau_3 = 0.564\text{s}$, $\tau_4 = 4.69\text{s}$ | 0.45 | |
| | CSF | Murnaghan Equation of State model | $\beta = 200\text{MPa}$, $P_0 = 0.001\text{g/mm}^3$ | N/A | |

Table 2: Material models used for computational studies of spinal cord injury (adapted from Persson et al. [14]). E is the elastic modulus. $C_1$ and $C_2$ are viscous coefficients. $C_{01}$ and $C_{10}$ are Mooney-Rivlin constants. For the hyperelastic Ogden models, G and $\alpha$ are the Ogden material constants.
*Linear elastic models with different tangent moduli used. Hyperelastic models with parameters based on the fit of a third-order hyperelastic Ogden model to the linear elastic tangent modulus.
**Adapted from Maikos et al. [93], converted due do differences in notation between ABACUS (AB) and PAM-CRASH (PC) software: $\mu^{(PC)} = G_0^{(AB)}/\alpha^{(AB)}$, $g_i^{(PC)} = G_i^{(AB)}/G_0^{(AB)}$.

### 1.4.4 *Assumptions and simplifications*

The models used in these studies share a number of common limitations, the foremost being the computational expense and subsequently long processing times incurred by the models, which restricts the level of detail. Researchers have had to think carefully about which parts of the model require high numbers of elements, and which parts can be simplified while minimising the loss of biofidelity.

In all cases only a subsection of the spinal cord was modelled, offering substantial reductions in the elements required for the model and the computational complexity. These short sections were sufficient for these experiments and the parameters used. In the future longer sections may be required, to investigate the effects of injury further away for the point of impact, and to investigate the effects of multiple injuries to the spinal cord at different locations.

The study by Persson et al. highlighted the significance of the CSF, forces may be transferred from the dura, through the CSF, to the cord tissue. While previous experimental studies have taken the assumption that cord deformation does not occur prior to subdural collapse, there is evidence to suggest this is not the case [26]. Forces may be transferred from the dura, through the CSF, to the cord tissue. Unfortunately it is not possible to verify this experimentally, future investigations will likely need to be computational, the inability to validate the computational results against experimental data remains an issue.

## 1.5 ALTERNATIVES TO FEM

Finite element models have been established as a useful tool for modelling SCI, but the required complexity is currently restricting due to a number of limiting factors. In order to further investigate the relationship between cord deformation and the resultant biological damage in the future, it will be necessary to create more complex models. These will be more closely representative of SCI in humans, eventually incorporating the complex structure of spinal sub-components, including cells, fibres, and fluid [14].

Key limitations with FEM are difficulty in implementing the fluid-structure interaction between the cord and the surrounding CSF, the inability of FEM to cope with large deformations, and poor amenability to implementation on parallelised computing systems. As a result, simplifications must be made to the models. For example, studies have simplified the properties of spinal cord tissue into a homogeneous material, and assumed linear elastic behaviour [18]. Other studies divide the cord material into grey and white matter, assigning appropriate mechanical properties to each [53].

The effect of the CSF on the biomechanics of SCI is significant, as discussed in section 1.3.2.4, this warrants the inclusion of CSF into SCI models [26]. To incorporate the CSF it is necessary to capture the fluid structure interactions. Incorporating the fluid dynamics of the fluid and its interaction with the solid phases adds greatly to the computational cost. The computational expense of implementing the FSI is generally the first limiting factor, restricting the complexity of the FEM model. A simple approach is to eliminate one field, fluid or solid, and abstract away the eliminated field to a specific motion or force at the interface between solid and fluid [84]. A partitioned approach may be used, whereby the solid and fluid mechanics are solved separately, with algebraic constraints on the boundary between the two. Alternatively, a monolithic approach can be taken, treating the solid, fluid and the interactive boundary as one system of equations [14]. In either case simplifications must be made within the confines of computational processing power.

SCI simulations often involve large material deformations, over very short time spans, requiring a high number of time steps. Frequent computationally complex remeshing operations are consequently

required, this again increases computation time required and can introduce additional error into the results if suboptimal (but more stable) element types are used. This issue is even more prevalent when the CSF is incorporated into the model, and FEM methods struggle to cope with large deformations and dislocations [14, 85].

Modern high powered computer systems achieve improved performance using multiple processors running in parallel, rather than increasing the speed of a single processor, which is restricted by technical hardware limitations. Traditional FEM is difficult to implement for parallel computing, generally these simulations can only run on a single processing core, subsequently, FEM techniques remain limited in their complexity. This justifies the investigation of alternative algorithms for modelling SCI computationally, which are amenable to parallelisation and better placed for creating more complex models. The basic principle of FEM is to divide the structure of interest into a finite number of smaller sub-structures (elements), joined together at nodes. This creates a deformable mesh of linked elements, traditional FEM is considered a mesh based technique. Alternative mesh-free techniques use a different approach; although a mesh is defined, it does not move, it acts as a frame of reference against which the movement of material points can be described [121]. Mesh-free techniques have the key advantage of being easily implemented for highly parallelised computing, and are inherently suited to handling large deformations [122]. The Material Point Method (MPM) and Lattice Boltzmann Method (LBM) are mesh-free algorithms that have been identified as having the potential for modelling SCI in the future. Although not yet applied to spinal biomechanics, these techniques have been successfully applied in other engineering applications[123, 124].

MPM has already been used to model multicellular tissue constructs [125], as well as in larger scale models of soft tissue failure [122, 126]. It should, therefore, be possible to apply this technique to neural tissue.

### 1.5.1 *The Material Point Method*

MPM is a numerical technique based on particle in cell (PIC) methods. PIC methods were originally developed for modelling fluid dynamics and dealing with highly distorted flows, they evolved from the work of Harlow in 1963 [127]. PIC methods were later adapted into the Fluid-Implicit-Particle (FLIP) method for application to computational solid mechanics problems, from which the MPM method was derived by Sulsky [128, 129]. Despite its roots in fluid dynamics, MPM has also been successfully applied to solid mechanics problems [130].

In MPM a grid (or mesh) overlay is used to discretize complex material shapes into a finite set of material points that hold all attributes and constitutive state response variables. This is typically a regular grid, defined on the basis of computational convenience. Points are generated at positions lying within the given body, the points can interpolated using the overlying mesh. MPM may be classed as a meshless technique. Although a mesh is defined, it does not move, it acts as a frame of reference against which the movement of material points can be described [130].

MPM is an Arbitrary Lagrangian-Eulerian (ALE) method. Most algorithms in continuum mechanics make use of one of two classical descriptions of motion: Lagrangian and Eulerian. In Lagrangian al-

gorithms each node of the computational mesh follows the associated material particles. This description is well suited to tracking free surfaces and interfaces between differing materials. However, it is less suitable for tracking large distortions, which requires frequent remeshing operations. Most mesh based FEM methods, including those discussed in section 1.4.2, use the Lagrangian description. In Eulerian algorithms the mesh is fixed, and the continuum moves in relation to the grid. In this description it is easy to track large distortions, however, this comes at the expense of precise interface definition and high resolution flow details. This description is widely used in fluid dynamics. Table 3 summarises the relative strength and weaknesses of both descriptions. Using ALE algorithms, nodes of the mesh may be moved with the continuum in the Lagrangian fashion, or be held fixed in an Eulerian manner, or be moved in some arbitrarily defined way. The ALE description combines the two descriptions, aiming to incorporate the strengths of both, without the drawbacks [131–133]. Table 4 compares the relative strengths and weaknesses of MPM (an ALE method) vs FE (a Lagrangian method).

#### 1.5.1.1 *The MPM Algorithm*

Figures 5 outlines the steps in a single MPM cycle. Lagrangian point masses (the material points) move in relation to an Eulerian background mesh. At each convective step in the cycle the material points are projected onto the grid nodes, the following equations are solved using the grid. The material points are then updated and the grid is reset. The following is an explanation of the steps in each cycle of the MPM algorithm, adapted from Chen and Brannon [130]. This description assumed that the continuum body has already been dis-

| | Lagrangian | Eulerian |
|---|---|---|
| Grid | Attached on the moving material | Fixed in space |
| Tracking | Movement of any point on materials | Mass, momentum, and energy flux across grid nodes and mesh cell boundary |
| Time history | Easy to obtain time-history data at a point attached on materials | Difficult to obtain time-history data at a point attached on materials |
| Moving boundary and interface | Easy to track | Difficult to track |
| Irregular Geometry | Easy to model | Difficult to model with high accuracy |
| Large deformation | Difficult to handle | Easy to handle |

Table 3: Comparison of Eulerian and Lagrangian methods, adapted from Liu and Liu [133].

| MPM | FE |
|---|---|
| Easy to generate initial point cloud | Defining initial geometry requires work and skill |
| First order accuracy | Second order accuracy |
| More computationally expensive per timestep | Less computationally expensive per timestep |
| Scales well for parallel computation | Scales poorly for parallel computation |
| No mesh tangling/element inversion | Prone to mesh tangling/element inversion |
| Good for high rate of strain problems | Good for low rate of strain problems |
| Averaged FSI when using MPMICE | Monolithic FSI with partitioned phases |
| Arbitrary Lagrangian-Eulerian mesh | Lagrangian mesh |

Table 4: Relative strength and weaknesses of MPM versus FE

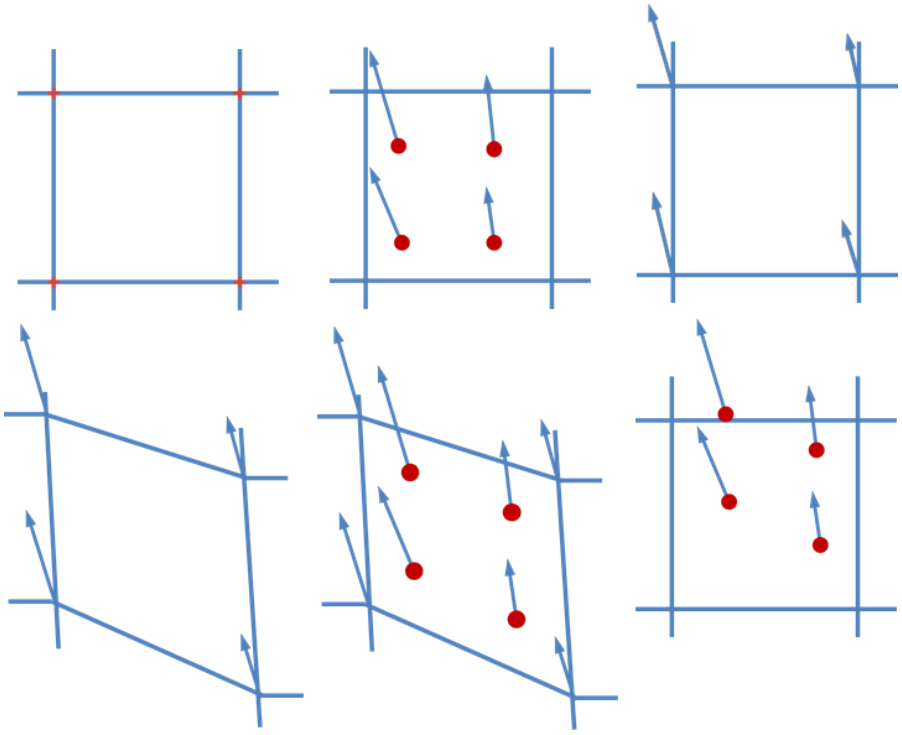


Figure 5: Illustration of the steps in the MPM algorithm from top–left to bottom–right: (1) A regular grid is defined (2) Four material points occupying (filled circles) overlaid on single cell of the background grid (solid lines). Arrows represent displacement vectors. (3) The material point state vectors (mass, volume, velocity, etc.) are projected onto the nodes of the computational grid. (4) The discrete form of the equations of motion is solved on the computational grid, the results are used to update the velocities and positions grid nodes. (5) The updated nodal kinematics are interpolated back to the material points, and their state is updated. (6) The computational grid is reset to its original configuration, and the process repeats.

cretized into a set of material points and that all state variables have been initialised.

The masses of the particles in each cell are mapped to the nodes of the cell containing those particles:

$$m_i^t = \sum_{p=1}^{N_p} M_p N_i(\boldsymbol{x}_p^t) \tag{1}$$

where $m_i^t$ is the mass at node $i$ time $t$, $M_p$ is the mass of the particle, and $N_i$ is the shape function associated with node $i$, $\boldsymbol{x}_p^t$ is a vector giving the location of the particle (in a Cartesian coordinate system) at time $t$.

The momentum of the particles is mapped to the appropriate nodes in the same way:

$$(m\boldsymbol{v})_i^t = \sum_{p=1}^{N_p} (M\boldsymbol{v})_p^t N_i(\boldsymbol{x}_p^t) \tag{2}$$

where $(m\boldsymbol{v})_i^t$ is the nodal momentum of node $i$ at time $t$, and $(M\boldsymbol{v})_p^t$ is the particle momentum at time $t$.

The internal force vectors for the grid nodes can be found using:

$$(\mathbf{F}_i^t)^{int} = \sum_{p=1}^{N_p} \mathbf{G}_i(\boldsymbol{x}_p^t) \cdot \mathbf{s}_p^t \frac{M_p}{\rho_p^t} \tag{3}$$

where $\mathbf{G}_i(\boldsymbol{x}_p^t)$ is the gradient of the shape function for node $i$ evaluated at $\boldsymbol{x}_p^t$, $\mathbf{s}_p^t$ is the particle stress tensor at time $t$, and $\rho_p^t$ is the particle mass density at time $t$.

Essential and natural boundary conditions are applied to the grid nodes and the nodal force vectors computed as follows:

$$(\mathbf{F}_i^t) = (\mathbf{F}_i^t)^{int} + (\mathbf{F}_i^t)^{ext} \tag{4}$$

where $(\mathbf{F}_i^t)^{ext}$ represents the external nodal force vector. Essential (or geometric) boundary conditions are imposed on the primary variable, e.g. displacements, whereas natural (or force) boundary conditions are imposed on the secondary variable, e.g. forces and tractions. Essential boundary conditions remove degrees of freedom from the domain boundary, while natural boundary conditions do not.

The momenta of the grid nodes is updated:

$$(m\boldsymbol{v})_i^{t+\Delta t} = (m\boldsymbol{v})_i^t + \mathbf{F}_i^t \Delta t \tag{5}$$

The mapping operation, from the nodes of the cell containing a particle to the particle itself, is now performed for each particle.

Nodal accelerations are mapped back the the particles:

$$\boldsymbol{a}_p^t = \sum_{i=1}^{N_n} \frac{f_i^t}{m_i^t} N_i(\boldsymbol{x}_p^t) \tag{6}$$

Current nodal velocities are mapped back to the particles:

$$\boldsymbol{v}_p^{-t+\Delta t} = \sum_{i=1}^{N_n} \frac{(m\boldsymbol{v})_i^{t+\Delta t}}{m_i^t} N_i(\boldsymbol{x}_p^t) \tag{7}$$

Compute the current particle velocity for strain calculations:

$$\boldsymbol{v}_p^{t+\Delta t} = \boldsymbol{v}_p^t + \boldsymbol{a}_p^t \Delta t \tag{8}$$

Compute the current position for the particle (this equation represents a reverse integration):

$$\boldsymbol{x}_p^{t+\Delta t} = \boldsymbol{x}_p^t + \boldsymbol{v}_p^{-t+\Delta t} \Delta t \tag{9}$$

Compute the displacement vector:

$$\boldsymbol{u}_p^{t+\Delta t} = \boldsymbol{x}_p^{t+\Delta t} - \boldsymbol{x}_p^0 \tag{10}$$

Nodal shape functions are used to continuously map the nodal velocity interior to the interior of the cell, equations (7) and (9), this allows the positions of the particles to be updated by moving them in a single-valued continuous velocity field. Using $\boldsymbol{v}_p^{-t+\Delta t}$ instead of $\boldsymbol{v}_p^{t+\Delta t}$ to update the particle position reduces numerical error within the system, and interpenetration between material bodies is prevented. This is a feature of MPM that allows it to handle penetration and impact scenarios without the need for specialist contact algorithms [130].

The updated particle momenta is now mapped back to the nodes of the cell containing those particles:

$$(m\boldsymbol{v})_i^{t+\Delta t} = \sum_{p=1}^{Np} (M\boldsymbol{v})_p^{t+\Delta t} N_i(\boldsymbol{x}_p^t) \tag{11}$$

Find the updated nodal velocities:

$$\boldsymbol{v}_i^{t+\Delta t} = \frac{(m\boldsymbol{v})_i^{t+\Delta t}}{m_i^t} \tag{12}$$

Essential boundary conditions are now applied to the grid nodes of cells containing boundary particles, consistent with the weak form of the governing equations; $\boldsymbol{w}_i^t$ are assumed to be zero on the essential boundary.

If needed for a constitutive model, find the current gradient of particle velocity:

$$\mathbf{L}_p^{t+\Delta t} = \sum_{i=1}^{N_n} \boldsymbol{v}_i^{t+\Delta t} \mathbf{G}_i(\boldsymbol{x}_p^t) \tag{13}$$

the particle strain increment:

$$\Delta \boldsymbol{e}_p = (sym\mathbf{L}_p^{t+\Delta t})\Delta t \tag{14}$$

where, $sym\mathbf{L}_p^{t+\Delta t}$ denotes the symmetric product applied to the gradient of the particle velocity. The updated deformation gradient tensor:

$$\mathbf{F}_p^{t+\Delta t} = \sum_{i=1}^{N_n} \mathbf{x}_i^{t+\Delta t} {}_p\mathbf{G}_i(\mathbf{x}_p^t) \cdot \mathbf{F}_p^t \tag{15}$$

then find the stress increment from the constitutive model for the given strain increment and use it to update the particle stress tensor:

$$\mathbf{s}_p^{t+\Delta t} = \mathbf{s}_p^t + \Delta \mathbf{s} \tag{16}$$

Finally the cells to which each of the particles now belong to are identified, and the natural coordinates of the particles updated. This is the convective phase for the next increment, the process now repeats until a predefined termination time is reached.

Note that the numerical solution would break if $m_i^t$ is close to zero, i.e. no particles exist in the support domain of $N_i$. Therefore, when implemented in code, equations (7), (9), and (12) are not calculated if $m_i^t$ is less than a specified value [130].

#### 1.5.1.2 *Explicit and implicit MPM*

Standard MPM used an explicit time integration method. The computational grid is convected with the material points during the deformations that take place during each time-step. The grid is reset at the end of each time-step. Conditional stability of the backward Euler integration method limits the size of the time-step, Equation 9. In some scenarios this can necessitate prohibitively small time-steps. Guilkey and Weiss sought to address this limitation by introducing

an implicit version of MPM: IMPM [124]. IMPM was based on the observation that calculations on the computational grid were carried out in the same way as for finite element calculations, the material points act as integration points for the assembly. The authors exploited this to develop an implicit time integration strategy allowing for larger time-steps and a reduction in computational run-time. In IMPM the computational grid is not reset at the end of each time-step, and is instead allowed to move with the particles, retaining optimum particle distribution in relation to the grid, Figure 5. IMPM is more efficient for low rates of loading (with respect to the wavespeed of the material), for faster loading rates explicit MPM is more computationally efficient [125].

### 1.5.2 *MPMICE*

Material Point Method Implicit Continuous Fluid Eulerian (MPMICE) is an extension of MPM to incorperate the ICE (Implicit Continuous Fluid Eulerian) method, facilitating FSI modelling. ICE is a CFD formulation with full Navier-Stokes representation of fluids using an Eulerian grid, with the capability for including chemical and physical transformations between phases, such as fuel combustion. The multi-material MPMICE algorithm was developed at Los Alamos National Laboratory (NM, USA) for simulating explosions of energetic devices, where there was a need for an approach capable of handling very large material deformations over a short time involving both solid and fluid phases [134]. A detailed description of this algorithm is given is Chapter 4, Section 4.2.1.

### 1.5.3 *Lattice Boltzmann Method*

Lattice Boltzmann methods (LBM) evolved out of Lattice-Gas Cellular Automata (LGCA), statistical models based on the kinetic theory of gases [121, 135]. LBM is now considered a computational fluid dynamics (CFD) method and has been widely applied to hydrodynamic fluid flow simulations. It may be used to model single phase and multiphase flows [136]. LBM is based on the principle that macroscopic fluid behaviour is the result of the collective behaviour of microscopic particles, but is not sensitive to the underlying details of those particles. It involves building simplified kinetic models that incorporate the physics of microscopic and microscopic models in such a way that their averaged properties conform to the desired macroscopic equations. The velocity space is discretized, particle velocities are restricted to a finite set of orientations, and particles move in relation to an overlying lattice (mesh), Figure 6, greatly simplifying Boltzmann's original conceptual view. The lattice is comprised of a set of connected nodes, with state variables set at each node. The model utilises composite update rule based on collision and streaming is used, Figure 7. At each time step, particles stream in their relative directions. If more than one particle enters a site then the collision rule is applied, redistributing particles such that conservation laws are satisfied. Although theoretically simple, this method models fluid like behaviours in good agreement with experimental results and other numerical techniques [121, 137]. LBM has been successful in applications involving modelling interfacial dynamics and complex boundaries [123]. It is a technique with the potential to be used for modelling CSF in SCI simulations. LBM can produce clear physical pictures; boundary

conditions are easily implemented and in addition to being computationally efficient, it is also well suited for implementation of highly parallelised computing systems [123].

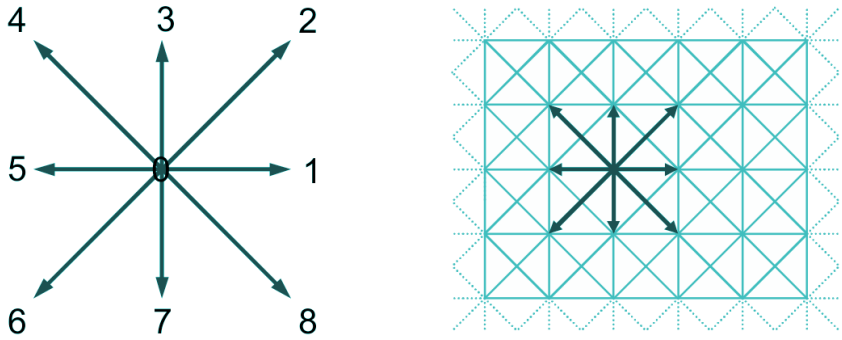



Figure 6: Discretization of velocities for a $D_2Q_9$ lattice, two dimensions, 8 discrete velocities (1-8) plus particle at rest (0) [138].

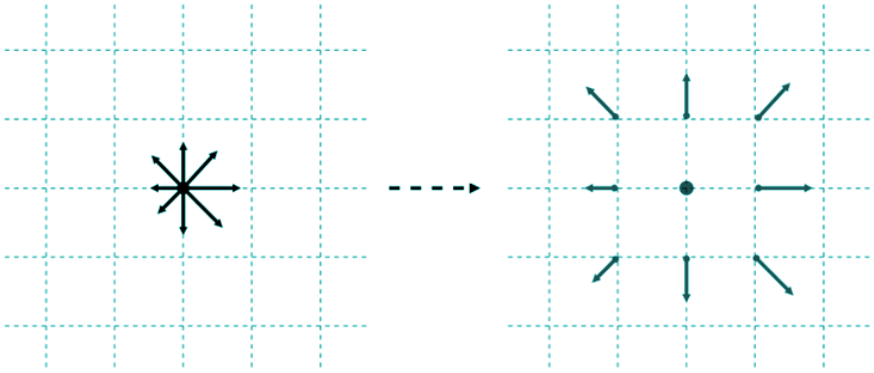
#### 1.5.3.1 *The LBM Algorithm*

The notation used to define the underlying lattice is $D_\alpha Q_\beta$, where $\alpha$ denotes space dimensionality and $\beta$ denotes the number of discrete velocities, Figure 6. The following is a description of the LBM algorithm, using a $D_2Q_9$lattice for simplicity, adapted from Begum and Basit [137] and Chirila [139].

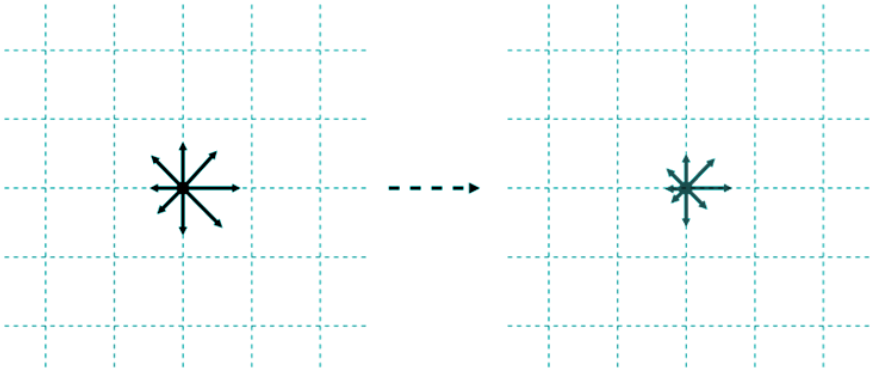
In non-equilibrium statistical mechanics the Boltzmann Equation gives a statistical description of the movement of fluids that are not in thermodynamic equilibrium, the equation is given as:

$$\frac{\partial f}{\partial t} + \nabla \boldsymbol{v} f = Q \tag{17}$$

where $f$ is the distribution function and $Q$ is the collision integral. The distribution function, $f$, depends on space, velocity, and time: $f(\boldsymbol{x}, \boldsymbol{v}, t)$.



(a) Streaming



(b) Collision

Figure 7: LBM Streaming and Collision process on a $D_2Q_9$ lattice. (a) Streaming: The magnitude of the particle distribution functions do not change, but they move to neighbouring nodes based on their direction. (b) Collision: Local density and velocity are conserved, but the distribution functions change according to the collision rules [138, 139]

For LBM the BGK (Bhatnagar Gross Krook) approximation is used for the collision operator [135, 140], giving:

$$\frac{\partial f}{\partial t} + \nabla \boldsymbol{v} f = -\tau^{-1}(f - f^{eq}) \tag{18}$$

where $\tau$ is the collision time and $f^{eq}$ is the equilibrium distribution function. The velocity space $v$ is discretized into a finite set of $i$ dis-

crete velocities: $\boldsymbol{v}_i$, the associated distribution function is governed by the following equation:

$$\frac{\partial f_i}{\partial t} + \nabla \boldsymbol{v} f_i = -\tau^{-1}(f_i - f_i^{eq}) \tag{19}$$

Macroscopic fluid density is given by:

$$\rho = \sum_{i=0}^{\beta-1} f_i \tag{20}$$

Macroscopic velocity, $\mathbf{u}$, is the average of the microscopic velocities, $\mathbf{c}_i$, and the directional densities:

$$\mathbf{u} = \frac{1}{p} \sum_{i=0}^{\beta-1} f_i \mathbf{c}_i \tag{21}$$

The two basic steps in LBM are streaming and collision, the distribution functions at each lattice point may be updated using the following equation, the left hand side is the streaming part, and the right hand side is the collision term:

$$f_i(\mathbf{x} + \mathbf{c}\Delta t, t + \Delta t) = (1 - w) f_i(\mathbf{x}, t) + w f_i^{eq}(\mathbf{x}, t) \tag{22}$$

where $w = \Delta \frac{t}{\tau}$ is the relaxation frequency, which is related to the fluid viscosity. Although they can be combined in a single statement the streaming and collision steps are often separated into two steps in actual numerical implementations, Figure 7 [139]. This is because the combined equation (22) works for lattice points within the fluid do-

main, but does not work for the domain boundaries, where boundary conditions must be used to compensate for the insufficient number of distribution functions. In the streaming step the distribution functions are translated to their neighbouring lattice points based on their respective discrete velocities. In the collision step the distribution functions are redistributed towards the local discretized Maxwellian equilibrium distribution functions in such a way that local mass and momentum is conserved, collision of fluid particles is considered a relaxation towards local equilibrium [137]. The equilibrium distribution functions can be obtained from the local Maxwell-Boltzmann probability density function [141]:

$$f_i^{eq}(\mathbf{x}) = w_i \rho(\mathbf{x})[1 + 3\frac{\mathbf{c}_i.\mathbf{u}}{c^2} + \frac{9}{2}\frac{(\mathbf{c}_i.\mathbf{u})^2}{c^4} - \frac{3}{2}\frac{\mathbf{u}^2}{c^2}] \tag{23}$$

where c is the propagation speed on the lattice (lattice units per time step), normally taken to be 1 [139]. For a $D_2Q_9$ lattice this gives the weights:

$$w_i = \begin{cases} \frac{4}{9}, & i = 0 \\ \frac{1}{9}, & 1 \leqslant i \leqslant 4 \\ \frac{1}{36}, & 5 \leqslant i \leqslant 8 \end{cases} \tag{24}$$

The weights are artificial, they are used to recover the macroscopic properties of density and momentum per unit volume. Through ap-

plication of multi-scale technique it is possible to recover the incompressible Navier-Stokes equations:

$$\nabla.\mathbf{u} = 0 \tag{25}$$

$$\rho\partial_t\mathbf{u} + \rho\mathbf{u}\nabla.\mathbf{u} = -\nabla P + \rho\nu\nabla^2\mathbf{u} \tag{26}$$

where P is pressure.

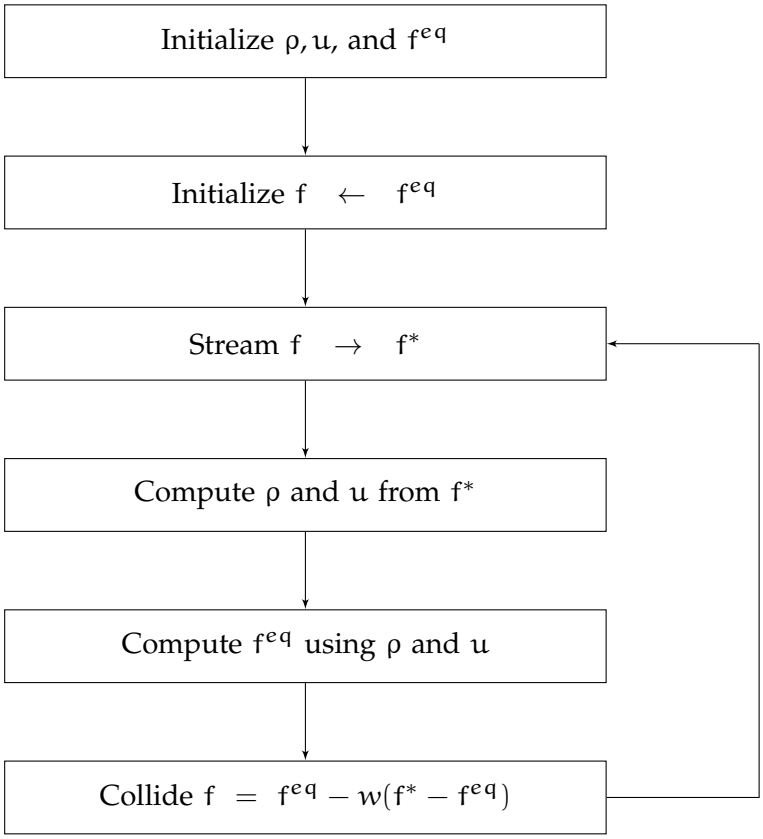



Figure 8: LBM algorithm for a $D_2Q_9$ lattice [137].

### 1.5.4 *Smoothed Particle Hydrodynamics*

Smoothed particle hydrodynamics (SPH) is a mesh free particle method initially developed by Gingold and Monaghan [142] and Lucy [143] in 1977, for studying astrophysics. It is well suited to computational fluid dynamics [133], and may used to model fluid flows such as the movement of the CSF during SCI. SPH was recently been adapted by Hoover for simulating solid mechanics problems, dubbed smoothed particle applied mechanics (SPAM) [144]. These techniques are well suited to parallelisation. Fluid is divided into a discrete number of particles (elements). SPH makes use of a smoothing function that is applied to smooth the particle properties, each of which has an associated smoothing distance. The physical quantities of an arbitrary particle can be obtained based on the sum of the surrounding particles within range of the smoothing distance. By assigning each particle its own individual smoothing length, the resolution of the overall model can be made to self adapt to loading conditions. A key advantage of this approach is that there is no need to track boundaries, a free surface for interacting fluids is generated directly [109, 133, 144].

### 1.5.5 *Mesh vs. Mesh-free Methods*

Both mesh based and mesh free methods for computational modelling have various strengths and weaknesses, some of which are summarised in Table 5. Mesh-free methods are well suited to problems that involve large localised deformations, separation of continuum, or the propagation of cracks, which are very difficult to model using conventional mesh based techniques [145]. MPM is able to avoid the

Table 5: Relative strengths and weaknesses of mesh based and mesh free methods [138].

| Mesh | Meshless |
|---|---|
| Mature technique, wide range of commercial software | Lacks numerical maturity, incorporated by few software packages |
| Advanced numerical techniques, stable and accurate | First order accuracy |
| Established method for modelling SCI | Not widely used for SCI, but successful in other engineering applications |
| Less computationally expensive | More computationally expensive |
| Scales poorly for parallel computation | Easily parallelised |
| Mesh tangling occurs with large deformations | Can handle large deformations |
| Unable to handle break-up and coalescence | Can model more complex physics, break-up and coalescence |

issue of mesh tangling, which often occurs using FEM for large local deformations, through the process by which material points are mapped to grid nodes and grid nodes mapped to material points at each timestep in the simulation [130]. Mesh-free Methods also have a key advantage in that they are easily implemented for highly parallelised computing. Modern high powered computer systems achieve improved performance using multiple processors running in parallel, rather than increasing the speed of a single processor, which is restricted by technical hardware limitations. The use of GPUs provides a low cost hardware platform for parallel computing. Going forward, algorithms amenable to computational parallelisation are better placed for creating more complex models.

### 1.5.6 *Material Point Method for Modelling Soft Tissues*

Modelling soft tissue constructs using FE methods is highly challenging due to the complex geometry, material properties, boundary conditions, especially in cases where cells embedded within an extracellular matrix are to be modelled. A key difficulty in the application of FE modelling to this kind of problem lies in the meshing process, creating an unstructured FE mesh representative of the highly complex geometry of the sample is often tedious and very time consuming [146], despite the availability of imaging techniques and image processing software. Boundaries and contacts between structure must be explicitly defined with FE contact algorithms, and mesh entanglement can occur, especially for large deformations [125]. The limitations encountered with FEM led Guilkey et al. and Ionescu et al. to investigate an alternative approach to modelling soft tissue constructs using the MPM [122, 125, 126]. These authors made use of the UINTAH computational framework, a parallel computing problem solving environment, developed at the University of Utah [147]. UINTAH provides software implementations of MPM, and is freely available to researchers.

Ionescu et al. used MPM to investigate penetrating trauma of soft tissues, specifically bullet and knife wounds [122, 126]. The aim of the study was to develop a realistic computational soft tissue model, which could be subjected to finite deformation and failure, and used to simulate various types of injury, elucidating the failure mechanisms of the tissue in response to the complex loading patterns of the injury. A slab of myocardium was modelled, the muscular middle layer of the heart wall, composed primarily of spontaneously contract-

ing cardiac fibres. The tissue was represented as a composite matrix of collagen fibres, with different strain failures defined for each. Fibre directions varied throughout the layers of tissue to match the structure of myocardial tissue. MPM was selected for this work because it simplifies the modelling of complex geometries, and can handle large deformations and fragmentation, features which are typical of penetrating trauma such as bullet and knife wounds. Explicit time integration was used. Bullet penetration simulations of the myocardial slab were performed, two modes of failure were defined: matrix failure (shear strain over 50%) and fibre failure (tensile strain over 40%), upon failure the contribution of the failed component to the stress was set to zero. Following the injury simulation the material points were separated into groupings of failed matrix, failed fibre, and intact points for analysis [122, 126].

The aim of this study by Guilkey et al. was to apply MPM modelling to a vascularised tissue construct, adapting the MPM algorithm to better handle quasi-static large deformation mechanics, using a discretization technique based on the input from confocal imaging of the sample [125]. The standard explicit MPM algorithm was modified to allow the background computational grid to remain fixed with respect to the spatial distribution of the material points during the analysis, an implicit version of MPM. Using explicit time integration, the state of the system at a later time is calculated from the state of the system at the current time. Using implicit time integration, a solution is found to an equation involving both the current state of the system and the later state of the system. Implicit methods can be more difficult to implement and incur additional computational expense. Implicit methods are more suitable in situations where an explicit

method would require impractically small timesteps, in such cases it may require less computation to achieve a certain level of accuracy, by using larger timesteps. In this case, implicit MPM is more efficient for low load rates in relation to the wavespeed of the material, while explicit MPM is superior for faster load rates [125, 128].

Microvascular constructs were grown in a collagen gel, the modified algorithm was used to investigate the biomechanics of the 3D vascularised scaffold under tension. Stress distributions and reaction forces were analysed. The reaction force was found to be highly sensitive to the modulus of the micro vessels embedded in the construct. It would be possible to estimate a suitable modulus through the simulations, using a parameter estimation scheme. Reaction force was far less sensitive to the Poisson's ratio of the entire sample. The overall model consisted of 13.6 million material points and the implementation scaled well for up to 200 processors working in parallel. The modified algorithm was found to be more robust and accurate for this application than the traditional algorithm, although the authors warn that modifying MPM to not reset the grid could potentially result in a severely distorted mesh, one of the problems MPM was developed to avoid. The process of spatial discretization was shown to be easy relative to FEM equivalents.

The advantages demonstrated by these studies, would also make MPM suitable for modelling neural tissue in the case of SCI. Although the problem of modelling the spine is complicated further by the need to model the surrounding CSF in addition the the solid tissue, this technique appears to be a strong candidate for SCI modelling, and could potentially be combined with other techniques, such as LBM (described in section 1.5.3), to handle the fluid phases.

## 1.6 CONCLUSIONS

The spinal cord is an extremely complex structure, modelling it accurately poses significant challenges. Different mechanisms of injury result in different injury patterns, in turn affecting the evolution of the secondary pathological cascade. For the development of effective medical interventions for reducing the severity of secondary injury, it is necessary to better understand the mechanics of the primary injury and how the cord interacts with the spinal column and surrounding CSF. Mesh based FEM techniques are a useful tool in investigating the biomechanics of SCI, in compliment to laboratory based studies. However, they are currently held back by computational limitations, as the number of operations required to simulate injury quickly becomes too high. The complexity is further increased when the interaction of the spinal cord with the surrounding cerebrospinal fluid is considered. Computational expense, difficulty in coping with large material deformations, difficultly in implementing the FSI, coupled with poor potential for parallelisation warrant the investigation of alternative computational approaches. Mesh-free techniques, such as MPM, LBM, SPH and SPAM are in some cases more computationally intensive. However, these techniques are well suited for handling large local material deformations and separation of continuums. Furthermore, they are highly amenable to parallelisation, the computational limitations encountered by traditional FEM techniques may be overcome using high powered parallel computing.

## 1.7 AIMS AND OBJECTIVES

This project aims to generate a validated spinal cord model for simulating SCI that overcomes the limitations of FEM spinal cord models, as discussed in section 1.7.1.The model will use the MPM/MPMICE methods, allowing execution on highly parallelised high-powered computing systems. The key project milestones are as follows

- Development of MPM SCI model with a simple representation of the cord using a single, homogeneous, isotropic material, which crudely mimics the cord in geometry and material properties
- Implementation of an Ogden constitutive material model for MPM/MPMICE
- Evaluation of the solid models against existing conventional FEM models and experimental studies
- Incorporation of the CSF and the FSI between phases using MPMICE
- Evaluation of the FSI models against existing conventional FEM models and experimental studies
- Further validation and model refinement.

### 1.7.1 *Project Rationale*

Due to limitations with FEM models of SCI, simplifications must be made. For example, commonly the properties of spinal cord tissue are simplified into a homogeneous, isotropic material, the geometry

of the model is highly simplified, and the fluid structure interaction involving the CSF is excluded [14]. Key limitations with FEM are as follows

- Inability to cope with large deformations over very short times
- Poor amenability to implementation on parallelised computing systems
- Difficulty in implementing the fluid-structure interaction between the cord and the surrounding CSF

#### 1.7.1.1 *Large deformations*

SCI simulations often involve large material deformations, over very short time spans, requiring a high number of time steps. Frequent computationally complex remeshing operations are consequently required. This again increases computation time required and can introduce additional error into the results. This issue is even more prevalent when the CSF is incorporated into the model, and FEM methods struggle to cope with large deformations and dislocations [14, 85].

#### 1.7.1.2 *Parallelisation*

Modern high powered computer systems achieve improved performance using multiple processors running in parallel, rather than increasing the speed of a single processor, which is restricted by technical hardware limitations. Traditional FEM is difficult to implement for parallel computing, generally these simulations can only run on a single processing core, subsequently, FEM techniques remain limited in their complexity. While difficult, parallelisation of FEM is possible and commercial software that can do this is available, however

the scalability may be lacking, particularly when the additional complexity of the FSI is included. Investigation is justified into alternative algorithms for modelling SCI computationally, which are amenable to parallelisation and better placed for creating more complex models. The basic principle of FEM is to divide the structure of interest into a finite number of smaller sub-structures (elements), joined together at nodes. This creates a deformable mesh of linked elements, traditional FEM is considered a mesh based technique. Alternative mesh-free techniques use a different approach; although a mesh is defined, it does not move, it acts as a frame of reference against which the movement of material points can be described [121]. Meshless techniques have the key advantage of being easily implemented for highly parallelised computing, and are inherently suited to handling large deformations [122]. The Material Point Method (MPM) and Lattice Boltzmann Method (LBM) are mesh-free algorithms that have been identified as having the potential for modelling SCI in the future. Although not yet applied to spinal biomechanics, these techniques have been successfully applied in other engineering applications [123, 124].

#### 1.7.1.3 *Fluid-structure interaction*

The effect of the CSF on the biomechanics of SCI is significant, this warrants the inclusion of CSF into SCI models [26]. To incorporate the CSF into a model it is necessary to capture the fluid structure interactions. Incorporating the fluid dynamics of the fluid and its interaction with the solid phases adds greatly to the number of calculations required. The computational expense of implementing the FSI is generally the first limiting factor, restricting the complexity of the FEM model. A simple approach is to eliminate one field, fluid or solid,

and abstract away the eliminated field to a specific motion or force at the interface between solid and fluid [84]. A partitioned approach may be used, whereby the solid and fluid mechanics are solved separately, with algebraic constraints on the boundary between the two. Alternatively, a monolithic approach can be taken, treating the solid, fluid and the interactive boundary as one system of equations [14]. In either case simplifications must be made within the confines of computational processing power. FEM is advantageous in that it is amenable to monolithic solvers for the FSI, however it is restricted due to the difficulty in scaling such models for parallel computation. The use of alternative techniques, such as the MPM coupled with LBM, have the potential to more accurately model the FSI, exploiting parallel computing to allow for greater levels of detail.

In summary, the level of detail achievable using FEM computational models is limited by its inability to cope with large material deformations, the difficulty in implementing the fluid structure interaction, and poor suitability to parallel computation. It should be noted that MPM models are often more computationally expensive than the equivalent FEM models, however, MPM scales well when parallelised. This will allow an MPM simulation to be processed more quickly, using parallel processing, than a comparable FEM simulation, even though the MPM simulation requires more calculations.

### 1.7.2 *Outcomes*

This project is expected to produce a model that will demonstrate the feasibility of using MPM/MPMICE for modelling spinal cord injury biomechanics. Once developed this model will allow a greater level

of anatomical accuracy to be incorporated compared to existing FE models. The sensitivity of the model to the inclusion of the dura and the inclusion of both the dura and CSF combined will be established by comparing results to the base model, consisting of the spinal cord tissue only. The results of the analysis will be validated against existing FE and experimental models. The eventual goal is to develop a full FSI spinal cord model, more complex, with anatomically accurate geometry, allowing a virtual model of the thoracolumbar region of the spinal cord to be constructed. The process of development, validation, and refinement will be an on-going cycle throughout the project, not simply an iterative process. The models developed for this project will provide an insight into the biomechanics of SCI, the relationship between the primary injury, the distributions of strain in the cord, and (in the longer-term) the possible neurological deficit. In addition to the model itself and code preparations, this project aims to output research publications that will be of interest to the scientific community. Research findings will be made available through conference papers for dissemination to patient groups and the wider public interest.

# 2

# CONSTITUTIVE MATERIAL MODELS

## 2.1 OVERVIEW

The hyperelastic constitutive models available within the Uintah Computational Framework (UCF) implementation of the Material Point Method (MPM) were a neo–Hookean model and a Mooney–Rivlin model. Neither of these provided a good fit to the experimental data to which these models were being fitted (Section 2.2); therefore, a hyperelastic compressible Ogden model was implemented in `C++` and integrated into the UCF implementation of MPM. Several versions of the Ogden model were implemented, while all of these are based on the same equation, the compressibility components differ. The mathematics behind these models is described in Section 2.5.

Numerous issues were encountered during the implementation of the Ogden model, early versions of the code gave rise to non-physical behaviour, which was observed when running the Colliding Disks problem (Section 2.3). A process of elimination was used to debug the code and identify the source of the problems. Where possible mathematics software (Maple Version 17, Maplesoft, Waterloo, Ontario, Canada) was used to generate the code for the given equations, in addition to working manually, as the process is complex with wide scope for human error.

A test case was created to test models during the development phase, described in Section 2.3. Using this example it was easy to determine whether an error existed in the implementation as the visualised results would display non physical behaviour (assuming that

the simulation did not crash first). Checks were also added within the code, throwing an error on the detection of non–physical behaviour. However, these were not able to catch all potential issues.

The check was simple in the case of the Eigenvectors, it made use of the mathematical relationship described in Section 2.5.2. In the case of the principal stretches, the compressible Mooney-Rivlin Model was re-written in terms of the principal stretches rather than the invariants, as described in Section 2.5.2. Equivalent behaviour to the original model showed that the principal stretch calculation was correct in accordance with the underlying equations.

Having checked the dependent functionality, the cause of the Ogden model issues was discovered to lie within the compressibility component of the Ogden strain energy density function (not the coding thereof). The variants of the compressibility component are described in Section 2.5.1. With the Ogden model complete a linear elastic model was also implemented, this was more straightforward and is described in Section 2.6. Eventually suitable constitutive models were successfully implemented for all of the component materials in the SCI simulation using MPM.

## 2.2 FITTING DATA TO HYPERELASTIC MODELS

The constitutive material model gives a mathematical description of how a material responds to mechanical loading. As with the FE model (Section 3.4.2) the mechanical properties of the neural tissue were based on stress–strain mechanical spinal cord testing data reported by Hung et al. [97].This was mechanical testing data based on compression testing of cat spinal cord tissue at traumatic loading rates. The data reported by Hung et al. was extracted from the graphs using Engage Digitizer (Version 5.1, SourceForge Inc.).

Curve fitting was performed using the `lsqcurvefit` function in MATLAB (R2012b, Mathworks, Inc.), to fit this experimental data to commonly used hyperelastic constitutive material models. Material constants were calculated for neo-Hookean, Mooney-Rivlin, and Ogden models, the uniaxial stress–strain curves for which are shown in Figure 9. In the case of the neo-Hookean and Mooney-Rivlin models a poor fit was achieved and the material constants calculated for these models did not produce the equivalent hyperelastic behaviour seen in the experimental data, Figure 9. The Ogden model provided a close approximation of the experimental results, demonstrating the appropriate hyperelastic curve. The Ogden model was selected as being best suited for modelling the spinal cord tissue, a decision supported by previous studies [19, 26, 54].

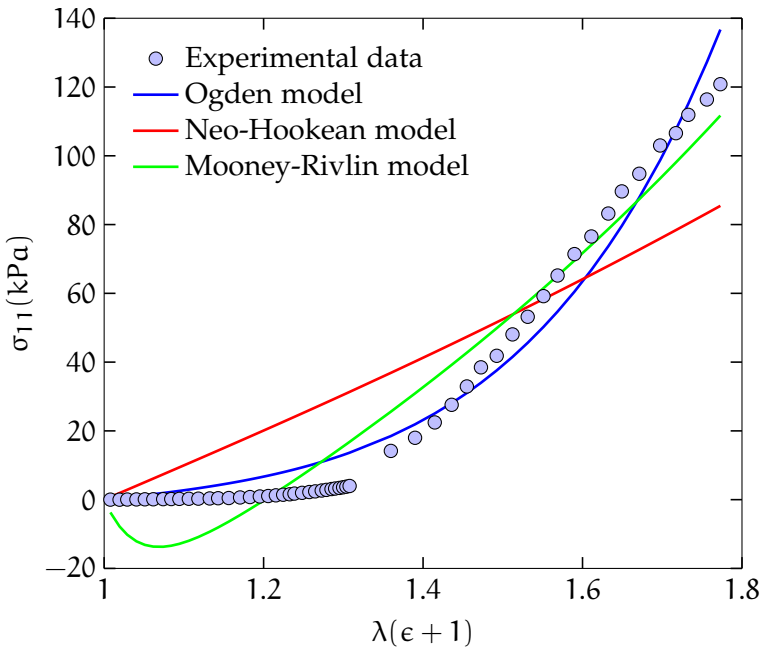
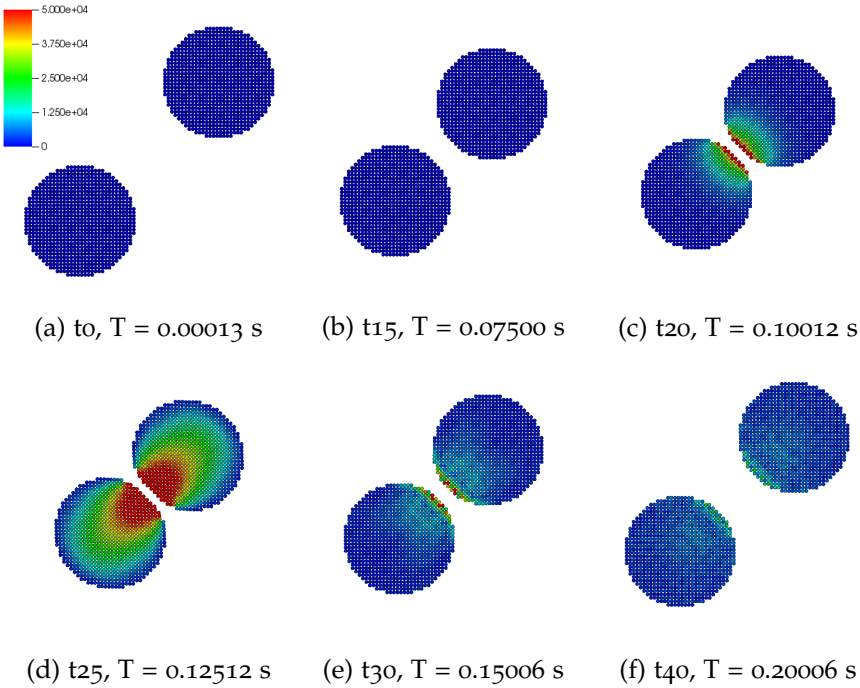



Figure 9: Experimental data reported by Hung et al. fitted to the hyperelastic models models [91]. The Ogden model provides a good fit, the other models failed to accurately capture the hyperelastic behaviour of the spinal cord.

No implementation of the Ogden model was available in the UCF (version 1.6.0). The available hyperelastic models were a compressible Mooney-Rivlin model, and a compressible neo-Hookean model.




(a) to, T = 0.00013 s (b) t15, T = 0.07500 s (c) t20, T = 0.10012 s

(d) t25, T = 0.12512 s (e) t30, T = 0.15006 s (f) t40, T = 0.20006 s

Figure 10: Test problem: two hyperelastic disks, each move towards each other at 2 $ms^{-1}$, collide, and recoil. Particles are coloured by equivalent stress (Pa).

However, these were not suitable. Being unable to accurately fit the available experimental stress–strain data to the available models the decision was taken to implement a compressible Ogden model and integrate it into the UCF implementation of MPM.

## 2.3 TEST PROBLEM - COLLIDING DISKS

A simple test case was created to help evaluate the behaviour of the material model whilst it was under development. This was adapted from an example that appeared in Sulsky's paper on MPM [128]. Two hyperelastic disks are given an initial velocity (2 $ms^{-1}$), move towards each other, collide, deform, and recoil. Figure 10 shows a visualisation of the results, the distribution of stress through the disks is as expected. A compressible Mooney-Rivlin material model `comp_mooney_rivlin` was used ($C1 = 100,000$, $C2 = 20,000$, $\nu = 0.49$) in this example.

## 2.4 HYPERELASTIC MATERIAL STRESS CALCULATION

The Cauchy stress tensor for a hyperelastic material can be derived from the strain energy density function $W(\mathbf{F})$

$$\boldsymbol{\sigma} = \frac{1}{J}\frac{\partial W}{\partial \mathbf{F}} \cdot \mathbf{F}^{T} \tag{27}$$

where $\mathbf{F}$ is the deformation gradient and $J$ is the Jacobian of the deformation gradient (the determinant of $\mathbf{F}$).

$$J = |\mathbf{F}| \tag{28}$$

If the material is incompressible then $J = 1$, this effectively switches off the compressibility components of the constitutive models. The appropriate strain energy density function, $W$, can be substituted into Equation 27 to calculate the Cauchy stress tensor according to that model. To accommodate this Equation 27 may we rewritten a number of ways depending on the form of $W$. It may be more convenient to use the invariants of the unimodular component of the left Cauchy–Green deformation tensor $\mathbf{B}$.

$$\mathbf{B} = \mathbf{F} \cdot \mathbf{F}^{T} \tag{29}$$

The invariants of $\mathbf{B}$ ($I_1$, $I_2$, and $I_3$) are defined as follows

$$I_1 = \mathrm{tr}(\mathbf{B}) = \mathbf{B}_{kk} \tag{30}$$

$$I_2 = \frac{1}{2}(I_1^2 - \mathbf{B} : \mathbf{B}) = \frac{1}{2}(I_1^2 - \mathbf{B}_{ik}\mathbf{B}_{ki}) \tag{31}$$

$$I_3 = |\mathbf{B}| = J^2 \tag{32}$$

where $|\mathbf{B}|$ denotes the determinant of $\mathbf{B}$. With the strain energy density function in the form of $W(I_1, I_2, I_3)$, Equation 27 can be rewritten as

$$\boldsymbol{\sigma} = \frac{2}{\sqrt{I_3}}[(\frac{\partial W}{\partial I_1} + I_1\frac{\partial W}{\partial I_2})\mathbf{B} - \frac{\partial W}{\partial I_2}\mathbf{B}\cdot\mathbf{B}] + 2\sqrt{I_3}\frac{\partial W}{\partial I_3}\mathbf{1} \tag{33}$$

where $\mathbf{1}$ is the identity matrix or Kronecker Delta.

Some constitutive models, such as the classic Mooney–Rivlin model, express the strain energy density function in terms of $\bar{I}_1$, $\bar{I}_2$, and $J$, these are used in compressible models where $J \neq 1$.

$$\bar{I}_1 = J^{-\frac{2}{3}} I_1 \tag{34}$$

$$\bar{I}_2 = J^{-\frac{4}{3}} I_2 \tag{35}$$

With the strain energy density function in the form of $W(\bar{I}_1, \bar{I}_2, J)$, Equation 27 can be rewritten as

$$\begin{aligned}\boldsymbol{\sigma} = \frac{2}{J}[(\frac{\partial W}{\partial \bar{I}_1} + \bar{I}_1 \frac{\partial W}{\partial \bar{I}_2})\mathbf{B} - \frac{\partial W}{\partial \bar{I}_2}\mathbf{B}\cdot\mathbf{B}] \\ +[\frac{\partial W}{\partial J} - \frac{2}{3J}(\bar{I}_1 \frac{\partial W}{\partial \bar{I}_1} + 2\bar{I}_2 \frac{\partial W}{\partial \bar{I}_2})]\mathbf{1}\end{aligned} \tag{36}$$

The Ogden model is typically expressed in terms of the principal stretches ($\lambda_1$, $\lambda_2$, and $\lambda_3$), which are related to the invariants as follows

$$I_1 = \mathrm{tr}(\mathbf{B}) = \lambda_1^2 + \lambda_2^2 + \lambda_3^2 \tag{37}$$

$$I_2 = \frac{1}{2}(I_1^2 - \mathbf{B} : \mathbf{B}) = \lambda_1^2\lambda_2^2 + \lambda_2^2\lambda_3^2 + \lambda_3^2\lambda_1^2 \tag{38}$$

$$I_3 = |\mathbf{B}| = J^2 = \lambda_1^2\lambda_2^2\lambda_3^2 \tag{39}$$

$$\bar{I}_1 = J^{-\frac{2}{3}} I_1 = J^{-\frac{2}{3}}(\lambda_1^2 + \lambda_2^2 + \lambda_3^2) \tag{40}$$

$$\bar{I}_2 = J^{-\frac{4}{3}} I_2 = J^{-\frac{4}{3}}(\lambda_1^2\lambda_2^2 + \lambda_2^2\lambda_3^2 + \lambda_3^2\lambda_1^2) \tag{41}$$

J is equal to the product of the three principal stretches, which is equal to 1 if the material is incompressible.

$$J = \lambda_1 \lambda_2 \lambda_3 \tag{42}$$

The Ogden strain energy density function is of the form $W(\lambda_1, \lambda_2, \lambda_3)$, Equation 27 may be expressed in terms of the principal stretches using a spectral decomposition for $\mathbf{B}$ in terms of its Eigenvalues $(\lambda_1^2, \lambda_2^2, \lambda_3^2)$ and Eigenvectors $(\mathbf{n}_1, \mathbf{n}_2, \mathbf{n}_3)$, which define the principal stretches and their directions.

$$\boldsymbol{\sigma} = \frac{1}{\lambda_1 \lambda_2 \lambda_3}[\lambda_1 \frac{\partial W}{\partial \lambda_1} \mathbf{n}_1 \otimes \mathbf{n}_1 + \lambda_2 \frac{\partial W}{\partial \lambda_2} \mathbf{n}_2 \otimes \mathbf{n}_2 + \lambda_3 \frac{\partial W}{\partial \lambda_3} \mathbf{n}_3 \otimes \mathbf{n}_3] \tag{43}$$

where $\otimes$ denotes the dyadic product, which produces a second order tensor, a $3 \times 3$ matrix in the practical terms of the implementation. This equation was used to calculate the Cauchy stress tensor in the implementation of the Ogden model for MPM. The left Cauchy-Green deformation tensor, $\mathbf{B}$, may be expressed in terms of its eigenvalues and eigenvectors

$$\mathbf{B} = \lambda_1^2 \mathbf{n}_1 \otimes \mathbf{n}_1 + \lambda_2^2 \mathbf{n}_2 \otimes \mathbf{n}_2 + \lambda_3^2 \mathbf{n}_3 \otimes \mathbf{n}_3 \tag{44}$$

where $\mathbf{n}_1$, $\mathbf{n}_2$, and $\mathbf{n}_3$ are mutually perpendicular unit eigenvectors of $\mathbf{B}$. As $\mathbf{B}$ is a symmetric matrix, the eigenvectors will always be mutually perpendicular unless two or more of the eigenvalues are the same. In this case the eigenvectors are not uniquely defined and

it is possible to use any set of mutually perpendicular eigenvectors [148].

$$\mathbf{n} \otimes \mathbf{n} = \mathbf{n}\mathbf{n}^{T} \tag{45}$$

## 2.5 OGDEN HYPERELASTIC MODEL

The strain energy density for the standard incompressible Ogden material model is shown in Equation 46[149].

$$W = \sum_{n=1}^{N} \frac{\mu_n}{\alpha_n}(\lambda_1^{\alpha_n} + \lambda_2^{\alpha_n} + \lambda_3^{\alpha_n} - 3) \tag{46}$$

where $N$, $\alpha$, and $\mu$ are the Ogden material constants. For a single term Ogden model ($N = 1$) this is simply

$$W = \frac{\mu}{\alpha}(\lambda_1^{\alpha} + \lambda_2^{\alpha} + \lambda_3^{\alpha} - 3) \tag{47}$$

As this is an incompressible material model the condition $J = \lambda_1\lambda_2\lambda_3 = 1$ applies. For a compressible version of the Ogden model it is necessary to remove this condition and add the volumetric strain energy density $W_V$ to the deviatoric strain energy density, such that

$$W = W_D + W_V \tag{48}$$

where $W_D$ is the incompressible Ogden strain energy density shown in Equation 47. The compressibility component of the Ogden model can be achieved in a number of ways. Several variants were imple-

mented, these are described in Section 2.5.1. For the purpose of the detailed description in this section; the following is used

$$W = \sum_{i=1}^{N} (\frac{\mu_i}{\alpha_i}[\bar{\lambda}_1^{\alpha_i} + \bar{\lambda}_2^{\alpha_i} + \bar{\lambda}_3^{\alpha_i} - 3]) + \frac{\kappa_1}{2}(J-1)^2 \tag{49}$$

where $\alpha$ and $\mu$ are the Ogden material constants, and $\kappa$ is the bulk modulus, and $\bar{\lambda}_i$ denote the reduced principal stretches.

$$\bar{\lambda}_i = \lambda_i J^{-\frac{1}{3}} \tag{50}$$

In the case of the a single term ($N = 1$), Equation 49 becomes

$$W = \frac{\mu}{\alpha}(\bar{\lambda}_1^{\alpha} + \bar{\lambda}_2^{\alpha} + \bar{\lambda}_3^{\alpha} - 3) + \frac{\kappa}{2}(J-1)^2 \tag{51}$$

For small strains, the bulk modulus, $\kappa$, equates to the equivalent Poisson's ratio, $\nu$, as follows

$$\kappa = \frac{E}{3(1-2\nu)} \tag{52}$$

where

$$E = \frac{3}{2}\mu\alpha \tag{53}$$

The Ogden strain energy density function (Equation 51) was substituted into Equation 43 to obtain the equation to calculate the Cauchy stress tensor for this constitutive model. This requires some lengthy algebra, which is shown in Appendix A.3. To minimise the chance of

error this equation was worked out both by hand and using mathematics software (Maple Version 17, Maplesoft, Waterloo, Ontario, Canada). With both workings in agreement, Maple was used to generate corresponding `C` code. Table 6 shows the error checks used to validate various aspects of the Ogden model implementation and the process of debugging.

To solve Equation 43 it is also necessary to calculate the outer product (also referred to as the tensor product) of the eigenvectors of $\mathbf{B}$, $\mathbf{n}_i \otimes \mathbf{n}_i$. This results in a $3 \times 3$ matrix, each of which is scaled according to the strain energy density function. The result is the Cauchy stress tensor, $\boldsymbol{\sigma}$, which is also a $3 \times 3$ matrix as far as the `C++` implementation is concerned.

| Component | Check |
|---|---|
| Deformation gradient | Checked that Jacobian is greater than zero at each timestep (Section 2.5.4) |
| Eigenvectors | Checked using Equation 66 (Section 2.5.3) |
| Principal Stretches | Checked using modified version of Mooney–Rivlin model (Section 2.5.2) |
| Strain energy density function | Worked out Cauchy stress by hand and by using Maple mathematics software |
| Overall material behaviour | Multiple versions of Ogden model implemented, checked using colliding disks test (Section 2.3) |

Table 6: Table showing the error checks used to develop and debug the Ogden model implementation in `C++`

.

### 2.5.1 *Ogden variants*

Several variants of a compressible Ogden model were implemented in c++ and integrated into the UCF software. Each was implemented using the same approach described in Section 2.5. However, the strain energy density functions all differ, these are shown in this section. The following is the variant available in the ADINA FE software (ADINA R & D, Inc., 71 Elton Avenue, Watertown, MA 02472 USA) [150]

$$W = \sum_{i=1}^{N} (\frac{\mu_i}{\alpha_i}[\bar{\lambda}_1^{\alpha_i} + \bar{\lambda}_2^{\alpha_i} + \bar{\lambda}_3^{\alpha_i} - 3]) + \frac{\kappa_1}{2}(J-1)^2 \tag{54}$$

where $\alpha$ and $\mu$ are the Ogden material constants, $\kappa$ is the bulk modulus, and $\bar{\lambda}_i$ denote the reduced principal stretches. This was used to model the spinal cord tissue in the MPM model.

The strain energy density function for the compressible Ogden model available in ABAQUS (SIMULIA, Providence, RI, USA) is [151]

$$W = \sum_{n=1}^{N} \frac{2\mu_n}{\alpha_n^2}(\lambda_1^{\alpha_n} + \lambda_2^{\alpha_n} + \lambda_3^{\alpha_n} - 3 + \frac{J^{-\alpha_n\beta} - 1}{\beta}) \tag{55}$$

where

$$\beta = \frac{\nu}{1-2\nu} \qquad G = \sum_{n=1}^{N} \mu_n \tag{56}$$

This variant is also known as the Ogden-Storakers hyperelastic model. This was used to model the dura mater in the MPM model

The strain energy density function for the compressible Ogden model available in FeBio (University of Utah, Salt Lake City, UT, USA) is as follows [151, 152]

$$W = \sum_{n=1}^{N} [\frac{\mu_n}{\alpha_n^2}(\lambda_1^{\alpha_n} + \lambda_2^{\alpha_n} + \lambda_3^{\alpha_n} - 3 - \alpha_n \ln J)] + \frac{\mu'}{2}(J-1)^2 \quad (57)$$

where

$$\mu' = \frac{2G\nu}{1-2\nu} \qquad G = \frac{1}{2}\sum_{n=1}^{N} \mu_n \quad (58)$$

This model was not used in the MPM model after 2D test simulations using it gave results that were further from the experimental mean pellet trajectory compared to the other Ogden variants.

The following was the first variant of a compressible Ogden model to be implemented

$$W = \sum_{i=1}^{N} (\frac{2\mu_i}{\alpha_i^2}[\bar{\lambda}_1^{\alpha_i} + \bar{\lambda}_2^{\alpha_i} + \bar{\lambda}_3^{\alpha_i} - 3]) + \frac{\kappa_1}{2}(J-1)^2 \quad (59)$$

which appeared in the book *Applied Mechanics of Solids* by Bower [148]. Despite a number of attempts this was never successfully implemented, always giving rise to non physical–behaviour. Development was abandoned in favour of the other strain energy density functions shown in this section.

### 2.5.2 *Principal stretches*

The principal stretches ($\lambda_i$) may be calculated from the eigenvalues of the right or the left stretch tensor ($\mathbf{U}$ and $\mathbf{V}$ respectively). These may

also be calculated from the square roots of the eigenvalues of the right or the left Cauchy-Green deformation tensor (**C** and **B** respectively) [148]. In this implementation, the principal stretches were calculated from the square roots of the eigenvalues of **B**

$$\lambda_1 = \sqrt{e_1}, \quad \lambda_2 = \sqrt{e_2}, \quad \lambda_3 = \sqrt{e_3} \tag{60}$$

A test was created to test that the principal stretches were being correctly calculated in the software implementation. The existing UCF implementation of a compressible Mooney-Rivlin model was modified such that the first invariants were calculated using

$$I_1 = \lambda_1^2 + \lambda_2^2 + \lambda_3^2 \qquad I_2 = \lambda_1^2\lambda_2^2 + \lambda_2^2\lambda_3^2 + \lambda_3^2\lambda_1^2 \tag{61}$$

instead of using

$$I_1 = \mathrm{tr}(\mathbf{B}) \qquad I_2 = \frac{1}{2}(I_1^2 - \mathbf{B} : \mathbf{B}) \tag{62}$$

hence rewriting $W$ as

$$\begin{aligned} W = C_1(J^{-\frac{2}{3}}(\lambda_1^2 + \lambda_2^2 + \lambda_3^2) - 3) \\ + C_2(J^{-\frac{4}{3}}(\lambda_1^2\lambda_2^2 + \lambda_2^2\lambda_3^2 + \lambda_3^2\lambda_1^2) - 3) \\ + D(J-1)^2 \end{aligned} \tag{63}$$

instead of

$$W = C_1(\bar{I}_1 - 3) + C_2(\bar{I}_2 - 3) + D(J-1)^2 \tag{64}$$

which may be substituted into Equation 43 to determine the Cauchy stress. This requires some lengthy algebra, a full working is shown in Appendix A.4.

The modified Mooney-Rivlin model, where the first invariants were calculated using the principal stretches, was compared to the standard Mooney-Rivlin model, where the first invariants are calculated from $\mathbf{B}$. The colliding disks test problem (described in Section 2.3) was run using both the original and the modified constitutive model. The von Mises stress for each particle in the first simulation was compared to that of the corresponding particle in the second simulation. The sum of squared errors (SSE) was calculated for all particles for each timestep. The von Mises stress (or equivalent stress), in terms of Cauchy stress components, is given by

$$\begin{aligned}\sqrt{\sigma_{vm}} = \frac{1}{2}[&(\sigma_{11}-\sigma_{22})^2+(\sigma_{22}-\sigma_{33})^2+\\ &(\sigma_{33}-\sigma_{11})^2+6(\sigma_{12}^2+\sigma_{23}^2+\sigma_{31}^2)]\end{aligned} \tag{65}$$

Figure 11 shows a comparison equivalent stress over time for a single particle. It shows very strong agreement between the original Mooney-Rivlin model using and the modified version using principal stretches calculated from the square roots of the eigenvalues of $\mathbf{B}$. This confirms that the principal stretches are being calculated correctly. Table 7 shows the results of this comparison over all particles, the error is minute and can be attributed to rounding accuracy. From this it is possible to conclude that the code for calculating the principal stretches works correctly.

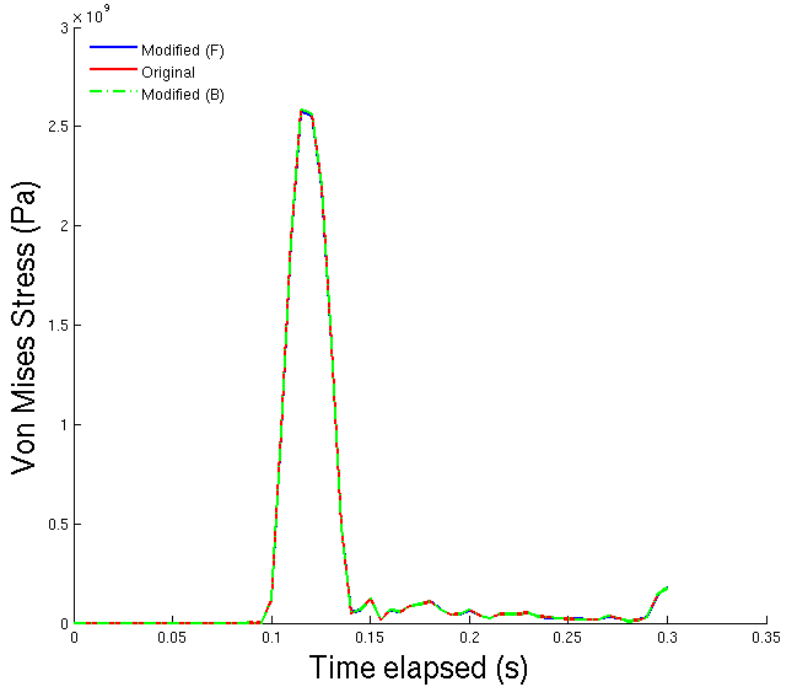



Figure 11: Equivalent (von Mises) stress over time for a single selected particle in the colliding disks simulation.

| | SSE Original vs Modified |
|---|---|
| Mean Average | 0.034072 |
| Total | 110.666 |
| Minimum | $4.51118 \times 10^{-45}$ |
| Maximum | 40.8123 |

Table 7: Comparison of the Original vs Modified Mooney–Rivlin models. Sum of Squared Errors summed over every particle over the entire duration of the colliding disks simulation.

### 2.5.3 *Eigenvectors*

The principal stretches were calculated from the square roots of the eigenvalues of $\mathbf{B}$, to solve Equation 43 and the calculate Cauchy stress the corresponding eigenvectors must also be calculated. These relate to $\mathbf{B}$ as follows

$$\mathbf{B} = \mathbf{F}\mathbf{F}^T = \lambda_1^2 \mathbf{n}_1 \otimes \mathbf{n}_1 + \lambda_2^2 \mathbf{n}_2 \otimes \mathbf{n}_2 + \lambda_3^2 \mathbf{n}_3 \otimes \mathbf{n}_3 \tag{66}$$

This relationship was used to check that the Eigenvectors were being calculated correctly. $\mathbf{B}$ was constructed in both ways (from the deformation gradient, and from the eigenvalues and eigenvectors) and compared at each timestep, for each particle. A check was implemented to throw an error if the difference between these two forms exceeded $1 \times 10^{-12}$. In this way it was verified that the eigenvectors (and eigenvalues) were being calculated correctly.

### 2.5.4 *Jacobian of deformation gradient*

The Jacobian of the deformation gradient, J, provides a measure of volume change caused by a deformation. J is equal to the product of the principal stretches, the determinant of $\mathbf{F}$, and the square root of the determinant of $\mathbf{B}$

$$J = \lambda_1 \lambda_2 \lambda_3 = |\mathbf{F}| = \sqrt{|\mathbf{B}|} \tag{67}$$

Note that

$$\rho = \frac{\rho_0}{J} \tag{68}$$

where $\rho$ is mass density and $\rho_0$ is the undeformed mass density [148]. For an incompressible material $J = 1$ as volume remains constant. For a compressible material J must satisfy the condition $J > 0$. If J is smaller than or equal to zero then this is indicative of non–physical behaviour. This check was implemented at each timestep, if J does not meet this condition then an error is thrown and the simulation is halted to prevent it continuing unnecessarily and producing results that are not physically admissible.

## 2.6 LINEAR ELASTIC MODEL

A compressible linear elastic model was implemented and integrated into the UCF. For an isotropic linear elastic material, ignoring the thermal expansion coefficient, the Cauchy stress may be calculated as follows [148]

$$\boldsymbol{\sigma} = \frac{E}{1+\nu}(\epsilon_{ij} + \frac{\nu}{1-2\nu}\epsilon_{kk}\mathbf{1}) \tag{69}$$

where $\nu$ is the Poisson's Ratio, $E$ is the elastic modulus, $\epsilon_{ij}$ is the infinitesimal strain tensor, and $\mathbf{1}$ is the identity matrix

$$\epsilon_{ij} = (\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i})/2 \tag{70}$$

Note that $\epsilon_{kk}$ is shorthand notation for $\sum_k \epsilon_{kk}\delta_{ij}$ using Einstein summation convention, in practice this constitutes the infinitesimal strain

tensor multiplied by a $3 \times 3$ identity matrix. Use of the infinitesimal strain tensor is only appropriate if the solid only experiences very small deformations. This linear elastic model is required to handle large deformations, the infinitesimal strain tensor in Equation 69 is therefore replaced by the Lagrangian finite strain tensor (also referred to as the Green-Lagrangian strain tensor or as the Green–St–Venant strain tensor). The Lagrangian finite strain tensor, $\mathbf{E}$, is suitable for large deformations and may be calculated as follows

$$\mathbf{E} = \frac{1}{2}(\mathbf{F}^{\mathsf{T}}\mathbf{F} - \mathbf{1}) \tag{71}$$

where $\mathbf{F}$ is the deformation gradient, $\mathbf{F}^{\mathsf{T}}$ is the transpose of the deformation gradient, and $\mathbf{1}$ is the identity matrix. This material model was used to model the impactors and the backplate.

## 2.7 MOONEY-RIVLIN MODEL

As part of the debugging process an incompressible Mooney–Rivlin model was implemented, a compressible version of the Mooney–Rivlin model was then also implemented. These were not used for the SCI model (in favour of the Ogden model), but were useful for preliminary tests and for debugging purposes. A compressible Mooney–Rivlin did also exist within the UCF, the details of which were undocumented. The code was reverse engineered to recover the strain energy density function. The compressible Mooney–Rvilin model implemented here differs from the existing one by its compressibility component, the existing Mooney–Rivlin model was restrictive in that it would fail if either C1 or C2 were set below zero.

Having a working Mooney–Rivlin model was useful in the development of the Ogden model. Much of the code was the same for both

models, transplanting sections of code was performed to test specific areas of functionality. In particular, the calculation of the principal stretches, the calculation of the Eigenvectors, and the implementation of the general equation for calculating Cauchy Stress (in terms of principal stretches) in a hyperelastic material (Equation 43). Both incompressible a compressible Mooney-Rivlin models were implemented and integrated into the UCF software. The strain energy for a compressible Mooney–Rivlin hyper material is given by

$$W = C_1(\bar{I}_1 - 3) + C_2(\bar{I}_2 - 3) + D(J-1)^2 \tag{72}$$

where

$$\frac{\partial W}{\partial \bar{I}_1} = C_1, \quad \frac{\partial W}{\partial \bar{I}_2} = C_2, \quad \frac{\partial W}{\partial J} = -2D(J-1) \tag{73}$$

The bulk and shear moduli relate to the constants as follows

$$\kappa = 2D \tag{74}$$

$$\mu = 2(C_1 + C_2) \tag{75}$$

Using this in conjunction with Equation 36 the Cauchy stress is given by

$$\begin{aligned}\boldsymbol{\sigma} = \frac{2}{J}[\frac{1}{J^{\frac{2}{3}}}(C_1 + \bar{I}_1 C_2)\mathbf{B} - \frac{1}{J^{\frac{4}{3}}}C_2\mathbf{B}\cdot\mathbf{B}] \\ +[2D(J-1) - \frac{2}{3J}(C_1\bar{I}_1 + 2C_2\bar{I}_2)]\mathbf{1}\end{aligned} \tag{76}$$

If the material is incompressible then $J = 1$ and Equation 72 becomes simply

$$W = C_1(\bar{I}_1 - 3) + C_2(\bar{I}_2 - 3) \tag{77}$$

and the Cauchy stress is then given by

$$\boldsymbol{\sigma} = 2(C_1 + \bar{I}_1 C_2)\mathbf{B} - 2C_2\mathbf{B}\cdot\mathbf{B} - \frac{2}{3}(C_1\bar{I}_1 + 2C_2\bar{I}_2)\mathbf{1} \tag{78}$$

## 2.8 AUTOMATIC TIMESTEPPING

To reduce computation time the software attempts to use the largest allowable timestep whilst maintaining stability. The size of the timestep depends on the computational cell dimensions, the particle velocity, and the dilational wave speed, $C_{dil}$, in the materials. The size of the timestep is therefore calculated within the constitutive material models, when there are multiple materials present then each calculates its maximum allowable timestep and the smallest of these is used. How the dilatational wavespeed is calculated depends on the nature of the constitutive model, for the single term Ogden model it is calculated using

$$C_{dil} = \sqrt{\mu(\alpha\lambda_1^{\alpha-1} + \frac{1}{2}\alpha\lambda_1^{-\frac{1}{2}\alpha-1})/\rho} \tag{79}$$

where $\lambda_1$ is the first principal stretch, and $\alpha$ and $\mu$ are the Ogden material constants, and $\rho$ is the density [149]. For the Linear elastic model $C_{dil}$ is given by the Newton-Laplace equation

$$C_{dil} = \sqrt{\kappa/\rho} \tag{80}$$

where $\kappa$ is the bulk modulus and $\rho$ is the density.

Dilational wavespeed is calculated within the constitutive material models, thus the implementations of the material models described in this chapter contain this functionality. A timestep multiplier is also specified within the input file, which roughly corresponds to the Courant-Friedrichs-Lewy (CFL) number. This provides some additional control over the timestepping, and was set below zero to scale down the size of the timesteps and increase stability. The CFL condition, in one dimension, is given as

$$C = \frac{\mu \Delta t}{\Delta x} \leqslant C_{max} \tag{81}$$

where $C$ is the Courant number, which is dimensionless, $\mu$ is the magnitude of the velocity, $\Delta t$ is the size of the timestep, and $\Delta x$ is the length interval. The CFL condition must be met in the solving of partial differential equations and dictates the maximum allowable size of the timestep for the simulation to provide physically permissible results.

## 2.9 CONCLUSIONS

The constitutive models are essential in defining the mechanical behaviour of the materials used in the MPM SCI model. Hyperelastic

compressible Ogden models were implemented to model the spinal cord tissue and dura mater. A compressible linear elastic model was implemented for the purposes of modelling the impactors and back-plate. Numerous issues were encountered during the Ogden model development, a process of elimination was used to debug the code and ensure the necessary calculations were being performed correctly. Variable timestep size calculation was also implemented for these models, reducing overall computation time.

# 3

# SOLID MODEL DEVELOPMENT

## 3.1 OVERVIEW

This section discusses the development of the solid base model (i.e. excluding the cerebrospinal fluid, which is included later). Initially this was a bare cord model (spinal cord tissue only excluding the surrounding layers), the dura mater is also added once the bare cord model was completed. The aim was to reproduce an experimental SCI model computationally, using the Material Point Method (MPM) instead of the traditional approach using Finite Element Analysis (FEA). The MPM software used for this work, the Uintah Computational Framework (UCF), is introduced in Section 3.2. The experimental configuration on which the MPM model was based is described in Section 3.4.1, a complimentary FE model based on these experiments is described is Section 3.4.2.

Initial work focussed on creating a computational geometry equivalent to this configuration, this is described in Section 3.5. Having established this, the focus moved on to determining and selecting other parameters, including parallelisation (Section 3.11), contact algorithms (Section 3.9), and boundary conditions (Section 3.8).

With the basic problem set–up in place the next step was to define material behaviour using suitable constitutive material models and parameters. The spinal cord exhibits hyperelastic material behaviour, unfortunately an appropriate hyperelastic constitutive model was not available within the UCF; therefore, additional constitutive models were implemented in C++ and integrated into the UCF software. These

models and the equations on which they are based are described in detail in Chapter 2.

With this new capability it was possible to create the bare cord SCI simulation using MPM, this was then refined and optimised for performance. This solid model, with and without the dura mater, was validated against the existing experimental models

## 3.2 COMPUTATIONAL MODEL

The software selected for this research is the Uintah Computational Framework (UCF), which is a parallel computing problem solving environment developed at the University of Utah [147]. The UCF was chosen as it provides a sophisticated implementation of the Material Point Method (MPM) algorithm, and the source code may be obtained freely and modified under the an open source software license (MIT Licence).

The UCF includes an implementation of MPM that has been demonstrated to scale well over thousands of processors working in parallel [153, 154]. This allows MPM simulations to make use of parallel computing to enable computationally expensive modelling within a reasonable time frame. Data visualisation can be achieved using VisIt, an open source data visualisation tool suited to large scale data sets. A VisIt plug-in is available that allows the software to interpret UCF output data (Uintah Data Archive format).

## 3.3 THE MATERIAL POINT METHOD

The general MPM algorithm was described in Chapter 1 Section 1.5.1, the UCF specific version of MPM will be detailed here, the description is based on the UCF documentation and the literature [128, 129,

155, 156]. As described previously, initially the materials in the MPM simulation are discretised into a finite number of material points, i.e. generation of the point cloud. Each particle carries all of its own state variables, which (at the least) include position ($\mathbf{x}_p$), mass ($m_p$), volume ($v_p$), velocity ($\mathbf{v}_p$), stress ($\boldsymbol{\sigma}_p$), and deformation gradient ($\mathbf{F}_p$). Mass typically remains constant throughout. The following charts the progression of a single timestep. Initially the particle states are mapped to the grid nodes (individually referred to as $p$ and $i$ respectively) using the shape function, $S_{ip}$ (the shape function of node $i$ evaluated at the position of particle $p$). The choice of the shape function is significant, this is discussed in Section 3.3.1.

$$m_i = \sum_p S_{ip} m_p \tag{82}$$

$$\mathbf{v}_i = \frac{\sum_p S_{ip} m_p \mathbf{v}_p}{m_i} \tag{83}$$

$$\mathbf{F}_i^{ext} = \sum_p S_{ip} \mathbf{F}_p^{ext} \tag{84}$$

Following these operations, $\mathbf{F}^{int}$ is calculated using the volume integral of the divergence of stress on the particles.

$$\mathbf{F}^{int} = \sum_p \mathbf{G}_{ip} v_p \tag{85}$$

where $\mathbf{G}_{ip}$ is the gradient of the shape function, $S_{ip}$ (the shape function of node $i$ evaluated at the position of particle $p$), $\mathbf{F}^{int}$ is a second

order tensor obtained from the dyadic product of $\mathbf{G}_{ip}$ and $\boldsymbol{v}_p$. It is then possible to calculate the acceleration as follows

$$\boldsymbol{a}_i = \frac{\mathbf{F}_i^{ext} - \mathbf{F}_i^{int}}{m_i} \tag{86}$$

The time advanced grid velocity, $\boldsymbol{v}_i^L$, is calculated using an explicit forward Euler method for time integration,

$$\boldsymbol{v}_i^L = \boldsymbol{v}_i + \boldsymbol{a}_i \Delta t \tag{87}$$

this is then used to compute a velocity gradient for each particle.

$$\nabla \boldsymbol{v}_p = \sum_i \mathbf{G}_{ip} \boldsymbol{v}_i^L \tag{88}$$

where $\nabla \boldsymbol{v}_p$ is a second order tensor. The particle velocity gradient is used to calculate an incremental particle deformation gradient,

$$d\mathbf{F}_p^{n+1} = \mathbf{I} + \nabla \boldsymbol{v}_p \Delta t \tag{89}$$

which is in turn used to calculate the particle volume, $\boldsymbol{v}_p^{n+1}$, and particle deformation gradient, $\mathbf{F}_p^{n+1}$, at timestep $n+1$.

$$\boldsymbol{v}_p^{n+1} = Det(d\mathbf{F}_p^{n+1}) \boldsymbol{v}_p^n \tag{90}$$

$$\mathbf{F}_p^{n+1} = d\mathbf{F}_p^{n+1} \mathbf{F}_p^n \tag{91}$$

Where $\mathrm{Det}()$ denotes the determinant. The time advanced stress calculation is performed by the relevant constitutive material model, based on either the velocity gradient or deformation gradient (in this case the deformation gradient). The constitutive models used for the SCI model are the subject of Chapter 2.

$$\boldsymbol{\sigma}_p^{n+1} = \boldsymbol{\sigma}_p(\mathbf{F}_p^{n+1}) \tag{92}$$

Finally, the particle velocities and positions are explicitly updated.

$$\boldsymbol{v}_p(t+\Delta t) = \boldsymbol{v}_p(t) + \sum_i S_{ip}\,\boldsymbol{a}_i \Delta t \tag{93}$$

$$\boldsymbol{x}_p(t+\Delta t) = \boldsymbol{x}_p(t) + \sum_i S_{ip}\boldsymbol{v}_i^L \Delta t \tag{94}$$

Thus concludes the progression of a single timestep, with all of the state variables having been updated (except for mass, which remained constant). Conceptually, in the description of MPM, the grid deforms and is reset to its original position at the end of each timestep. In practice this is not necessary as the the movement of particles is computed using Equation 94, the computational grid serves only as a frame of reference. Contact between materials is handled by contact algorithms, this is discussed in Section 3.9.

### 3.3.1 *Shape Function*

In MPM, the shape function, or interpolation function, handles the interaction between the particles and the underlying grid, the choice of

shape function is significant. Typically linear functions are used (trilinear in the case of three dimensional simulations). However, a generalisation of MPM, referred to as Generalized Interpolation Material Point Method (GIMP), has been reported by Bardenhagen and Kober and offers a more stable and accurate approach [157]. MPM being considered a subset of the GIMP family of methods, in these methods the particles are represented by a characteristic function, $\chi_p$, and the grid nodes using a shape function, $S_i$. The differences between the GIMP approach versus the traditional approach is described here. A convolution of $\chi_p$ and $S_i$ gives the effective shape function, $\bar{S}_{ip}$, shown in Equation (95).

$$\bar{S}_{ip}(\mathbf{x}_p) = \frac{1}{V_p} \int_{\Omega_p \cap \Omega} \chi_p(\mathbf{x} - \mathbf{x}_p) S_i(\mathbf{x}) \, d\mathbf{x}. \tag{95}$$

In MPM, $S_i$ is typically given by

$$S_i(x) = \begin{cases} 1 + (x - x_i)/h & -h < x - x_i \leqslant 0 \\ 1 - (x - x_i)/h & 0 < x - x_i \leqslant h \\ 0 & \text{otherwise,} \end{cases} \tag{96}$$

where $x_i$ is the vertex location and $h$ is the cell width. For simplicity, Equation (96) the descriptions in this section are limited to one dimension, multi-dimensional versions can be obtained using the tensor product of the one dimensional equations in the orthogonal directions. Traditional MPM is recovered when the Dirac delta function is selected as the characteristic function,

$$\chi_p(\mathbf{x}) = \delta(\mathbf{x} - \mathbf{x}_p) V_p \tag{97}$$

where $\mathbf{x}_p$ is the particle position, and $V_p$ is the particle volume. The effective shape function is then still that given by Equation (96). The gradient of this effective shape function is given by

$$G_i(x) = \begin{cases} 1/h & -h < x - x_i \leqslant 0 \\ -1/h & 0 < x - x_i \leqslant h \\ 0 & \text{otherwise,} \end{cases} \tag{98}$$

Figures 12 13 show plots of Equations (96) and (98), note the sharp discontinuity in the gradient seen in Figure 13. This gives rise to numerical artifact noise which can result in poor accuracy and stability. The noise occurs due to the discontinuity in the gradient of the linear interpolation function, arising when material points cross the boundaries on the computational cells comprising the underlying grid. This affects the evaluation of the constitutive response and results in non-physical local variations at the material points [157].

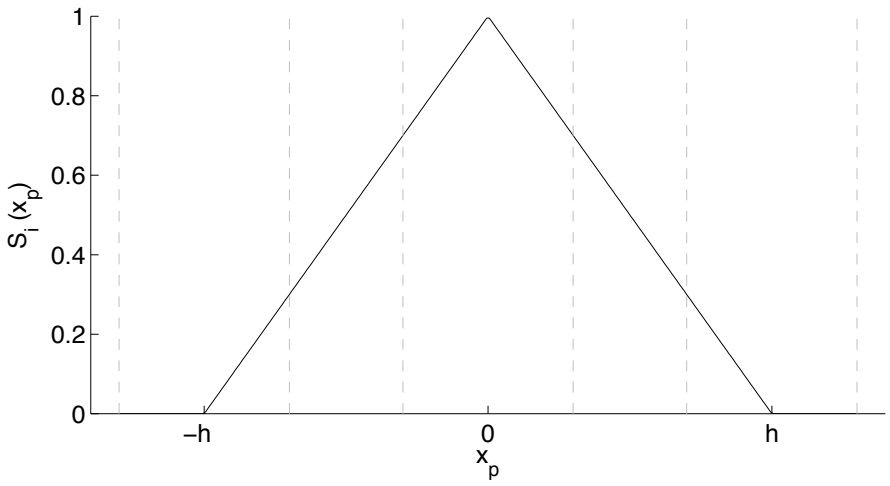



Figure 12: Effective shape function when using traditional MPM [156].

The GIMP method addresses the issue of numerical artifact noise using an effective shape function with smoother gradient. When using GIMP, the linear shape function (Equation (96)) is typically used in

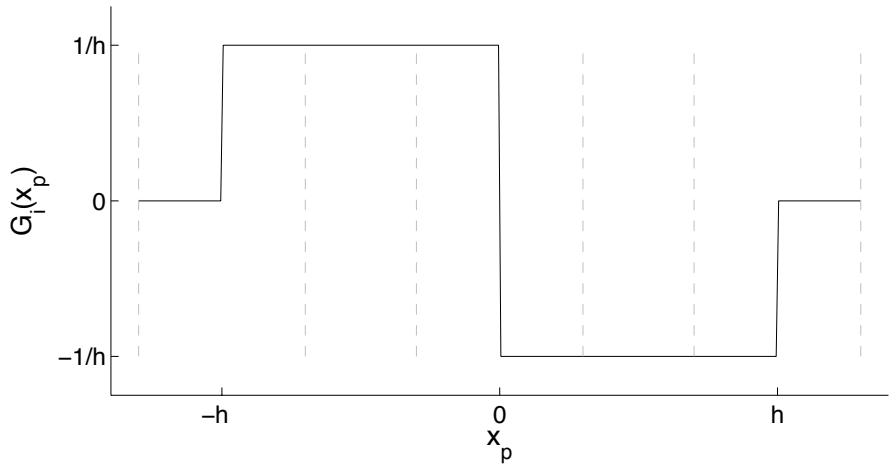



Figure 13: Gradient of the effective shape function when using traditional MPM [156].

conjunction with a *top hat* characteristic function, in one dimension this is given by

$$\chi_p(x) = H(x - (x_p - l_p)) - H(x - (x_p + l_p)) \tag{99}$$

where $H(x)$ is the Heaviside function ($H(x) = 0$ if $x < 0$ and $H(x) = 1$ if $x \geqslant 0$) and $l_p$ is the particle half-length, which is given by

$$l_p^n = F_p^n l_p^0 \tag{100}$$

where $F_p^n$ is the particle deformation gradient at time $n$. A fixed particle size is assumed here as it is in the UCF implementation of MPM, although the method does permit variable particle sizes. The convolu-

tion shown in Equation (95) is performed on Equations (96) and (95) to obtain a closed form effective shape function,

$$S_i(x_p) = \begin{cases} \frac{(h+l_p+(x_p-x_i))^2}{4hl_p} & -h-l_p < x_p - x_i \leqslant -h+l_p \\ 1+\frac{(x_p-x_i)}{h} & -h+l_p < x_p - x_i \leqslant -l_p \\ 1-\frac{(x_p-x_i)^2+l_p^2}{2hl_p} & -l_p < x_p - x_i \leqslant l_p \\ 1-\frac{(x_p-x_i)}{h} & l_p < x_p - x_i \leqslant h-l_p \\ \frac{(h+l_p-(x_p-x_i))^2}{4hl_p} & h-l_p < x_p - x_i \leqslant h+l_p \\ 0 & \text{otherwise,} \end{cases} \tag{101}$$

the gradient of which is

$$G_i(x_p) = \begin{cases} \frac{h+l_p+(x_p-x_i)}{2hl_p} & -h-l_p < x_p - x_i \leqslant -h+l_p \\ \frac{1}{h} & -h+l_p < x_p - x_i \leqslant -l_p \\ -\frac{(x_p-x_i)}{hl_p} & -l_p < x_p - x_i \leqslant l_p \\ -\frac{1}{h} & l_p < x_p - x_i \leqslant h-l_p \\ -\frac{h+l_p-(x_p-x_i)}{2hl_p} & h-l_p < x_p - x_i \leqslant h+l_p \\ 0 & \text{otherwise,} \end{cases} \tag{102}$$

Figures 14 15 show plots of Equations (101) and (102), note that the discontinuity seen in Figure 13 is not present, the gradient is much smoother. This continuity of the gradient reduces the noise making the method more accurate and robust. In this way GIMP makes MPM viable for problems in which strength is a significant component, and becomes increasingly vital for large deformation problems. For these

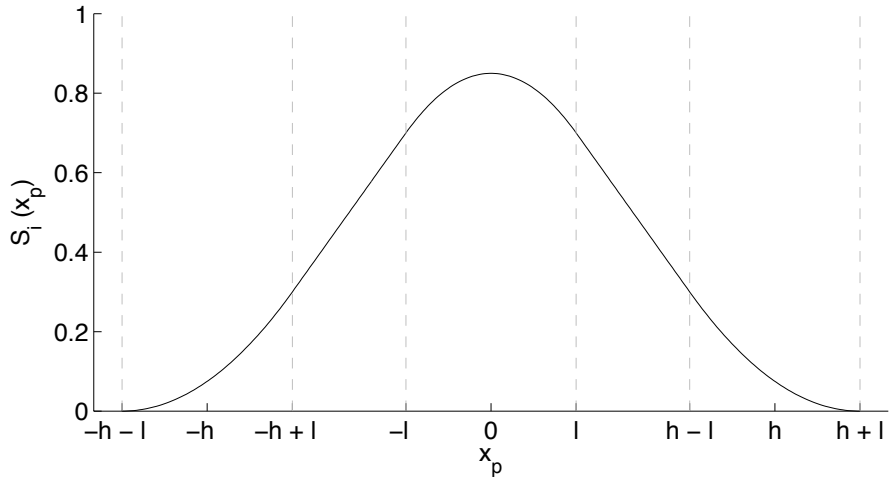



Figure 14: Effective shape function when using GIMP [156].

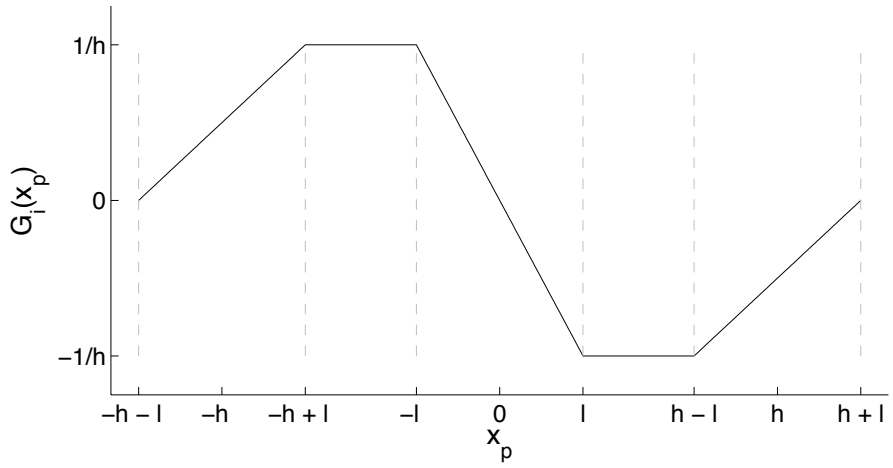



Figure 15: Gradient of the effective shape function when using GIMP [156].

reasons the GIMP approach was highly preferable to linear interpolation for the SCI model.

## 3.4 EXISTING MODELS

### 3.4.1 *Experimental Model*

The MPM model was validated against existing experimental and FEA models. Vertebral burst fracture models have been reported by Persson [71, 73]. The experimental model involved the simulation

of a transverse contusion injury representative of a vertebral burst fracture, where 140mm sections of thoracolumbar bovine spinal cord were positioned in front of a steel backplate, representing the posterior element of the spinal canal, each end fixed with clamps.

A simulated bone fragment (pellet/impactor), moving at 4.5m/s, transversely impacted the surface of the spinal cord specimens. These pellets were crafted from TUFNOL Grade 6F/45, a cotton fabric based epoxy laminate with mechanical properties similar to bone [71, 158]. Three pellets with different morphologies were used, the dimensions of which are shown in Figure 16. Circular pellet faces were used to avoid end effects, the area of the impact face of Pellet 2 is 50% of P1, while P3 is 25% of P1 (Table 8). The mass of all the pellets is equal, hence the impact energy or each pellet is equal, assuming the same impact velocity (Equation 103). The experimental burst fracture model, using these pellet dimensions, found that a decreasing the area of the impact face lead to an increase in maximum spinal cord deformation [71].

$$E = \frac{1}{2}mv^2 \tag{103}$$

Each spinal cord specimen was tested in three configurations: the bare spinal cord (dura mater and fluid removed), the cord with dura mater (CSF drained), the cord with both the dura mater and CSF (the CSF filling the subdural space). Each configuration was tested with each of the 3 impactors, with each specimen receiving 9 impacts in total. 16 specimens were tested in total [71, 73]. The deformation of the cord was measured by tracking the leading face of the impactor using a high speed camera placed at a right angle to the axis of impact. The trajectory of the pellet, from first impact to the point at which the impactor recoils (leaving contact with the surface), gives a

| | Natural bone fragment | P1 | P2 | P3 |
|---|---|---|---|---|
| Density $(kg/m^3)$ | $1.84(\pm 0.33) \times 10^3$ | $1.36 \times 10^3$ | $1.36 \times 10^3$ | $1.36 \times 10^3$ |
| Mass $(g)$ | $5.8(\pm 1.7)$ | 7 | 7 | 7 |
| Impact area $(mm^2)$ | unknown | 314 | 156 | 78 |

Table 8: Bone fragment properties [73].

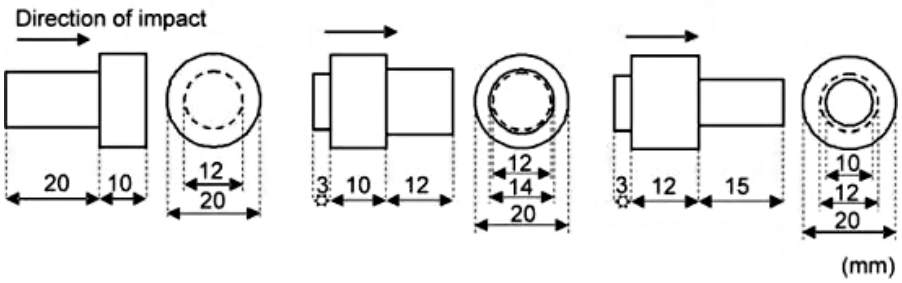



Figure 16: The dimensions of the 3 simulated bone fragments used by Persson [73]. Referred to as P1, P2, and P3 (from left to right).

reliable indication of the mechanical response of the spinal cord specimen in response to the traumatic impact. The results were plotted as deformation over time curves. These curves were used to validate the computational models.

### 3.4.2 *Finite Element Model*

Persson et al. also created a FEA model, to compliment the experimental model, as part of an investigation into the effect of the thickness of the CSF subdural layer on the spinal cord deformation [26, 73]. The model was created using ADINA (Version 8.5, ADINA R&D Inc., Watertown, MA), a FE based monolithic FSI modelling software package. The geometry mimics the experimental model described in Section 3.4.1. The constitutive material models used in this FEA model and their relevant parameters are shown in Table 9.

| Material | Constitutive model | Density ($kg/m^3$) | Poisson's ratio ($\nu$) |
|---|---|---|---|
| Cord Tissue | Ogden $\mu = 2kPa$, $\alpha = 9$ | 1050 | 0.4 |
| Dura Mater | Linear $E = 80MPa$ | 1000 | 0.49 |
| Impactor | Linear $E = 6.5GPa$ | 1360 | 0.3 |
| Backplate | Linear $E = 193GPa$ | 8000 | 0.3 |

Table 9: Material models and parameters used in the FEA model [73].

## 3.5 MODEL GEOMETRY

### 3.5.1 *Spinal Cord*

To generate the initial geometry for the simulation, it is possible to specify geometric objects from which the point cloud could be generated. The spinal cord was represented as an elliptical cylinder, diameter ranging from 10 mm to 15 mm, spanning the length of the computational domain from top to bottom ($z+$ to $z-$). The initial geometry of the cord was generated using an ellipsoid geometric object, the software does not provide the option for an elliptical cylinder. An ellipsoid with an extremely long length in the $z$ direction ($1 \times 10^{12} m$), the areas of the ellipsoid that exceeded the boundaries of the computational domain were truncated, leaving the elliptical cylinder with the desired dimensions. The spinal cord construct was positioned immediately in front of the backplate, which simulates the posterior element.

### 3.5.2 *Impactors*

The pellets modelled here are representative of the bone fragments that are propelled into the spinal canal during a vertebral burst frac-

ture. Pellet geometries were created according to the dimensions shown in Figure 16. It is possible to create more complex geometries though the union of geometric objects, the impactors were comprised of two or more cylinders combined. Pellet 1 was used throughout the development phase, once the initial model had been established the simulation was re–run using pellets 2 and 3. The pellets were positioned facing the centre point of the spinal cord specimen and given an initial velocity of $4.5m/s$ along the $y$ axis, causing them to approach and transversely impact the surface.

### 3.5.3 *Posterior element*

The steel backplate represents the posterior element of the spinal canal, the spinal cord construct is pressed against this by the impactor during the injury simulation. A study by Oakland found no significant difference in deformation when using a flat posterior element when compared to a more anatomically correct one [100]. The backplate was modelled using a rectangular box geometry, forming a wall on the $y-$ face of the domain, on the opposite side of the spinal cord to the impactor. The thickness of the plate was set to that of one computational cell ($0.5mm$). This constitutes a reduction in thickness from the $5mm$ seen in the experimental and FE models. This was done to reduce the overall number of particles, reducing computational time. As the backplate undergoes negligible deformation, this change was considered not to significantly affect the results.

As the backplate is effectively a wall, removing it entirely was considered. The backplate was removed and instead a Dirichlet boundary condition imposed on the $y-$ face of the computational domain (Velocity$= [0, 0, 0]$; particles that hit the face have their velocities reduced to 0). However, this produced a different behaviour to the

physical observation, particles that hit the back wall were effectively frozen in place. This caused the cord construct to stick to the back wall, rather than bouncing back from the backplate as the pellet recoiled. For this reason the backplate was retained, albeit with a reduced thickness. Interactions between other materials and the backplate are handled by the contact algorithm, and interactions between these particles and the domain boundary conditions are avoided. This allows the cord construct the collide with, slide against, and bounce back from the backplate, rather than becoming effectively adhered to the face of the computational domain due to the Dirichlet boundary.

### 3.5.4 *Dura Mater*

The dura mater was modelled as an elliptical tube, $0.5\mathrm{mm}$ thick, encircling the spinal cord and spanning the length of the computational domain from top to bottom ($z+$ to $z-$). The inner surface of the dura mater was positioned $0.1\mathrm{mm}$ from the surface of the cord in the direction of impact ($y$ axis). This reflects an observation in the experimental model; that the draining of the CSF brought the dura closer to the surface of the cord. The geometry was generated in a similar way to the cord itself (Section 3.5.1). It is possible to subtract one geometry piece from another. An inner elliptical cylinder with diameter ranging from $14\mathrm{mm}$ to $11.2\mathrm{mm}$, was subtracted from an outer elliptical cylinder, which had a diameter ranging from $14.5\mathrm{mm}$ to $11.7\mathrm{mm}$, resulting in the tube structure. The spinal cord construct, now including the dura mater, was positioned immediately in front of the backplate.

| Material | Constitutive model | Density ($kg/m^3$) |
|---|---|---|
| Cord Tissue | Ogden $\mu = 2kPa$, $\alpha = 9$, $\kappa = 45 \times 10^3$ | 1050 |
| Dura Mater | Ogden $\mu = 322kPa$, $\alpha = 19$, $\nu = 0.4$ | 1000 |
| Impactor | Linear $E = 6.5GPa$, $\nu = 0.3$ | 1360 |
| Backplate | Linear $E = 193GPa$, $\nu = 0.3$ | 8000 |

Table 10: Material models and parameters used in the MPM model.

## 3.6 CONSTITUTIVE MATERIAL MODELS

The constitutive models used for each of the solid materials, along with their parameters, are shown in Table 10. A detailed description of these models is given in Chapter 2.

### 3.6.1 *Spinal Cord*

The spinal cord tissue was modelled using a single term compressible hyperelastic Ogden model with the parameters $\mu = 2kPa$, $\alpha = 9$, $\kappa = 45 \times 10^3 kg/m^3$. The bulk modulus, $\kappa$, equates to an equivalent Poisson's ratio of $\nu = 0.4$ using Equation 104.

$$\kappa = \frac{E}{3(1-2\nu)} \tag{104}$$

where

$$E = \frac{3}{2}\mu\alpha \tag{105}$$

### 3.6.2 *Impactors*

The impactors, or simulated bone fragments, were modelled using a linear elastic constitutive model with the parameters $E = 6.5GPa$, $\nu = 0.3$. The parameters produce a material behaviour roughly equivalent to that of bone. The same constitutive model and parameters were used for all 3 impactors.

### 3.6.3 *Posterior Element*

The stainless steel backplate, representative of the posterior element of the spinal canal, was modelled using a linear elastic constitutive model the with parameters: $193GPa$, $\nu = 0.3$.

### 3.6.4 *Dura Mater*

The dura mater was initially modelled using a linear elastic model ($E = 80MPa$, $\nu = 0.49$), consistent with the FE model (Table 9). These early simulations all failed with a negative Jacobian of the deformation gradient being detected at particles in the dura mater, indicative of non-physical behaviour. It appeared to be the case that the linear mode, as implemented for MPM, used was unable to handle such a large deformation over a short timespan. Rather than continue using the linear model an Ogden model was used instead, which was better able to capture the hyperelastic behaviour of the dura mater.

The `lsqcurvefit` function in Matlab (R2021b, The MathWorks, Inc., Natick, MA, United States) was used to generate Ogden material constants producing an almost linear stress–strain response, equivalent to that of the linear model. This eliminated the issue. Having moved to using a hyperelastic model for the dura and incur-

ring the additional computational cost associated with doing so, it seemed prudent to apply some more anatomically relevant material properties. The dura mater was therefore modelled using a single term compressible hyperelastic Ogden model with material constants: $\mu = 322\text{kPa}, \alpha = 19, \nu = 0.4$ [149, 159]. These values were based on published results from mechanical testing of bovine dura mater [96].

## 3.7 RESOLUTION

The domain resolution was set to $0.5 \times 0.5 \times 0.5\text{mm}$, defining an underlying grid of $\frac{1}{8}\text{mm}^3$ cubes. The material resolution for the spinal cord and dura mater was set to $2 \times 2 \times 2$, therefore; at the start of the simulation each cell contained 8 material points for those materials. The material resolution for the impactor and backplate was set to $1 \times 1 \times 1$, each cell containing 1 material point for those materials. $2 \times 2 \times 2$ is the ideal material resolution for a 3D simulation. However, the reduction in detail had little effect on the rigid materials, whilst reducing the overall number of particles and hence reducing computational expense.

## 3.8 BOUNDARY CONDITIONS

Boundary conditions were specified on the boundaries of the domain only. Due to the nature of the MPM it is not necessary to track material boundaries or to apply explicit conditions to the boundaries between materials. Interaction between materials was handled by the `friction` contact algorithm, which is discussed in Section 3.9.

Symmetric boundary conditions were applied to exploit the symmetry of the model geometry and reduce computational expense. Symmetric boundaries were specified on the $x+$ and $z+$ faces of the

computational domain, effectively quartering the model. A Dirichlet boundary condition was specified on the $z-$ face. The velocity on the face was set to zero effectively fixing the end of the cord in place at the domain boundary. A Dirichlet boundary was also specified on the $y-$ face, fixing the backplate in place also. A Neumann boundary condition, which allows particles to advect freely past it and out of the domain, was specified on the $y+$ face. This allowed the pellet to advect freely out of the domain after recoiling, having impacted the cord. A Neumann boundary was also specified on the $x-$ face, no particles came into contact with this boundary during the simulation.

## 3.9 MATERIAL CONTACT

In MPM the material boundaries within the computational domain are not explicitly tracked. Each particle holds all of its own state variables (velocity, acceleration, temperature, etc.), the underlying computational mesh serves only as a point of reference. Contact models are therefore required to handle the interactions between materials on the grid, without these the materials would simply behave as if each were alone in its own computational domain.

The standard contact algorithm for MPM is referred to as `single velocity` within the UCF. This provides a no–slip, no–penetration model for contact between materials, and works on the assumption that all particle data communicates with a single field on the underlying grid [156]. The UCF provides an extended version of this contact algorithm referred to as `friction`, based on the work of Bardenhagen et al. [160]. The `single velocity` contact model is limited in that particles will be effectively stuck together, forming a single body, if they are initially positioned next to one another. The `friction` contact model allows frictional sliding, based on the coefficient of friction

(specified as an input) and the normal force, it overcomes this issue by adding an additional check to allow free separation of materials. The check applies to whether a particle is approaching or departing a given grid node. If it is approaching then contact is applied. If it is departing then a secondary check is performed to find whether the particle is in tension or compression. If it is in compression then the particle is still rebounding and contact is applied, if it is in tension then the particle is allowed to move away. This contact algorithm (`friction`) was selected for the SCI model and the coefficient of friction set to zero, effectively allowing free-slip between all materials. There does not seem to be any data in the literature regarding the coefficient between spinal cord or meningeal tissues and bone. In the FE model a coefficient of 0.1 was used, however the application of this (versus zero) was found to have very little effect on the overall deformation of the spinal cord construct ($< 0.1\%$) [73]. It was therefore assumed that this simplification was acceptable and would have little effect on the results of the MPM model.

## 3.10 TIME INTEGRATION SCHEME

An explicit time integration scheme, available with the UCF implementation of MPM, was used [128, 147]. This is the standard variant of MPM, an implicit version of MPM is also available, in which the mesh is not reset at the end of each timestep [124]. This variant was developed for low rate of strain problems to avoid the restrictions stemming from the use of an explicit time integration scheme [124]. Implicit MPM was not selected for this SCI model as it may have given rise to mesh tangling issues (traumatic SCI is a high rate of strain problem), which was one of the key drivers behind using MPM

rather than FEA. This issue is described in detail in Chapter 1 Section 1.7.1.

## 3.11 PARALLELISATION

The UCF software was built and installed for the ARC2 High Performance Computing facility at the University of Leeds. The system comprises of 190 HP BL460 blade servers, each housing one node. Each node was a dual socketed with 8–core Intel E5-2670(2.6GHz) processors. There were 16 cores per node; 380 CPUs delivering a total of 3040 cores. The computational domain for the solid model was divided into 320 rectangular patches, the patch layout remaining static throughout the simulation. Each patch was computed on a single core, with 320 cores being utilised in total. The patch layout was specified at $8 \times 1 \times 160$ in the $x$, $y$, and $z$ directions respectively. This gave a reasonably even distribution of particles per patch, however; due to the geometry of the model and the static nature of the patches, some cores were inevitably underutilised.

Dynamic Load Balancing (DLB) is a scheme that attempts to optimise the distribution of the computational load to improve efficiency and maximise throughput. In the static scheme described above each patch, containing a portion of the computational workload, is assigned to a single processor, a computational resource. Due to the model geometry, the workload is not distributed evenly across the patches; some patches contain more particles than others. This results in an underutilisation of the computational resource as each processor must wait until all the calculations for a given timestep for all patches are complete before moving on to the next timestep. Progress is limited by the time taken to resolve the most computationally expensive (i.e. slowest) patch, processors need to remain idle while they

wait for the completion of the slowest patch. This inefficiency can be mitigated by ensuring that each patch incurs roughly the same computational expense meaning that each processor will complete its tasks at around the same time, thus reducing the time processors spend idling. However, even if the distribution of the workload is even at the start of the simulation it is likely to change as it progresses and the materials move and deform. DLB addresses this by dynamically reassessing the distribution of the workload across the processors continuously throughout the simulation in order to maintain and even distribution. This can be done by rearranging the layout of the patches aiming to keep the computational cost for each as close to even as is possible.

A DLB scheme exists within the UCF, but this was not suitable for the SCI model and offered no performance gains. The DLB scheme presently available within the UCF does not alter the patch layout, which remains fixed throughout. Instead multiple patches are assigned to a single processor with the aim of maintaining an even workload. The uneven distribution of particles over the computational domain combined with the additional overheads incurred by single processors handling multiple patches (rather than the one patch per processor layout) meant that this approach was unsuitable for improving performance in the SCI model. A superior DLB approach would be to recompute the patch layout to maintain an even particle distribution whilst retaining the one patch per processor distribution, however; this approach is not presently implemented within the UCF. This is a limitation of the software and there is no reason that such a scheme could not be implemented for MPM.

## 3.12 VALIDATION

Figures 17-19 show the deformation over time for the bare cord model for each of the three impactors (P1-P3), comparing the MPM, FE, and Experimental (mean average of 16 bovine specimens) models. Figures 20-22 show the same for the cord/dura model. Tables 11 and 12 show the maximum deformation (MD) and time to maximum deformation (TTMD). The FE results each show a dip, more visible for the cord/-dura models (e.g. Figure 20 at 1.20ms). This is due to a small gap between the cord and backplate, not present in the MPM model and too slight to record in the experiments. At a certain point (e.g. Figure 20 at 1.40ms) the inertia is overcome, the construct then begins to move backwards.

Figures 17-19 show results for the bare cord models, both of the computational results show a close agreement in terms of both MD and TTMD. For all three impactor shapes the MD is very similar, with the MPM model being slightly lower than FE (by 0.18, 0.17, and 0.36mm for pellets 1–3 respectively), meaning that the MPM is slightly closer to the experimental compared against FE. In the cases of P1 and P2 the MPM result overestimates the maximum deformation (by 0.87mm and 0.65mm respectively), which is consistent with the FE result. Both fall above the experimental standard deviation. For P3 the MPM model underestimates the MD by 0.58 mm, again this is consistent with the FE model, both fall within the experimental standard deviation. Similarly, the TTMD for both computational results also show close agreement. TTMD for the MPM model was slightly lower compared to FE (by 0.1, 0.1, and 0.15 ms for pellets 1–3 respectively), this reflects the lower MD compared to FE. Similarly to the FE model, the MPM model significantly underestimates TTMD compared to the experimental mean (by 0.85, 1.56, and 2.34 ms for

pellets 1–3 respectively). This may be due to increased tissue compliance due to degradation and repeated testing, this is discussed in Chapter 6 Section 6.4.2.

In the case of the cord/dura models using P1, all of the models show a close agreement in terms of MD. In the case of P2 the MPM result overestimates the maximum deformation (by 0.44 mm) whilst the FE result is closer to the experimental mean. In the case of P3 the MPM MD is very close to the experimental mean, while the FE model underestimates the maximum deformation (by 0.91 mm). In all three cases the MD for both computational models fell within the experimental standard deviation. As with the bare cord MPM model, the cord/dura MPM model underestimates TTMD compared to the experimental mean (by 0.58, 0.83, and 0.85ms for pellets 1–3 respectively), albeit not to the same extent. Overall the cord/dura MPM model shows a closer agreement with the experimental model than the bare cord MPM model does. The same is true of the FE model, suggesting that this difference is not due to the use of MPM as both computational results were very similar.

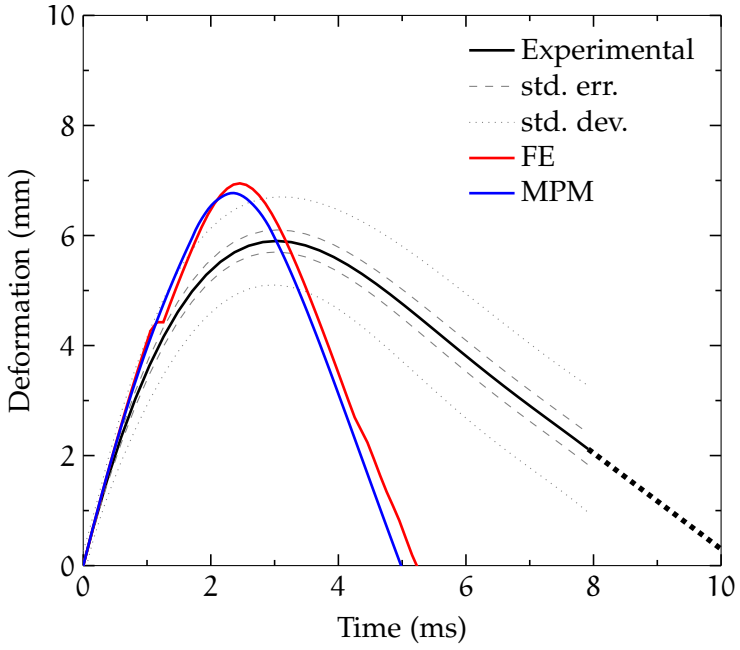



Figure 17: Pellet 1, bare cord model

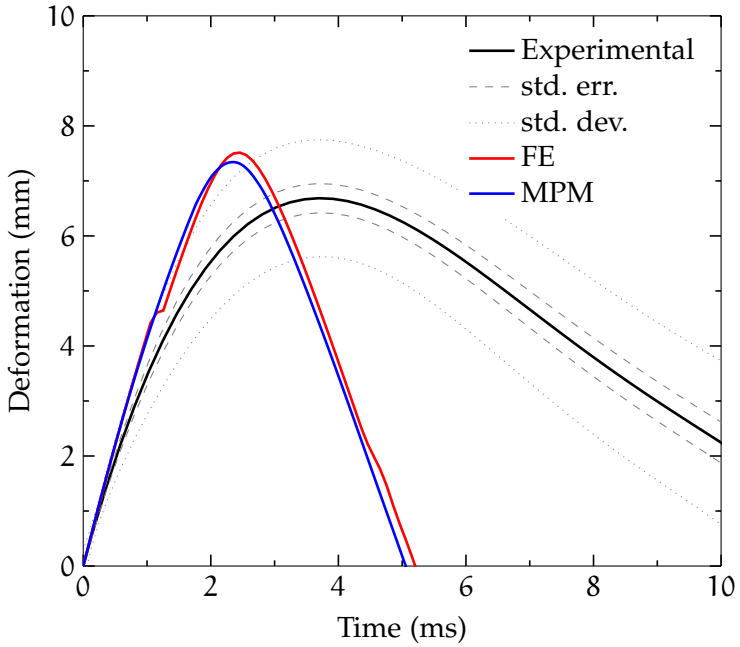



Figure 18: Pellet2, bare cord model

## 3.13 CONCLUSIONS

A burst fracture spinal cord injury model was created using the Material Point Method (MPM) implementation within the Uintah Computational Framework, this solid material SCI model served as the base for the full FSI model using MPMPICE. The model consisted of a simulated bone fragment transversely impacting the surface of the spinal cord in two configurations; with and without the dura mater (both excluding the CSF). Three impactors with different geometries were tested.

The MPM model was validated against existing FE and experimental results by comparing the deformation over time curves, maximum deformation, and time to maximum deformation. For the bare cord configuration the MPM results were extremely close to the equivalent FE results. Both the FE and MPM models overestimated the maximum deformation when compared to the experimental results, this

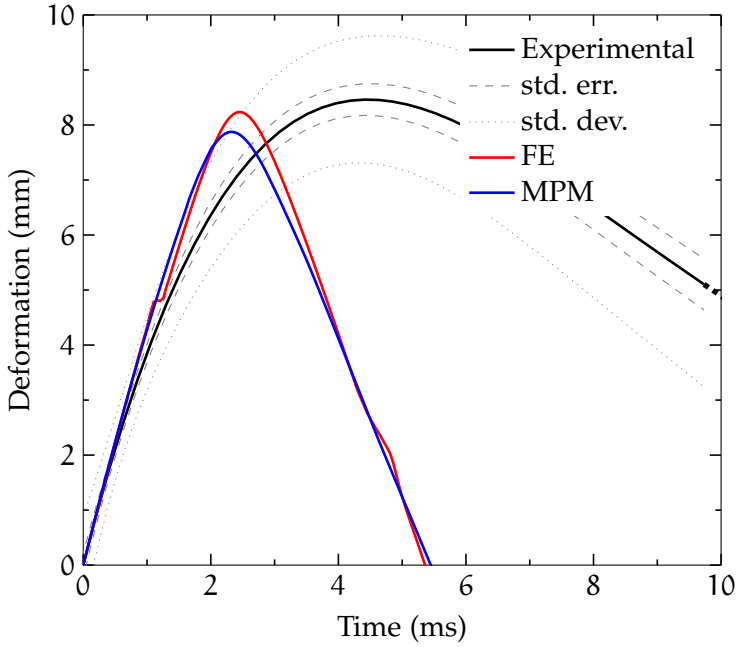



Figure 19: Pellet 3, bare cord model

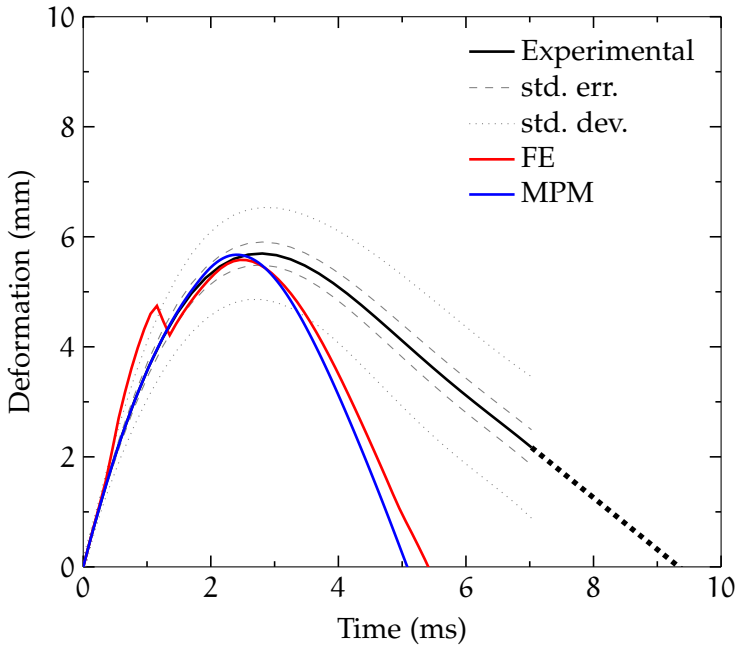



Figure 20: Pellet 1, cord/dura model

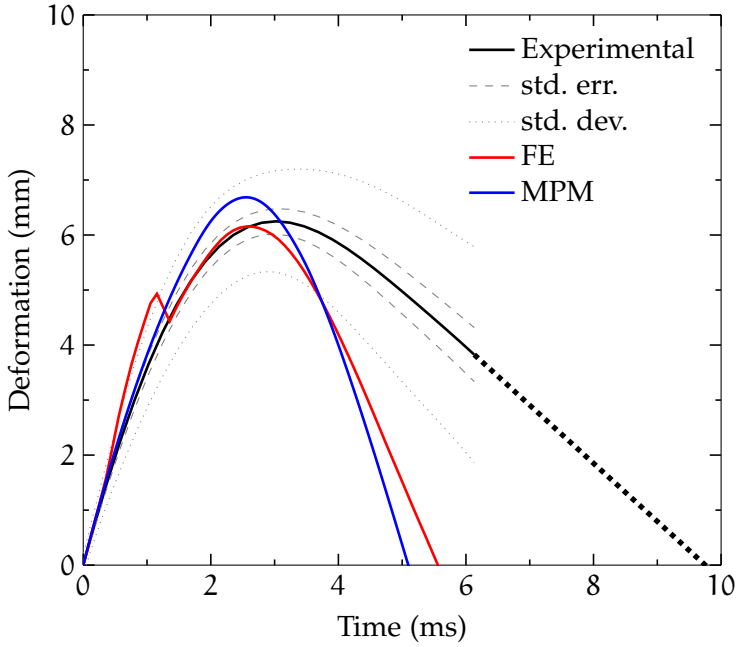



Figure 21: Pellet2, cord/dura model

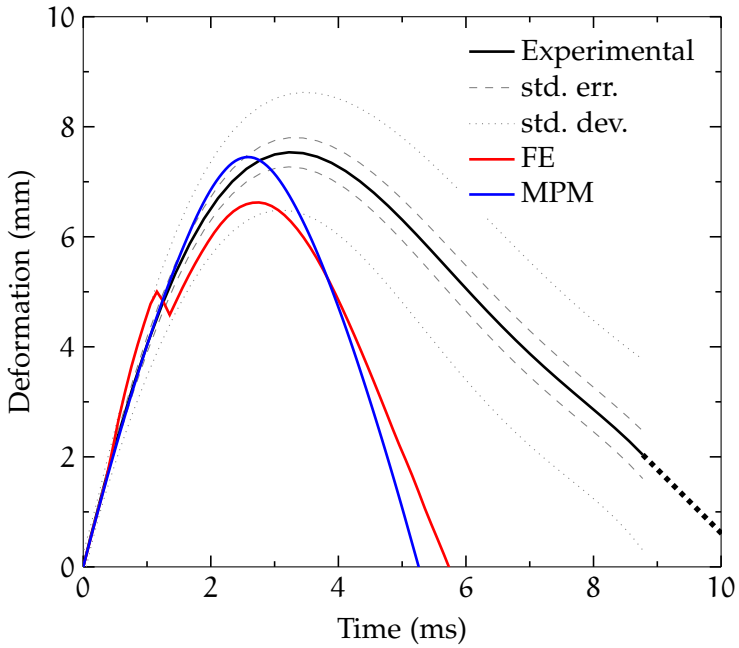



Figure 22: Pellet 3, cord/dura model

| | | MD (mm) | TTMD (ms) |
|---|---|---|---|
| P1 | Exp. | 5.90 (±0.20) | 3.21 (±0.12) |
| | FE | 6.95 | 2.46 |
| | MPM | 6.77 | 2.36 |
| P2 | Exp. | 6.69 (±0.27) | 3.92 (±0.18) |
| | FE | 7.51 | 2.46 |
| | MPM | 7.34 | 2.36 |
| P3 | Exp. | 8.46 (±0.29) | 4.65 (±0.20) |
| | FE | 8.24 | 2.46 |
| | MPM | 7.88 | 2.31 |

Table 11: Maximum Deformation (MD) and Time to Maximum Deformation (TTMD) for the 3 pellet types (P1-P3) in the experimental, FE, and MPM models, bare cord excluding the dura mater.

| | | MD (mm) | TTMD (ms) |
|---|---|---|---|
| P1 | Exp. | 5.70 (±0.21) | 2.97 (±0.11) |
| | FE | 5.58 | 2.46 |
| | MPM | 5.67 | 2.39 |
| P2 | Exp. | 6.25 (±0.23) | 3.36 (±0.23) |
| | FE | 6.15 | 2.56 |
| | MPM | 6.69 | 2.53 |
| P3 | Exp. | 7.53 (±0.27) | 3.43 (±0.13) |
| | FE | 6.62 | 2.76 |
| | MPM | 7.45 | 2.58 |

Table 12: Maximum Deformation (MD) and Time to Maximum Deformation (TTMD) for the 3 pellet types (P1-P3) in the experimental, FE, and MPM models, including the cord and dura mater.

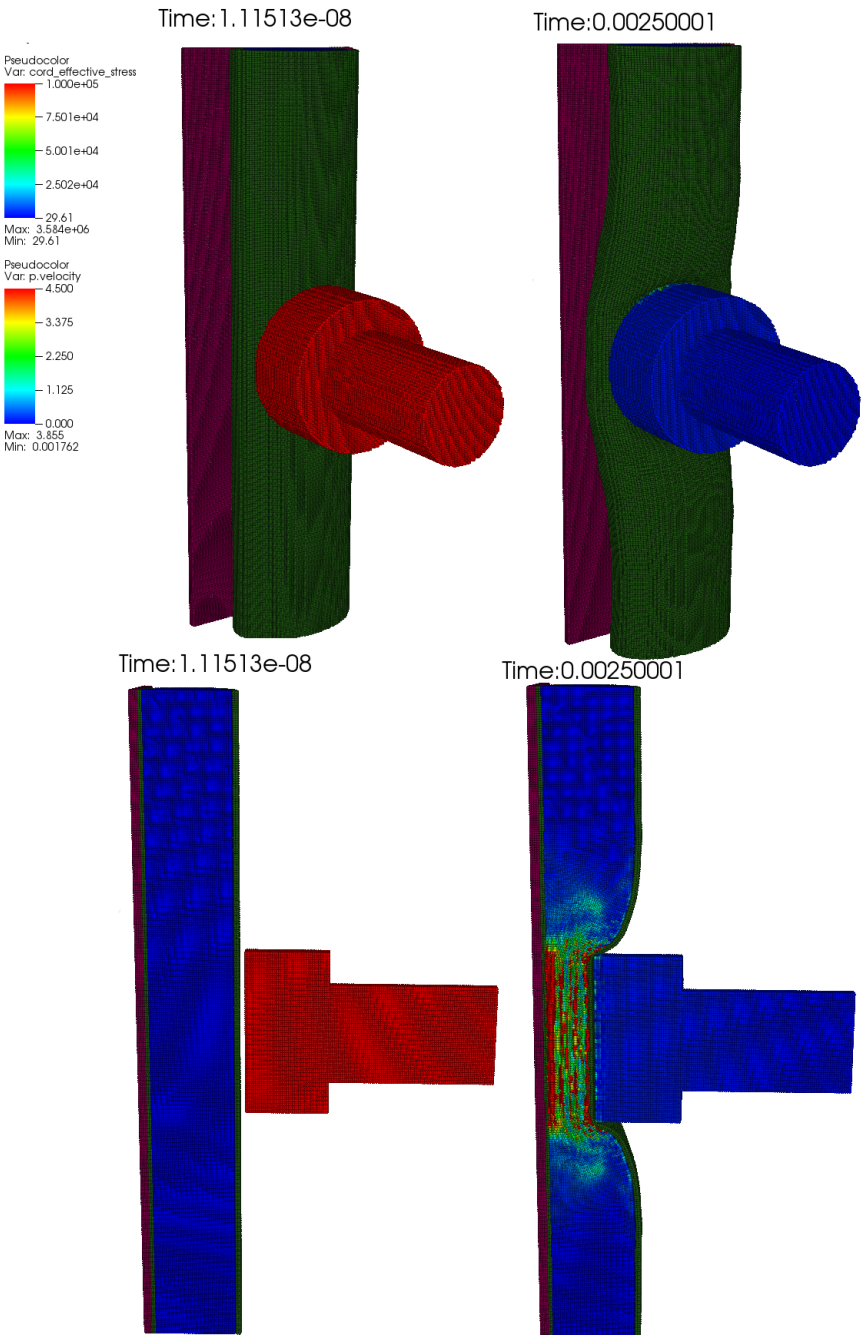



Figure 23: Visualisation of the SCI model including the cord and dura mater. The impactor is coloured by velocity, red indicates high velocity (maximum $4.5\mathrm{m/s}$), blue indicates low velocity (minimum $0\mathrm{m/s}$). The cord tissue is coloured by equivalent stress, red indicates high stress (maximum $1 \times 10^5\mathrm{Pa}$), blue indicates low stress (minimum $0\mathrm{Pa}$). Time is shown in seconds.

may have been caused increased compliance of the tissue used in the experiments due to repeated impacts. The fact that both computational results are very similar suggests that this disparity is due to the experimental methodology and not an issue inherent to MPM or FE. In the case of the cord/dura configurations the MPM, FE, and experimental results were all in close agreement and the overestimation observed for the bare cord model did not occur.

MPM is more computationally expensive per timestep than FE due to the additional steps required in interpolating between the particles and the computational grid nodes. However, MPM offers greatly superior parallel scaling compared to FE, with up to 320 parallel processors being utilised for the solid SCI model. MPM is also less accurate than FE (first order versus second order accuracy) and offers less clearly defined material boundaries as a result of the GIMP interpolation scheme. This can be mitigated by increasing the resolution of the underlying material grid, which reduces the error but also increases the number of particles and hence the computational expense. This increase may in turn be overcome by utilising additional parallel processors to reduce computation time.

# 4

# FLUID MODEL DEVELOPMENT

## 4.1 OVERVIEW

Having created a solid material Spinal Cord Injury (SCI) model using the Material Point Method (MPM), including the spinal cord and the dura mater, the next stage in the development of the model was to incorporate the cerebrospinal fluid (CSF). Once again, the aim was to create an computational Fluid–Structure interaction (FSI) model comparable to the existing experimental model, using the Material Point Method (MPM) instead of of the traditional approach using Finite Element Analysis (FEA). Two attempts were made to incorporate the CSF using MPM only (Section 4.3), using a Mooney–Rivlin fluid approximation for the fluid and using a water constitutive model. Neither of these was successful, which was unsurprising, MPM alone was not suited to modelling both fluid and solid phases.

The FSI model, including the cord, dura mater, and CSF, was achieved using MPMICE (Material Point Method Implicit Continuous Fluid Eulerian), this is a multi–material algorithm that utilises MPM for solid materials but also integrates Eulerian fluid dynamics using the same underlying computational grid. It is capable of simulating strong interactions between solid and fluid phases and, like MPM, an implementation is available within the Uintah Computational Framework (UCF). This method and its relationship with MPM is described in detail in Section 4.2. Solid material stresses are still calculated using the constitutive material models, fluid behaviour is governed by equations of state (EOS), described in Section 4.5. Existing equations

of state within the UCF were used for the fluid phases, the air was modelled using the ideal gas law (Section 4.5.2), and the CSF using a water model referred to as `THWater` (Section 4.5.1). This utilises a Gibbs free energy function to determine water fluid behaviour.

A constraint of MPMICE is that there cannot be empty space within the computational domain, hence the empty space in the MPM model was filled with air (Section 4.4.1), the motion of which was simulated but is of no interest. In turn the boundary conditions on the faces of the domain had to be adjusted, Neumann boundaries were changed to Dirichlet boundaries (Section 4.6), this prevented the surrounding air from escaping the domain, which would soon result in a failed simulation. As with MPM, no boundary conditions need be specified between materials as the are handled by the contact algorithm. In addition, MPMICE requires momentum and heat exchange coefficients to be specified between each material to determine the rates at which these quantities are transferred between materials 4.10.

A constraint of the `THWater` EOS was that it would fail if the pressure in the fluid fell below 1 atmosphere in any area. To work around this the pressure in the domain was increased and the densities of the materials adjusted accordingly (Section 4.7). Initially the pressure was increased to 20 atmospheres and once a working model had been created this was reduced to 3 atmospheres, which was determined to be the lowest pressure achievable whilst avoiding the constraint imposed by the `THWater` EOS. This situation is sub–optimal, however; implementation of an alternative approach was not possible within a feasible time–frame, and the model produced accurate fluid behaviour so long as the condition was met.

The completed FSI SCI model using MPMICE, including the cord, dura, and CSF was validated against the existing experimental model and FE computational model. A parametric study was then performed

to determine the sensitivity of the model to the thickness of the CSF layer, this is discussed in Section 4.13.

## 4.2 MPMICE

Material Point Method Implicit Continuous Fluid Eulerian (MPMICE) is a multi-material algorithm initially developed at Los Alamos National Laboratory (NM, USA) for simulating explosions of energetic devices, where there was a need for an approach capable of handling very large material deformations over a short time involving both solid and fluid phases [134]. This is a full physics formulation, i.e. incorporating strong interactions between solid and fluid phases in terms of both temperature and velocity. As the name suggests, MPMICE incorporates MPM and a CFD approach named ICE (Implicit Continuous Eulerian) for the fluid dynamics. ICE is a CFD formulation with full Navier-Stokes representation of fluids using an Eulerian grid, with the capability for including chemical and physical transformations between phases, e.g. fuel combustion.

It is essential for the solid material deformation history to be transported through the Eulerian grid, MPMICE achieves this through the incorporation of MPM, as the Lagrangian computational grid in MPM is effectively reset at the end of each timestep. MPM is considered to be an Arbitrary Lagrangian Eulerian (ALE) technique; it is possible to use the same underlying grid for MPM and to serve as the Eulerian grid for the ICE component. FE based FSI may utilise either a monolithic or partitioned approach, monolithic FSI models use a single stiffness matrix incorporating both phases with a single solver. Partitioned FSI models use separate solvers for the different phase, which are coupled with a interface algorithm (e.g. the Immersed Boundary Method) to give the overall solution. The FE model

using ADINA (ADINA R&D Inc., Watertown, MA), against which the MPMICE model was validated, used a monolithic approach.

MPMICE is an averaged FSI solution, allowing any material to exist at any point in space at any time. There is no need to track material interfaces as this is handled inherently by the MPMICE method. An implementation of MPMICE using an explicit time integration scheme is available in the UCF. Section 4.2.1 describes the underlying theory, Section 4.2.2 describes the integration of this approach with MPM, which was discussed previously in Chapter 3 Section 3.3).

### 4.2.1 *Theory*

The material derivative, $\frac{D\mathbf{F}}{Dt}$, describes the rate of change of a material element in response to a time dependent macroscopic velocity field. In the Lagrangian frame this is simply given by

$$\frac{D\mathbf{u}}{Dt} = \frac{\partial \mathbf{u}}{\partial t} \tag{106}$$

where $\mathbf{u}$ is the velocity gradient and $t$ is time. In the Eulerian frame the material derivative is given by

$$\frac{D\mathbf{u}}{Dt} = \frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} \tag{107}$$

whereas for Arbitrary Lagrangian Eulerian

$$\frac{D\mathbf{u}}{Dt} = \frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} - \dot{\mathbf{x}}) \cdot \nabla \mathbf{u} \tag{108}$$

note that if $(\mathbf{u} - \dot{\mathbf{x}}) \cdot \nabla \mathbf{u}$ is equal to zero then the frame is fully Lagrangian and Equation 106 is recovered. For an Eulerian grid $\dot{\mathbf{x}} = 0$,

for a Lagrangian grid $\dot{\mathbf{x}} = \mathbf{u}_r$, and in the case of ALE $\dot{\mathbf{x}} \neq \mathbf{u}_r$. MPMICE is an ALE method.

Consider a arbitrary finite volume of space $V(x,t)$ containing N arbitrary materials, denoted by subscript $r = 1, 2...N$. The material state vector $[M_r, \mathbf{u}_r, e_r, T_r, \upsilon_r, \theta_r, \boldsymbol{\sigma}_r, p]$ gives the averaged physical state for a given material. M is the material mass, $\mathbf{u}$ is the velocity, $e$ is the energy, T is the temperature, $\upsilon$ is the specific volume, $\theta$ is the volume fraction, $\boldsymbol{\sigma}$ is stress, and p is the equilibration pressure. The averaged density for a given material is $\rho = M_r/V$. Equations 109-111 are the averaged model equations for mass, momentum, and energy, which give the rate of change in the state in a volume moving with the velocity of the given material. In the following equations $\boldsymbol{\sigma}$ is taken to be the isotropic mean mixture stress (mixture of solid and fluid phases), such that $\boldsymbol{\sigma} = -p\mathbf{1}$ in terms of the hydrodynamic pressure, where $\mathbf{1}$ is the identity matrix.

$$\frac{1}{V}\frac{D_r M_r}{Dt} = \sum_{s=1}^{N} \Gamma_{rs} \tag{109}$$

$$\frac{1}{V}\frac{D_r(M_r\mathbf{u}_r)}{Dt} = \theta_r \nabla\cdot\boldsymbol{\sigma} + \nabla\cdot\theta_r(\boldsymbol{\sigma}_r - \boldsymbol{\sigma}) + p_r\mathbf{g} + \sum_{s=1}^{N}\mathbf{f}_{rs} + \sum_{s=1}^{N}\mathbf{u}_{rs}^{+}\Gamma_{rs} \tag{110}$$

In Equation 110, $\sum_{s=1}^{N}\mathbf{f}_{rs}$ signifies the model for momentum exchange between materials, a function of the relative velocity between

materials at the given point. This is derived from the deviation of the material specific field stress from the mean stress.

$$\frac{1}{V}\frac{D_r(M_r e_r)}{Dt} = \rho_r \rho \frac{Dr v_r}{Dt} + \theta_r \boldsymbol{\tau}_r : \nabla \mathbf{u}_r - \nabla \cdot \mathbf{j}_r + \sum_{s=1}^{N} q_{rs} + \sum_{s=1}^{N} h_{rs}^{+} \Gamma_{rs} \tag{111}$$

Here $\boldsymbol{\tau}_r$ is the viscous stress tensor, where $\boldsymbol{\sigma}_r = -p\mathbf{1} + \boldsymbol{\tau}_r$, the term $\theta_r \boldsymbol{\tau}_r : \nabla \mathbf{u}_r$ is the viscous heating component. Similarly to Equation 110, in Equation 111, $\sum_{s=1}^{N} q_{rs}$ signifies a model for heat exchange between materials. Heat flux is denoted by $\mathbf{j}_r$, according to Fick's first law

$$\mathbf{j}_r = -p_r b_r \nabla T_r \tag{112}$$

where $b_r$ is the thermal diffusion coefficient, which includes molecular and turbulent effects. The term $\sum_{s=1}^{N} h_{rs}^{+} \Gamma_{rs}$ is the rate of mass conversion from material $s$ into material $r$, due to phase change or chemical reaction, such as the burning of a solid reactant into gaseous products or the evaporation of a liquid into a gas. In Equations 110-111, $\mathbf{u}_{rs}^{+}$ is the velocity of the reactant material $s$ undergoing transformation into resultant material $r$, while $h_{rs}^{+}$ is the enthalpy. A solid reaction model governs the rate at which this occurs, as there is no combustion involved in the SCI model (no reaction model was included) this feature will not be described in detail here. The material masses (Equation 109) will remain constant throughout the SCI simulations.

Temperature $T_r$, specific volume $v_r$, volume fraction $\theta_r$, and equilibration pressure $p$ relate to material mass and material mass density, $\rho_r$, via equations of state.

$$e_r = e_r(v_r, T_r) \tag{113}$$

$$v_r = v_r(p, T_r) \tag{114}$$

Equations 113 and 114 are the caloric equation of state, which specifies the dependence of the energy, $e_r$, on the specific volume, $v_r$ and temperature, $T_r$, and the thermal equation of state, which specifies the volumetric reaction of the material to changes in temperature.

$$\theta_r = p v_r \tag{115}$$

$$0 = 1 - \sum_{s=1}^{N} \rho_s v_s \tag{116}$$

Equation 115, which is the volume fraction, is defined as the volume of a given material (labelled by subscript $r$) divided by the total volume of all the materials. As such, Equation 116 is the multi material equation of state, this defines the values of hydrodynamic pressure. This is required to calculate the equilibrium pressure and to calculate the specific volumes to allow any arbitrary mass to fill the the volume, $V$, identically. This pressure is referred to as equilibration pressure.

A closure relation is required to calculate the material stress. For solid materials Cauchy stress is used, this is computed by the relevant

solid constitutive material model, in the case of the SCI model an Ogden constitutive model is used, this is discussed in Chapter 2. In the case of fluids

$$\boldsymbol{\sigma}_r = -p\mathbf{1} + \boldsymbol{\tau}_r \tag{117}$$

Having obtained equations for all of the elements of the state vector it is now possible to calculate these for any volume of space moving with any material velocity. The frame of reference used is that best suited to the individual material. Lagrangian in the case of the solid materials, achieved using MPM, and Eulerian for fluid materials. It is possible that the volumes will remain consistent as different materials can be defined in different frames of reference. It is therefore necessary to treat the specific volume as a dynamic variable of the material state and to integrate it forward in time with each timestep as the simulation advances

$$V_t = \sum_{r=1}^{N} M_r v_r \tag{118}$$

subsequently, the volume fraction for a material is given by

$$\theta_r = M_r v_r / V_t \tag{119}$$

The sum of the volume fractions for all of the materials being equal to 1. i.e. $\sum_{r=1}^{N} \theta_r = 1$. An evolution equation for the specific volumes,

which defines how the specific volumes of the materials evolve as time advances, was developed by Kashiwa [161]

$$\begin{aligned}\frac{1}{V}\frac{D_r(M_r v_r)}{Dt} = f_r^\theta \nabla \cdot \mathbf{u} + \left[v_r \Gamma_r - f_r^\theta \sum_{s=1}^{N} v_s \Gamma_s\right] \\ + \left[\theta_r \beta_r \frac{D_r T_r}{Dt} - f_r^\theta \sum_{s=1}^{N} \theta_s \beta_s \frac{D_s T_s}{Dt}\right]\end{aligned} \tag{120}$$

where $\beta$ is the constant pressure thermal expansivity, $\kappa$ is the bulk modulus, and

$$f_r^\theta = \frac{\theta_r \kappa_r}{\sum_{s=1}^{N} \theta_s \kappa_s} \tag{121}$$

### 4.2.2 *Integration with MPM*

The following description shows the advancement of a single MPMICE timestep, from time $t$ to time $t + \Delta t$, and is based on the description given by Guilkey et al. [134]. Figure 24 shows a flow chart of the algorithm.

Step 1, projection of the particle states to to the underlying computational grid, the implementation described here uses a regular cartesian grid, $N = 8, w_{ij} = \frac{1}{8}$. Particle state data is projected onto the grid nodes. In the case of velocity, this is given by

$$\mathbf{u}_j = \frac{\sum_{i=1}^{N} w_{ij} m_i \mathbf{u}_i}{\sum_{i=1}^{N} w_{ij} m_i} \tag{122}$$

where $m_i$ is the mass on the grid nodes.

Step 2, computation of equilibration pressure. As mentioned previously in Section 4.2.1 The volume of a material occupying a cell is

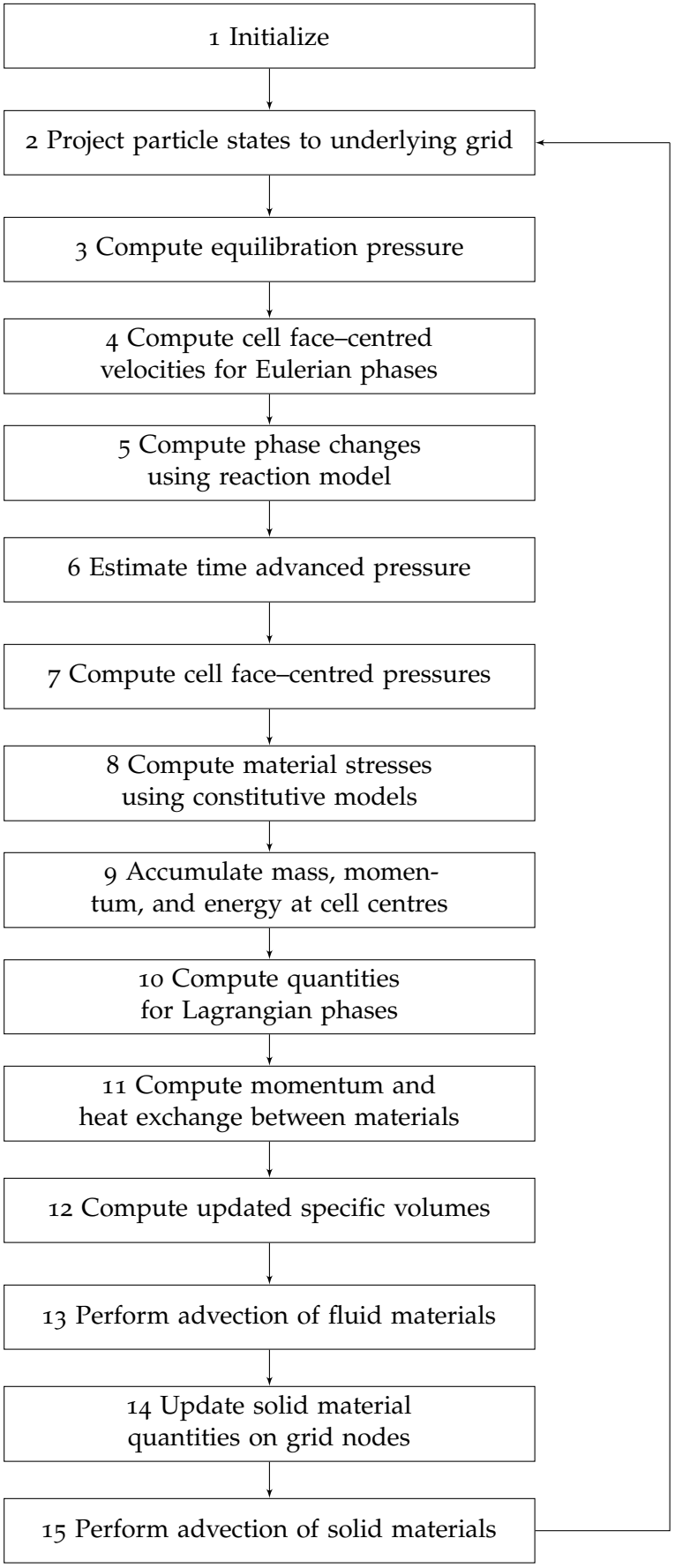



Figure 24: Evolution of a single timestep using the MPMICE algorithm.

not necessarily equal to the volume of the cell as different phases are computed using different frames of reference. As such, it is necessary to track the evolution of material volume using Equation 120. Having obtained the material masses and specific volumes it is possible to compute the material volumes $V_r = M_r v_r$, which are then summed to obtain the total volume across all materials. With this the volume fractions $\theta_r$ are calculated. The multi-material equation of state (Equation 116) can now be solved at each cell using the Newton-Raphson method [162], resulting in updated values for equilibrium pressure $p_{eq}$, volume fraction $\theta_r$, and specific volume $v_r$.

Step 3, computation of face centred velocities for Eulerian advection. The face centred velocities $\mathbf{u}_r^*$ are calculated based on adjacent values using the Equation 123 below. This uses a time advanced estimate for the cell centred velocities using the method developed by Kashiwa et al. [161, 163].

$$\mathbf{u}_r^* = \frac{\rho_{r_L}\mathbf{u}_{r_L} + \rho_{r_R}\mathbf{u}_{r_R}}{\rho_{r_L} + \rho_{r_R}} - \left(\frac{2v_{r_L}v_{r_R}\Delta t}{v_{r_L} + v_{r_R}}\right)\left(\frac{p_{eq_R} - p_{eq_L}}{\Delta x}\right) + g\Delta t \quad (123)$$

where $g\Delta t$ is the acceleration due to gravity.

The first term of Equation 123 is a mass weighted average of the left and right cell-centred velocities, denoted by subscripts L and R. The second term is a pressure accelerated gradient. The third term is the acceleration in relation to gravity in the face normal direction. Momentum exchange also occurs at the cell face centres, handled using the same methodology as described later in Step 10.

Step 4, multiphase chemistry. A solid reaction model is used to calculate sources of Mass, $\Gamma_r$, momentum, $\mathbf{u}_r\Gamma_r$, internal energy, $e_r\Gamma_r$, and specific volume, $v_r\Gamma_r$, for each material. The way in which this occurs, which determines the calculation of $\Gamma_r$, is model dependent.

Mass, momentum, and energy are conserved according to Newton's laws.

Step 5, Computation of an estimated time advanced pressure. An estimate of time advanced pressure p is computed based on the volume of material added to or subtracted from a cell. The pressure increment is calculated using

$$\Delta p = \Delta t \frac{\sum_{r=1}^{N} v\Gamma_r - \sum_{r=1}^{N} \nabla \cdot (\theta_r^* u_r^*)}{\sum_{r=1}^{N} \theta_r \kappa_r} \tag{124}$$

where the first term in the numerator signifies the volume change during a reaction. The second term represents the net change in volume of materials moving in to and out of a cell. The denominator is effectively a mean average of the compressibility of the materials occupying the cell. Having found the incremental pressure the estimated time advanced pressure is calculated using

$$p = p_{eq} + \Delta p \tag{125}$$

where $p_{eq}$ is the equilibration pressure which was calculated in Step 2 [163].

Step 6, computation of face centred pressures. The face centred pressure $p^*$ is calculated using the updated pressure obtained in the previous step

$$p^* = \frac{\left(\frac{p_L}{\rho_L} + \frac{p_R}{\rho_R}\right)}{\left(\frac{1}{\rho_L} + \frac{1}{\rho_R}\right)} \tag{126}$$

where L and R denote left or right centred values and $\rho$ is the sum density of all the materials in the cell.

Step 7, Computation of material stresses. For solid materials, the particle stresses are calculated by a constitutive material model, described in detail in Chapter 2. For fluid materials, stresses are calculated on the cell faces using the cell centred velocities.

Step 8, Accumulation of mass, momentum, and energy sources at the cell centres. Mass is accumulated using,

$$\Delta(m)_r = \Delta t V \sum_{s=1, s\neq r}^{N} \Gamma_s \tag{127}$$

where $V$ is the total volume. Momentum is accumulated using

$$\Delta(m\mathbf{u})_r = -\Delta t V \left[ \theta_r \nabla p^* + \nabla \cdot \theta_r(\boldsymbol{\sigma}_r - \boldsymbol{\sigma}) + \sum_{s=1, s\neq r}^{N} \mathbf{u}_s \Gamma_s \right] \tag{128}$$

and energy is accumulated using

$$\Delta(me)_r = -\Delta t V \left[ f_r^{\theta} p + \sum_{s=1}^{N} \nabla \cdot (\theta_r^* \mathbf{u}_r^*) + \sum_{s=1, s\neq r}^{N} e_s \Gamma_s \right] \tag{129}$$

Only the energy resulting from flow is included here, it is also possible to include other terms such as heat conduction.

Step 9, Computation of Lagrangian phase quantities. Using the increments calculated during the previous step the Lagrangian phase quantities are updated by adding the increment to the value from the previous timestep, denoted by superscript $t$. Here the superscript $L$ indicates that all physical processes for this parameter have been accounted for in the Lagrangian frame for the current timestep, i.e. on the deformable mesh that is reset at the end of each timestep. The superscript $L-$ indicates that all physical processes have been accounted

for with the exception of the transfer of heat and momentum between materials, which is dealt with in Step 10. Mass is updated using

$$(m)_r^L = (m)_r^t + \Delta(m)_r \tag{130}$$

momentum is updated using

$$(m\mathbf{u})_r^{L-} = (m\mathbf{u})_r^t + \Delta(m\mathbf{u})_r \tag{131}$$

and energy is updated using

$$(me)_r^{L-} = (me)_r^t + \Delta(me)_r \tag{132}$$

The superscript L indicates that all physical processes for these quantities have been accounted for in the Lagrangian frame, with the exception of material momentum and heat exchange.

Step 10, Momentum and heat exchange. The following equations are evaluated in an implicit pointwise manner allowing the arbitrarily large transfer of momentum between materials,

$$(m\mathbf{u})_r^L = (m\mathbf{u})_r^{L-} + \Delta t m_r \sum_{s=1}^{N} \theta_r \theta_s K_{rs} (\mathbf{u}_s^L - \mathbf{u}_r^L) \tag{133}$$

$$(me)_r^L = (me)_r^{L-} + \Delta t m_r c_{v_r} \sum_{s=1}^{N} \theta_r \theta_s H_{rs} (T_s^L - T_r^L) \tag{134}$$

where $K_{rs}$ is the momentum exchange tensor coefficient and $H_{rs}$ is the heat exchange tensor coefficient, both of which are second order

tensors. These are discussed further in Section 4.10. This operation is also performed at the end of Step 3 to compute the face centred velocities.

Step 11, Evolution of specific volume. As mentioned in Step 2, it is necessary to track the evolution of specific volume for each material in order to compute the volume fractions and equilibrium pressures. Taking into account volume changes due to changes in temperature, pressure, and phase conversion, the specific volumes are computed using

$$\Delta(m v)_r = \Delta t V \left[ v_r \Gamma_r + f_r^\theta \nabla \cdot \sum_{s=1}^{N} \theta_s^* \mathbf{u}_s^* + \theta_r \beta_r \bar{T}_r - f_r^\theta \sum_{s=1}^{N} \theta_s \beta_s \dot{T}_s \right] \quad (135)$$

$$(m v)_r^L = (m v)_r^t + \Delta(m v)_r \quad (136)$$

where $\beta$ is constant pressure thermal expansivity and $\dot{T}_r$ is the rate of temperature change for each material calculated in the Lagrangian frame.

$$\dot{T} = \frac{T^L - T^t}{\Delta t} \quad (137)$$

where $T^L$ is the updated temperature for the current timestep and $T^t$ is the temperature at the previous timestep.

Step 12, Advection of fluid materials. For the fluid phases, mass, momentum, energy, and specific volume are transported. The specific volume is converted to material volume for the purpose of advection and is then reconstituted into specific volume for use in the subsequent timestep.

Step 13, Update of solid material nodal quantities. Changes in mass, momentum, energy, and specific volume were previously computed at the cell centres, in this step these are interpolated to the grid nodes as field quantities. Changes in momentum can now be expressed as accelerations on the grid nodes.

Step 14, Solid materials advection. The time advanced grid velocity and acceleration is now interpolated back to the material points using shape functions and the material point velocities and positions are then updated, as discussed in Chapter 3 Section 3.3.1. This concludes the execution of a single timestep using MPMICE.

## 4.3 ADDING THE CSF

### 4.3.1 *Mooney-Rivlin fluid approximation*

The FE model created by Maikos et al. modelled the CSF using solid elements [95]. A hyperelastic Mooney-Rivlin constitutive model was used with the shear modulus set very low in relation to the bulk modulus, which approximates a fluid like behaviour. Adaptive meshing was used to maintain a regular mesh for the CSF, enabling flow between elements.

This approach was attempted using MPM only (not MPMICE). A Mooney-Rivlin constitutive model was used, the Poisson's Ratio was set to 0.49 (equivalent to a very high bulk modulus), both $C_1$ and $C_2$ were set to 33.5. The nature of MPM, whereby the mesh is reset at the end of each timestep, means that the CSF can flow between computational cells. This is handled inherently by MPM and no additional work is required to achieve this, in contrast to the FE model.

This approach was not successful using MPM, the simulation failed early on due to a negative Jacobian of the deformation gradient in-

dicative of non-physical behaviour (recall Chapter 2 Section 2.5.4). The timestep was lowered as far as was feasible in an attempt to overcome this, but without success. This approach was abandoned at this point, it was a crude approach to incorporating the CSF, inferior to the full FSI approach achievable using MPMICE.

### 4.3.2 *Water constitutive model*

A `water` constitutive model was available, this model offers water-like behaviour for an MPM material without using MPMICE. However, the SCI model failed (negative Jacobian of deformation gradient), the `water` constitutive model was unable to cope with such a large deformation over a short time at a feasible resolution and timestep size. MPM is designed for solid material problems, while simulating fluid flows is possible in MPM it is not what the method was intended for. As the SCI simulation involves substantial fluid deformation an MPM only approach is not optimal, necessitating the use of MPMICE.

## 4.4 GEOMETRY

As with the solid model (Chapter3, Section 3.5.4), the dura mater was modelled as an elliptical tube, $0.5mm$ thick, encircling the spinal cord and spanning the length of the computational domain from top to bottom ($z+$ to $z-$). However, the inner surface of the dura mater was positioned $1.5mm$ out from the surface of the cord. The subdural space, the space between the cord surface and the inner surface of the dura, was filled with CSF. The geometry for the CSF was again generated using an elliptical cylinder.

### 4.4.1 *Surrounding Air*

In the solid model, which used MPM only, the areas of the computational domain not containing the solid materials (cord, dura, backplate, and impactor) were filled with empty space. This was not possible using MPMICE, the pressure equilibration will fail in the presence of empty space as the volume fractions will not sum to 1. It was necessary to fill this space, therefore; air was added to surround the other materials and fill the domain. The surrounding air was modelled using the `Ideal Gas` equation of state.

## 4.5 EQUATIONS OF STATE

### 4.5.1 *Thomsen-Hartka Water*

This model for the thermodynamic behaviour of water was originally reported by Thomsen and Hartka [164], it uses a Gibbs function, which is given as follows

$$\begin{aligned} g = \frac{1}{2}b(T-T_0)^2 + (T-T_0)(c_0 + bT_0) \\ +(P - \frac{1}{2}k_0P^2)v_0 + P(\frac{1}{3}a^2P^2 + aP(T-T_0) \\ +(T-T_0)^2)\lambda v_0 + T(-c_0 - bT_0)\log(\frac{T}{T_0}) \end{aligned} \tag{138}$$

where $P$ is pressure, and $T$ is temperature. The units for these variables are given in Table 13

| Parameter | Unit |
|---|---|
| $\nu_0$ | $m^3kg^{-1}$ |
| $\lambda$ | $K^{-2}$ |
| $T_0$ | $K$ |
| $a$ | $KPa^{-1}$ |
| $k_0$ | $Pa^{-1}$ |
| $c_0$ | $Jkg^{-1}K^{-1}$ |
| $b$ | $Jkg^{-1}K^{-2}$ |

Table 13: Units for the Gibbs function parameters

### 4.5.2 *Ideal Gas*

The Ideal Gas law describes the behaviour of a hypothetically ideal gas

$$PV = nRT \tag{139}$$

where $P$ is pressure, $V$ is volume, $n$ is the quantity of gas in moles, $R$ is the ideal gas constant, and $T$ is temperature. Based on this, the equation of state for an ideal gas is given as

$$p = (\gamma - 1)c_v\rho T \tag{140}$$

where $c_v$ is the isochoric specific heat capacity, and $\gamma$ is the ratio of specific heats.

## 4.6 BOUNDARY CONDITIONS

As is the case with MPM, in MPMICE boundary conditions need only be specified on the boundaries of the domain. Interaction between

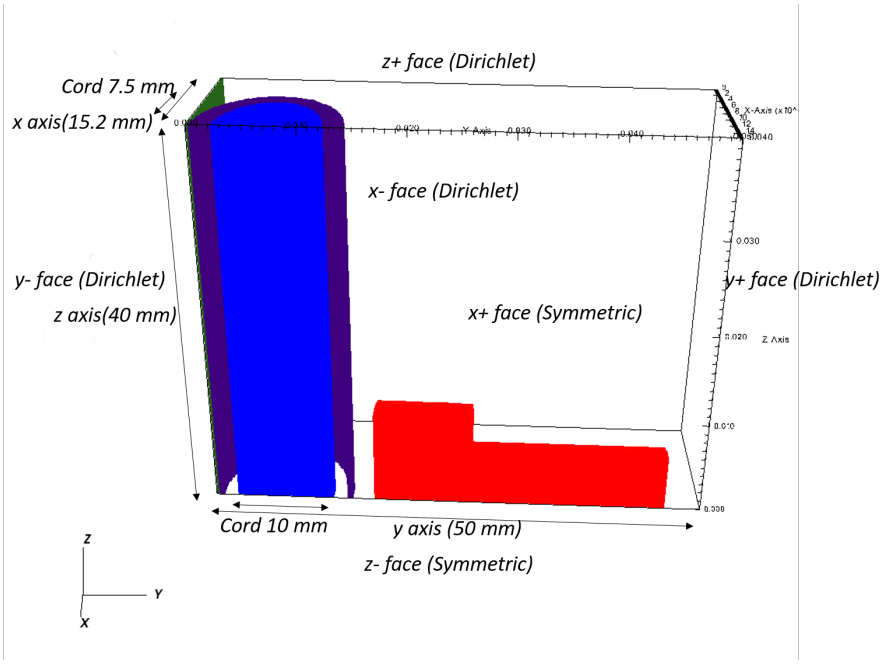



Figure 25: Computational domain with boundary conditions, symmetric boundary conditions were specified on the x+ and z− faces to exploit the symmetry of the geometry and reduce computational expense. The domain measures $15.2 \times 50 \times 40$mm.

materials is handled by the `friction` contact algorithm, as discussed in Chapter3 Section 3.9. The Fluid-structure interaction is handled inherently by the method. Symmetric boundaries are again specified on the x+ and z+ faces of the computational domain, effectively quartering the model to exploit the symmetry of the model geometry and reducing computational expense. Dirichlet boundary conditions were specified on all other faces, this effectively closed the domain with no materials entering or leaving. Neumann boundary conditions are not appropriate for the MPMICE model as they allowed the surrounding air to escape leading to a loss of pressure that would ultimately cause the simulation to fail.

## 4.7 MATERIAL DENSITY

The Thomson-Hartka water equation of state, as implemented in the UCF, will fail if the pressure in the fluid falls below 1 atmosphere (101,325 Pa). To work around this the reference pressure needs to be set much higher (20 atmospheres, 2,026,500 Pa). The pressure for individual materials is set indirectly, via the density, making it necessary to adjust the density for the component materials accordingly. It was therefore important that the density be set accurately, even tough the density of the solids varied only slightly at the increased pressure. Failure to account for this would mean that the pressure in the computational domain was not in equilibrium resulting in pressure waves emanating from the material boundaries as the pressure began to stabilise. This would cause unwanted interference in the simulations and in some case could cause them to fail. The densities for each of the materials at 20 atmospheres pressure are shown in Table 14. As the simulations are highly sensitive to density, these were calculated to 16 decimal places.

| Material | Density ($kg/m^3$) |
|---|---|
| Cord | 1092.7816666666667516 |
| Dura | 1000.0000012811723309 |
| Impactor | 1360.0003554169231847 |
| Backplate | 8000.0000119699998322 |
| CSF | 991.9896293640560998 |
| Air | 22.7617247826178648 |

Table 14: Densities for each of the materials at 20 atmospheres (2,026,500 Pa) pressure, 310.15 K temperature

Once a functioning model was achieved at 20 atmospheres the pressure was then reduced to 3 atmospheres, the material densities for which are shown in Table 15. Note that the density of the backplate

has been reduced to be equal to that of the impactor, the density of which is similar to bone. The reason for this change was that the high density of the steel was a limiting factor in determining the size of the timesteps, the lower density allowed for larger timesteps reducing overall computation time. As the backplate undergoes negligible deformation this change does not significantly affect the results. Ideally the pressure would be set at 1 atmosphere, however; the limitations of the available equation of state for the fluid prevented this, and the simulations failed when the pressure was set below this.

| Material | Density ($kg/m^3$) |
|---|---|
| Cord | 1054.5033333333333303 |
| Dura | 1000.0000001348602154 |
| Impactor | 1360.0000374123076199 |
| Backplate | 1360.0000374123076199 |
| CSF | 991.3235777144317353 |
| Air | 3.4142587173926797 |

Table 15: Densities for each of the materials at 3 atmospheres (303,975 Pa) pressure, 310.15 K temperature

### 4.7.1 *Solid Materials*

For the solid materials, the density at 20 atmospheres was calculated as follows. The bulk modulus, $\kappa$, may be defined as:

$$\kappa = -V\frac{dP}{dV} \tag{141}$$

where $\kappa > 0$, $P$ is pressure, and $V$ is volume. Equivalently:

$$\kappa = \rho\frac{dP}{d\rho} \tag{142}$$

where $\rho$ is density. Knowing both the bulk modulus of the material and its density at 1 atmosphere (101,325 Pa), it is possible to use this relationship to calculate the density at any given pressure, assuming it is an isotropic material at a constant temperature.

$$\frac{dP}{d\rho} = \frac{\kappa}{\rho} \Rightarrow \int d\rho = \int \kappa d\rho \Rightarrow \rho = \sqrt{\frac{2\rho}{\kappa}} + C \tag{143}$$

Knowing the density for the material at 1 atmosphere, it is possible to calculate C and subsequently calculate the density at 20 atmospheres, assuming a constant temperature.

### 4.7.2 *Air*

The density of the surrounding air was calculated according to the Ideal Gas law

$$\rho = P/(R_d T) \tag{144}$$

where P is the atmospheric pressure, set at 2,026,500 Pa, T is temperature, set at 310.15 K, and $R_d$ is the universal gas constant for dry air, $R_d = 287.058$.

### 4.7.3 *CSF*

The density of the CSF was calculated according to to Thomsen-Hartka Gibbs function, described in Section 4.5.1, using Equation 145.

$$\rho = ((1 - k_0 * P + (a * P + T - T_0)^2 * L) * v_0)^{-1} \tag{145}$$

## 4.8 RESOLUTION

The resolution of the underlying computational grid was set to $0.4 \times 0.4 \times 0.4\text{mm}$, giving each cubic computational cell a volume of $0.4\text{mm}^3$. The initial material resolution, which only affects the number of particles comprising the initial material geometry was set to $x = 2, y = 2, z = 1$ meaning that each cell initially contained up to 4 particles. This is a reduction in the overall number of particles compared to the solid model, where the initial material resolution was set to $x = 2, y = 2, z = 2$ (up to 8 particles per cell). This was done to reduce the computational expense, as the impact occurs transversely in the $y$ direction reducing the number of particles in the $z$ direction was considered to be acceptable and did not appear to significantly affect the results in terms of the maximum deformation.

## 4.9 PARALLELISATION

## 4.10 EXCHANGE COEFFICIENTS

The exchange coefficients determine the rate at which heat and momentum are transferred between materials, recall Step 10 in Section 4.2.2. The heat exchange coefficient between all materials was set to 0, effectively disabling heat exchange. Heat is not of interest in the SCI model and disabling heat exchange increases computational efficiency as the calculations facilitating this are no longer performed. The momentum exchange coefficients between all materials was set to $1 \times 10^{10.6}$ with the exception of the coefficient of the dura mater and the surrounding air, which was set to 1. The simulations were found to be stable at $1 \times 10^{10}$, increasing this to $1 \times 10^{10.6}$ offset the reduction of the dura–air coefficient to 1. The dura–air coefficient was

reduced to mitigate the formation of a pocket of high pressure between the dura and the impactor resulting from trapped air, which is an artefact of the simulation and not physically relevant. Reducing the impactor–air coefficient resulted in simulation failure, likely due to the extent to which air is displaced by the fast moving impactor.

## 4.11 HEAT PARAMETERS

MPMICE requires that the isochoric specific heat capacity, $c_v$, be set for all materials. In addition, the ratio of specific heats, $\gamma$, which is the ratio of isobaric heat capacity to isochoric heat capacity, must be set for the fluid materials. Heat is not a significant factor in the SCI simulation and heat exchange between materials is disabled meaning that these values have no effect on the mechanics of the SCI simulation, nevertheless it is necessary to set these values within an appropriate range to avoid the MPMICE simulation failing. For the tissues the isochoric specific heat of water at 3 atmospheres and 300 kelvin was used. For the impactor the value was based on the TUFNOL manufacturer's data, the same value was used for the backplate. For the CSF, the values were calculated based on the Gibbs function (Section 4.5.1) to be consistent with the other parameters. The values used are shown in Table 16.

| Material | $c_v$ | $\gamma$ |
|---|---|---|
| Cord | 4130 | N/A |
| Dura | 4130 | N/A |
| Impactor | 1500 | N/A |
| Backplate | 1500 | N/A |
| CSF | 4117.84 | 1.000039 |
| Air | 719.1 | 1.4 |

Table 16: Values for isochoric specific heat capacity, $c_v$, and ratio of specific heats, $\gamma$, used in the SCI model

## 4.12 VALIDATION

The FSI SCI model using MPMICE was validated by comparing the pellet trajectories against the existing experimental and finite element models. Figures 27-28 show the deformation over time for the bare cord model for two types of impactor (P1-P2), comparing the MPM, FE, and Experimental (mean average of 16 bovine specimens) models. Table 17 shows the maximum deformation (MD) and time to maximum deformation (TTMD). Due to time limitations on computational resource it was not possible to simulate P3.

The MPMICE results for P1 and P2 show a close agreement with the experimental results, falling within the standard deviation of the experimental results in both cases. The MPMICE results also show a close agreement with the FE results, although there is a slight difference in the maximum deformation (0.53mm for P1 and 0.56mm for P2), this appears to be due to the gap between the back of the cord construct and the backplate. As mentioned previously in Chapter 3, the FE trajectories each show a dip, this is due to a small gap between the cord and backplate, not present in the MPM model and too slight to record in the experiments. At a certain point the inertia is overcome, the construct then begins to move backwards, due the the way the MD is calculated this has a noticeable effect when comparing the MPMICE and FE results.

## 4.13 PARAMETRIC STUDY – CSF THICKNESS

In the experimental study the measured thickness of the CSF layer in the bovine specimens varied between animals from a minimum of 0.6mm to a maximum of 2.8mm with a mean thickness of 1.5mm. The CSF is known to have a protective effect on the spinal cord, a

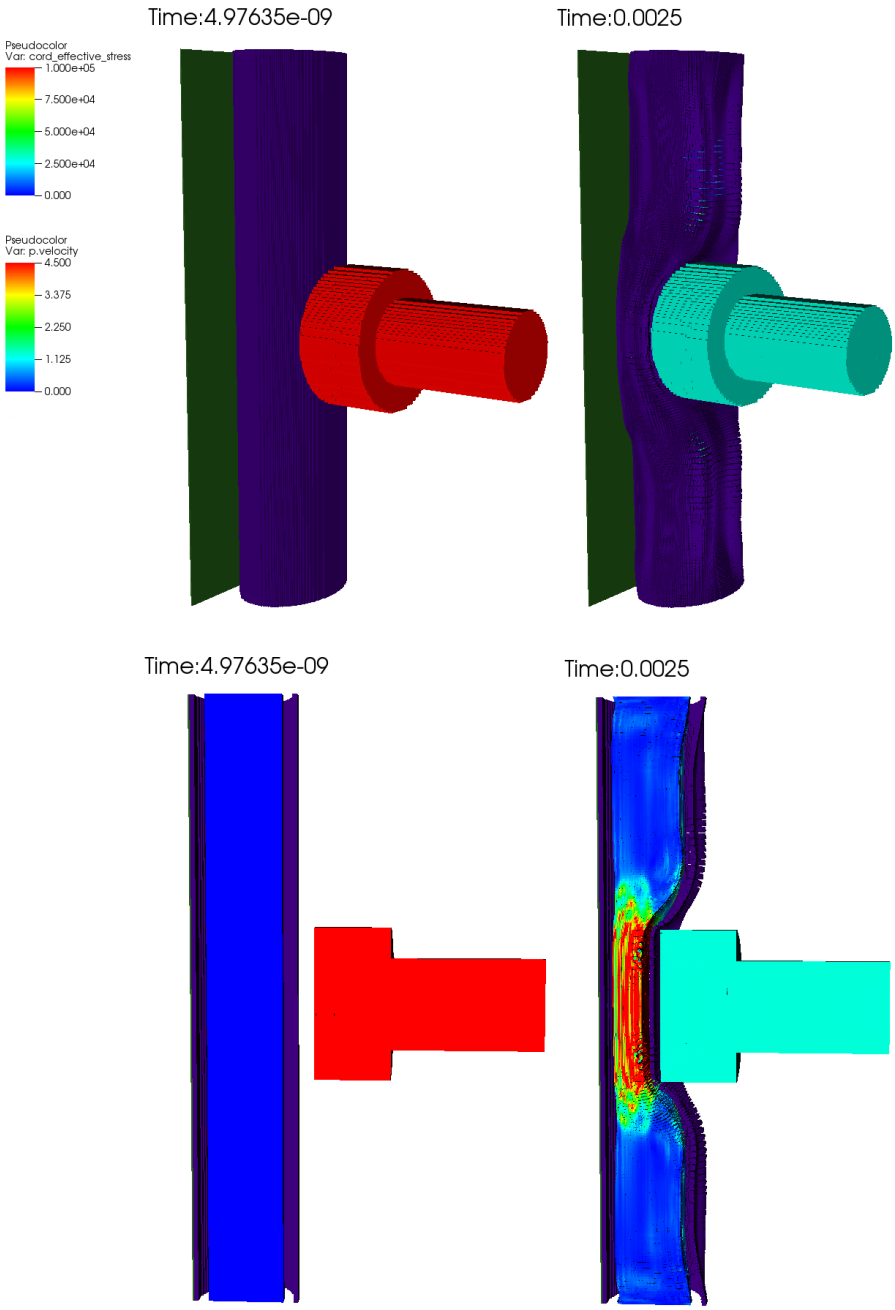



Figure 26: Visualisation of the SCI model including the cord, dura mater, and CSF. The impactor is coloured by velocity, red indicates high velocity (maximum 4.5m/s), blue indicates low velocity (minimum 0m/s). The cord tissue is coloured by equivalent stress, red indicates high stress (maximum $1 \times 10^5$Pa), blue indicates low stress (minimum 0Pa). Time is shown in seconds.

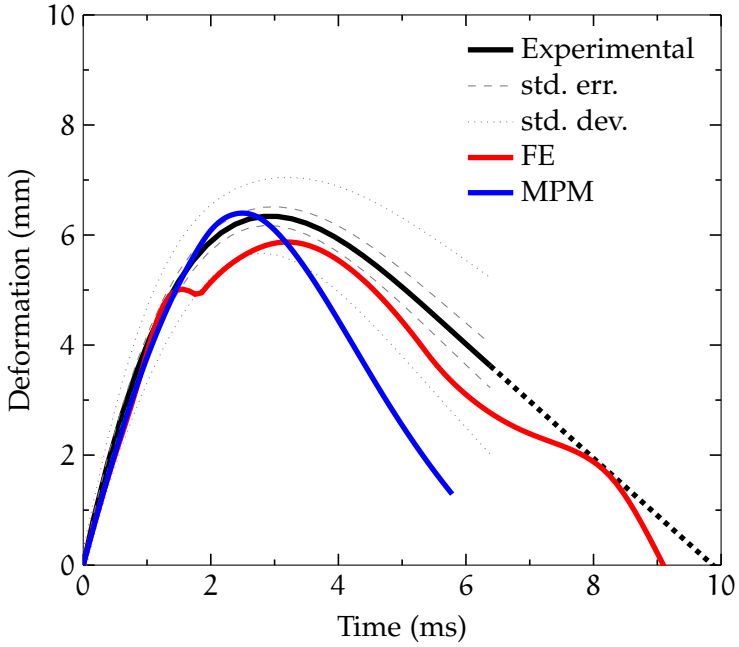



Figure 27: Pellet 1, cord/dura/CSF model

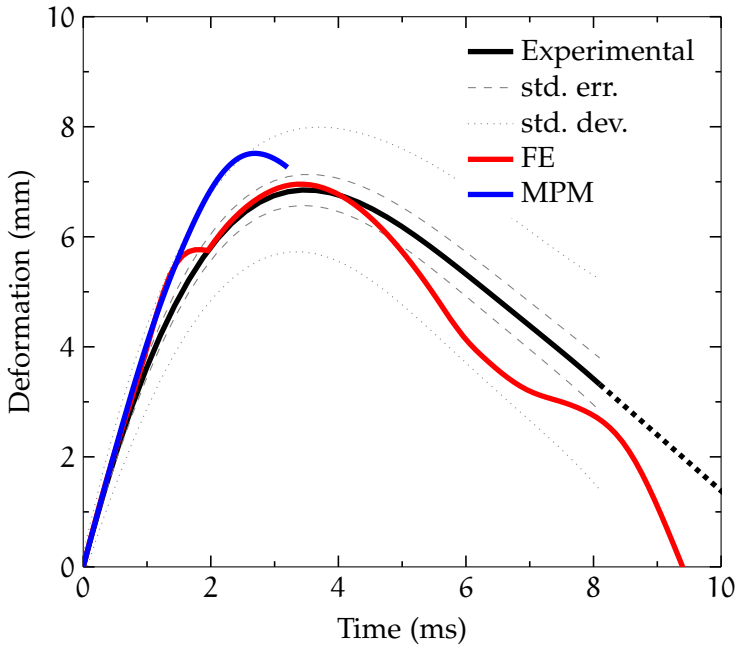



Figure 28: Pellet 2, cord/dura/CSF

| | | MD (mm) | TTMD (ms) |
|---|---|---|---|
| P1 | Exp. | 6.34 (±0.17) | 3.096 (±0.15) |
| | FE | 5.87 | 3.15 |
| | MPM | 6.4 | 2.5 |
| P2 | Exp. | 6.85 (±0.28) | 3.78 (±0.18) |
| | FE | 6.96 | 3.36 |
| | MPM | 7.52 | 2.70 |

Table 17: Maximum Deformation (MD) and Time to Maximum Deformation (TTMD) for the 3 pellet types (P1-P2) in the experimental, FE, and MPM models, FSI model including the cord, dura mater, and CSF.

preliminary parametric study was conducted to assess the impact of varying the thickness of the CSF layer on the internal deformation of the spinal cord. This study looked at deformation of the spianl cord only, not the entire cord construct including the dura mater. Three simulations were run with thickness of the CSF layer set to 0.6mm (thin), 1.5mm (base), and 2.8mm (thick), due to limitations on the available computational resource these were conducted in 2D.

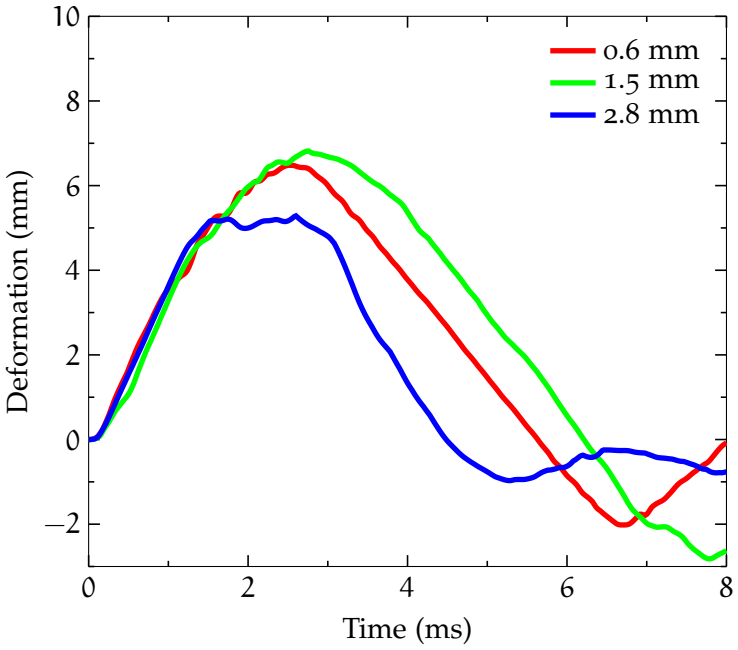



Figure 29: Deformation over time of the spinal cord, within the CSF and dura mater, for varying thickness of the CSF layer.

| CSF (mm) | MD (mm): | TTMD (ms) |
|---|---|---|
| 0.6 | 6.48 | 2.5 |
| 1.5 | 6.82 | 2.76 |
| 2.8 | 5.29 | 2.6 |

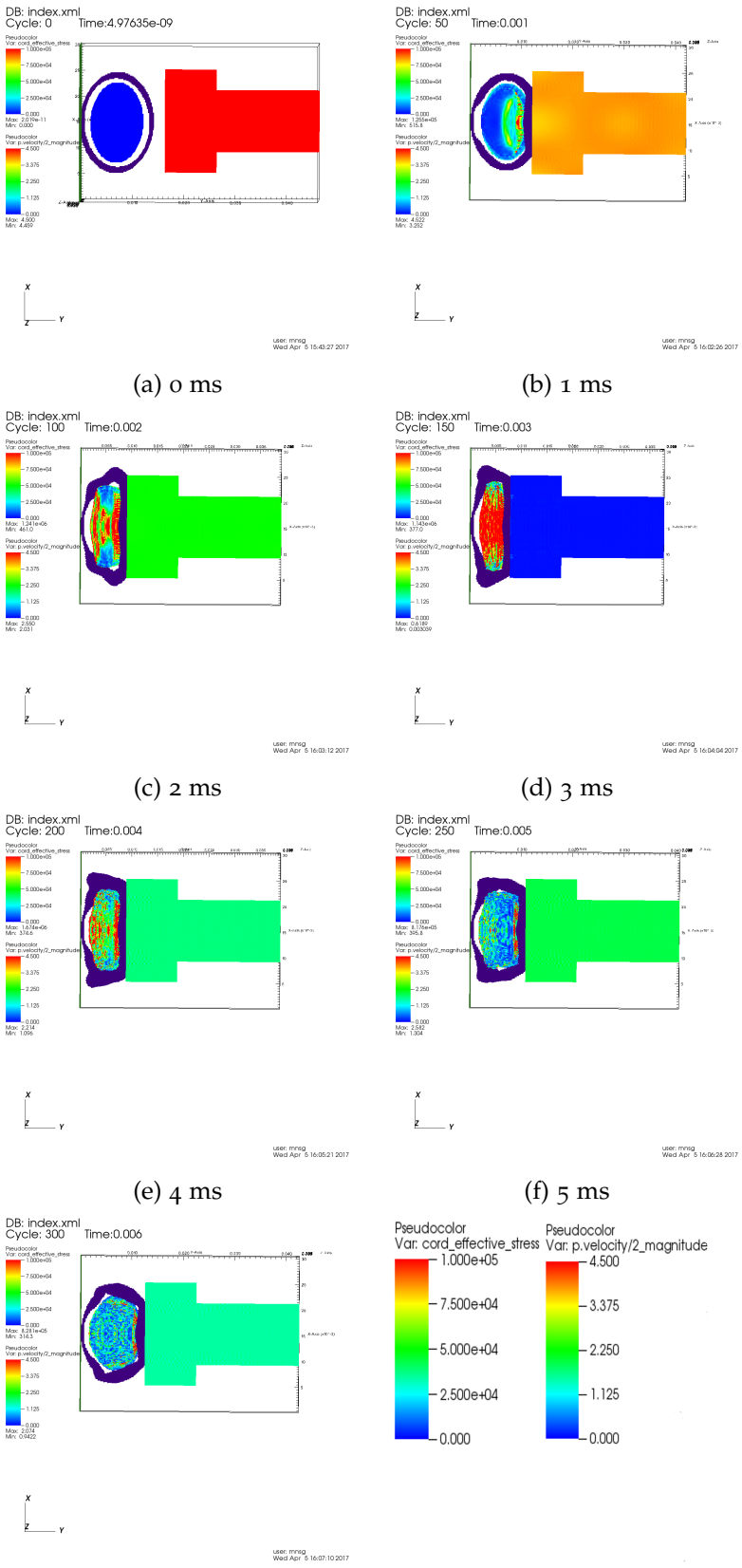
Table 18: Table showing the internal Maximum deformation (MD) and Time to Maximum Deformation (TTMD) of the spinal cord, within the CSF and dura mater, for varying thickness of the CSF layer.

Figure 29 shows the deformation over time curves for each of the simulations, the Maximum deformation (MD) and Time to Maximum Deformation (TTMD) is shown in Table 18. The thick layer resulted in a 22.4% reduction in maximum cord deformation compared to the base layer. The thin layer resulted in a 5% reduction in the maximum cord deformation compared to the base layer. The result for the thin layer is unexpected as a thinner layer of CSF is thought to absorb less energy and therefore offer less protection to the cord. It is likely that this unexpected result is due to the fact that these tests were conducted in 2D. It is not possible for the fluid to be displaced in the $z$ direction in the 2D simulations, therefore the protective effect of the fluid is diminished. In order to properly ascertain the sensitivity of the model to the thickness of the CSF layer these experiments will need to be repeated in 3D.

## 4.14 STRESS PATTERNS

Figure 30 shows the deformation of the spinal cord construct at various time points. The white space between the cord and dura is filed with CSF. The impactor, travelling at $4.5\mathrm{m/s}$, strikes the outer surface of the spinal dura mater and slows down to $0\mathrm{m/s}$ as the cord construct deforms to the point of maximum deformation, the impactor then begins to recoil. Internally, the cord begins to deform immedi-

ately after the impactor strikes the outer surface of the dura, energy of the impact is transferred through the dura mater and the CSF into the cord tissue. Following the impact a wave of high stress propagates from the area of impact through the spinal cord tissue in the transverse direction until it hits the rear of the construct, at which point it bounces back creating an area of high pressure behind the site of the impact. The pressure wave also propagates in the longitudinal direction both up and down the length of the cord. This wave travels through both the solid and fluid phases, as the impactor moves towards the cord construct the CSF is displaced upwards and downwards, reducing the stresses in the spinal cord. A pocket of CSF remains behind the cord throughout the deformation. The maximum stress within the cord tissue occurs at the point of maximum deformation.




(a) 0 ms (b) 1 ms

(c) 2 ms (d) 3 ms

(e) 4 ms (f) 5 ms

(g) 6 ms (h) Scale bars

Figure 30: Deformation of the spinal cord construct at various time points. The dura mater is coloured purple, the spinal cord is coloured blue for areas low stress (0Pa) to red for areas of high stress ($1 \times 10^5$Pa). The impactor is coloured by velocity, red for high velocity (4.5m/s) and blue for low velocity (0m/s). The white space between the cord and dura is filed with CSF.

## 4.15 CONCLUSIONS

A burst fracture spinal cord injury model was created using the MPMICE implementation within the Uintah Computational Framework, this is an FSI model incorporating both solid and fluid material phases and the physical interaction between the two. The model consisted of a simulated bone fragment transversely impacting the surface of the spinal cord construct, which included the cord, dura, and CSF to fill the subdural space. Two impactors with different geometries were tested.

The MPMICE FSI model was validated against existing FE and experimental results by comparing the deformation over time curves, maximum deformation, and time to maximum deformation. MPM is more computationally expensive per timestep than FE, this is because there are additional steps required to interpolate between the particles and the computational grid nodes. MPMICE further adds to the computational expense as the interactions between phases must also be accounted for. However, like MPM, MPMICE is highly suited to parallel computation and the increase in computational expense can be overcome by utilising additional processors working in parallel. The validation has shown that MPMICE is a viable alternative to FE for computationally modelling SCI.

# 5

# MECHANICAL CHARACTERISATION OF THE PIA MATER

## 5.1 INTRODUCTION

The pia mater, the innermost meningeal layer, is a thin and delicate layer of fibrous tissue that adheres to the surface of the spinal cord and brain, separating them from the cerebrospinal fluid that occupies the subdural space [21]. Computational models of traumatic spinal cord injury are often used in conjunction with animal studies to investigate the biomechanics of the primary injury and the subsequent physiological effects [14]. Material behaviour in these computational models is determined by constitutive material models, knowledge of the mechanical properties are essential in determining the appropriate material parameters for these models. The mechanical properties of spinal cord pia mater are relatively uncharacterised, the aim of this preliminary study characterise the mechanical properties and to determine and refine a suitable method for doing so. The mechanical testing results were used to determine Ogden hyperelastic material model constants, which may be used to replicate the mechanical behaviour of the tissue in computational biomechanical models.

## 5.2 METHODOLOGY

Samples of bovine spinal cord pia mater were obtained from 3 fresh sections of bovine spine, obtained from a local abattoir. Between tests

the spinal cords were kept refrigerated in a hermetically sealed container. Table 19 details the sample dimensions and the time from slaughter to testing for each sample.

The process of preparing the pia mater samples for testing was difficult, the delicate tissue was easily damaged during the process of isolating it from the other spinal cord tissues.

| Sample # | Length (mm) | Width (mm) | Specimen # | Time since slaughter (days) |
|---|---|---|---|---|
| 1 | 12.5 | 11 | 1 | 2 |
| 4 | 18.9 | 8.3 | 1 | 8 |
| 6 | 16.6 | 8.8 | 2 | 1 |
| 7 | 10.8 | 14.6 | 2 | 1 |
| 8 | 10.19 | 6.4 | 2 | 1 |
| 10 | 29.8 | 8.45 | 2 | 1 |
| 11 | 39.8 | 5.2 | 3 | 2 |
| 12 | 38.9 | 3.8 | 3 | 2 |
| 13A | 19 | 15 | 3 | 3 |
| 13B | 13.1 | 17 | 3 | 3 |

Table 19: Pia mater samples. Sample 13A slipped in the clamps at the start of the test, as it had barely deformed it was clamped again and re-tested (Sample 13B).

Section 5.3.1 describes the methodology used to prepare the samples and Section 5.3.2 describes the testing. Numerous issues were encountered over the course of the study, this gave way to refinements to the methodology which were carried forward, these are discussed in the Section 5.3. In addition to the Shimadzu tensile testing machine, stereo Digital Speckle Photography (DSP) was used to calculate the strain fields in the samples. This required the samples to be speckled with paint to provide points for the cameras to track.

## 5.3 METHOD REFINEMENT

### 5.3.1 *Initial sample preparation*

An initial attempt was to isolate the pia from a section of bovine spine, obtained from an abattoir. This specimen had been used previously for unrelated testing, and was frozen. This is likely to have significantly affected the material properties of the tissue, although this specimen may not be suitable for testing it was possible to use it to practice and refine the pia removal process. During the butchering process the cow was bisected along the median plane, in turn bisecting the spinal column and the spinal cord within. The spinal cord was removed from the spinal canal, the bisection made it easier to remove the dura mater, leaving the cord itself (predominately grey and white mater) still attached to the innermost meningeal layer: the pia mater. Initially, tweezers were used to grip one corner of the pia and a scalpel to cut and scrape away the underlying neural tissue. This approach was not optimal, as the pia was easily torn or punctured.

Aimedieu and Grebe outlined a method for isolating sections of pia mater from bovine brains [165]. Sections of paper are placed onto the brain, The moisture in the tissue causes the pia to adhere to the underlying paper. These sections are cut around using a scalpel to extract a small section of brain. The neural tissue is then scraped away, leaving the pia mater still adhered to the paper. The authors report that this is possible due to the high viscosity of the neural tissue in comparison to the pia. The sample may then be clamped into a testing rig with the paper still attached. The paper, weakened by the moisture, is then peeled away leaving only the pia mater in the test machine. This technique was adapted to remove the spinal pia mater.

A second bovine spinal was obtained, this one from an animal slaughtered the same morning. The spinal cord seemed to have avoided the saw and a whole section was removed intact. A small piece was then cut off and the remainder stored for later use. The dura was removed and the cord opened up by slicing along the anterior median fissure. The specimen was placed on a sheet of paper, pia facing downwards and in contact with the paper. Incisions were then made in the neural tissue to allow the cord to unfurl over the paper, such that all of the pia was in contact with it. Tweezers were used to grip a corner of the pia, and the neural tissue was scraped away with the reverse of the scalpel blade. As reported by Aimedieu and Grebe, the viscosity of the neural tissue allows it to be peeled back from the stiffer pia mater underneath. In this way it was possible to isolate a small section of pia mater, a small amount of neural tissue remained, although not ideal it was felt that attempting to remove it would damage the sample beyond use. The sample, still adhered to the paper, was placed in a sealed container alongside some moistened paper towels and kept in the the fridge until testing. Unfortunately, due to to equipment availability, it was not possible to test this first sample on the day it was prepared.

### 5.3.2 *Initial sample testing*

Prior to testing the sample was removed from its container, still adhered to the paper. Acrylic spray paint, both black and white for contrast, was sprayed onto the sample to give it the speckle pattern necessary for DSP. It was important at this stage that no large drops of paint fall upon the sample, as this would form a skin that would confound the tensile testing results, this issue was identified in previous experiments using the same technique. The sample was clamped

at both ends, and placed in the tensile testing machine, the paper was then peeled off. Unfortunately during the preparation the sample and paper had dried out and was difficult to remove, but it was possible by rehydrating the sample with distilled water. This is likely to have had a significant affect on the material properties. The type of paper used was not optimal, once the sample was clamped in place it was very difficult to remove. The paper had a tendency to tear and break off in small clumps, making the removal process slow and tedious, which was detrimental to the quality of the sample. Despite these issues a section of pia mater was successfully loaded into the tensile testing rig. The sample was preconditioned immediately prior to testing, five cycles of 0.5 mm extension at 100 mm/minute, the sample was then strained until the point of rupture at 1 mm/min. It was possible to calculate the strain field in the sample using DSP, however; parts of the sample were not recognised, this may have been caused by suboptimal coverage of the paint speckled exacerbated by the drying out and re-moistening of the tissue.

Overall the experiment was a partial success, the test was carried out demonstrating that the methodology is viable. However, substantial refinement of the process was required for the results to be anatomically relevant. Several key points were identified. The choice of paper is important, in terms of how it affects the initial isolation of the pia, and then the removal of the paper once the sample is in the clamps. All neural tissue should be completely removed. The sample should be kept hydrated throughout the process, and should be excised immediately prior to testing to prevent it drying out or otherwise degrading further. Finally, it was decided that a spray bottle filled with a saline solution should be used to periodically spray the sample to prevent dehydration, as a humidity chamber for the test rig was not available. As the tissue sample is very thin, it was

assumed that it had warmed to room temperature during the preparation phase, warming it to body temperature was not feasible.

### 5.3.3 *Sample preparation*

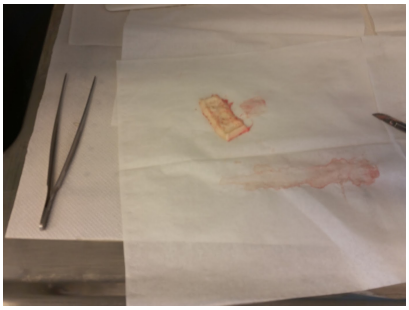
When removing the neural tissue from the pia, fingertips work best. Tweezers risk puncturing or tearing the pia. Once the neural tissue has been peeled away, a section of pia remains adhered to the underlying substrate. Some holes in the pia are unavoidable due to the protrusion of the nerve roots, which must be severed to remove the spinal cord from the spinal canal. It is preferable to cut the sample into the desired shape using scissors. For the purposes of the preliminary study, a roughly rectangular shape was used. When cutting the sample to shape it is preferable to remove any larger blood vessels, which may confound the results. While it is not feasible to completely remove the vasculature, the longer thicker blood vessels running longitudinally along the spinal cord should be removed if possible otherwise it may be that the test results better reflect the mechanical properties of the blood vessel rather than those of the pia. Finally the samples were sprayed with a speckle pattern of acrylic paint, being careful not to apply too much paint, ruining the sample. Spraying alternately with both white and black paint, twice with each colour, provided the best results. The speckle pattern on the earlier samples was poor and the DSP did not work well, the patterns on the later samples were much improved and the DSP worked well.

### 5.3.4 *Backing paper*

The purpose of the paper is twofold: firstly, it assists in separating the pia mater from the adjoining neural tissue. Secondly, it allows

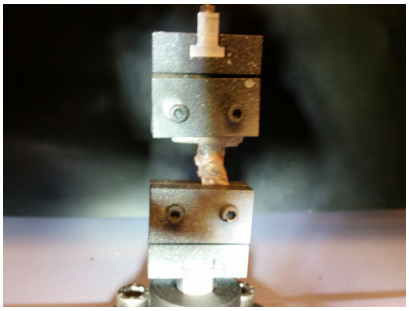
(a) Spinal cord removed from vertebral canal (dura mater also removed)

(b) Small section of spinal cord removed from larger specimen

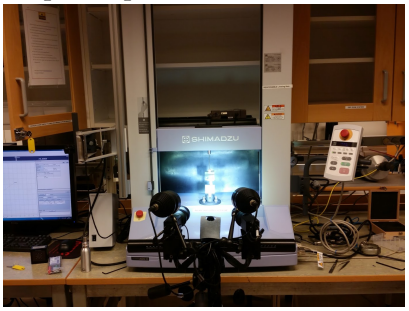


(c) Cord section cut and unfurled, pia mater facing down

(d) Pia mater on substrate with neural tissue scraped away, sprayed with black and white paint speckles



(e) Pia sample in clamps ready for tensile testing



(f) Cameras configured for digital speckle photography (DSP)

Figure 31: Preperation of bovine spinal cord pia mater for mechanical testing.

the sample to be placed into the clamps as a sheet, without it curling up or tearing. The properties of the paper are important, the initial paper used was not rigid enough to aid much in scraping away the neural tissue and was also difficult to peel away from the sample once clamped, resulting in some damage. Numerous varieties of paper were experimented with, after some practice the most difficult part of the procedure became getting the sample into the clamps rather than removing the pia from the neural tissue. The most effective substrate was found to be a thin sheet of polypropylene, obtained from a plastic CD sleeve. This material offers less friction than paper, making peeling away the neural tissue a slightly more difficult task, but making the removal of the substrate significantly easier once the sample was in the clamps. The polypropylene sheet does not tear, instead tweezers were used to gently separate the pia from the sheet near the lower clamp, which was then cut with scissors, peeled back, and cut again near the upper clamp. For earlier samples one end of the tissue was clamped and the other left hanging, before being clamped. It was found that clamping both ends before removing the backing sheet was more effective.

### 5.3.5 *Hydration*

Once removed from the surrounding tissues, the pia mater dries out quickly. To prevent this the sample was sprayed every 1 minute with a fine mist of distilled water, sprayed from an atomiser. Once the sample had been clamped and the backing sheet removed this was increased to every 30 seconds, as the pia then has twice as much surface area through which moisture may be lost.

### 5.3.6 *Clamps*

The samples all either failed at the clamps, rather than in the middle, or started slipping. With this delicate tissue there is a very small window between too loose (resulting in slippage) and too tight (resulting in breakage at the clamps). This issue may be helped by padding the clamps with paper to act as a buffer. Scissors were used to cut the samples shape whilst still adhered to the substrate.

### 5.3.7 *Preconditioning and Loading*

Samples were preconditioned with 5 strokes at 50 mm/min, each extending the samples by approximately 10% strain. From sample 11 onwards this was reduced to 5% strain based on analysis of previous results. Following preconditioning samples were extended at 2 mm/min to failure.

## 5.4 RESULTS

The pia mater is a thin and delicate membrane, adhered to the underlying neural tissue, removing it intact is challenging, furthermore; sections of pia that were removed would curl inwards in a manner that made it very difficult to obtain a rectangular section for testing without further damaging the tissue. The quality of the samples improved over the course of the experiments, reflective of practice at removing the tissue. A usable sample could not be obtained with the approach described in Section 5.3.1, usable samples were obtained using a backing sheet, as described in Section 5.3.4. The pia mater was observed to be an extremely thin, transparent tissue with a degree of

elasticity. The underlying neural tissue was observed to be soft and viscous, white/beige in colour.

The tensile testing results for the bovine pia mater samples is shown in Figure 32, the pia mater exhibits a hyperelastic response to loading. It was not possible to test all of the samples as some were damaged during preparation and loading into the test machine. The mean result (taken from the platen separation) is shown in Figure 33, the variance between experimental runs was substantial, the high standard error reflects this. The samples all broke near to the clamps, a small number began slipping in the clamps. Subsequently there is only valid stress-strain curves for small strains. It was not possible to determine the strength. The mean curve was fitted to a single term hyperelastic Ogden model, yielding the Ogden constants: $\mu = 0.020$ $\alpha = 47.93$. These are used in the Ogden strain energy density function:

$$W = \frac{\mu}{\alpha}(\lambda_1^\alpha + \lambda_2^\alpha + \lambda_3^\alpha - 3) \tag{146}$$

In addition to the stress strain curves the mean average stress in the direction of loading was calculated. The DSP provided a strain field over the whole tissue sample, although for many of the samples the system did not recognise the speckle pattern. The middle 50% of the tissue was analysed, the tissue near the clamps was excluded. Figure 34 shows the resultant stress strain curve. Overall the DSP results seem to be more robust that those obtained from the DIC. Unfortunately few of the samples worked well with the DSP, in many cases the speckle was not recognisable enough.

The tensile testing results for the bovine pia mater samples is shown in Figure 35. The pia exhibited hyperelastic material behaviour, the variance between experimental runs was substantial, the high stan-

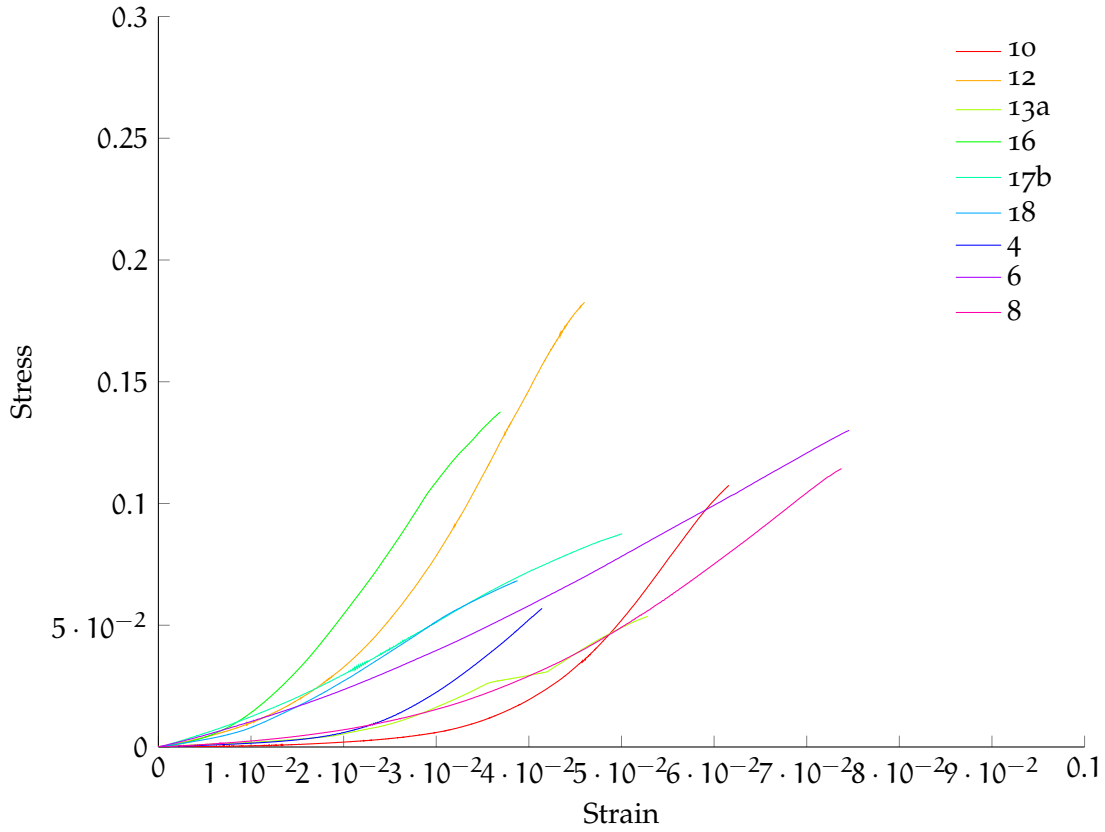



Figure 32: Stress strain curves obtained though tensile testing of pia mater samples. Strain calculated based on distance between the platen. Curves have been trimmed at the point of material failure.

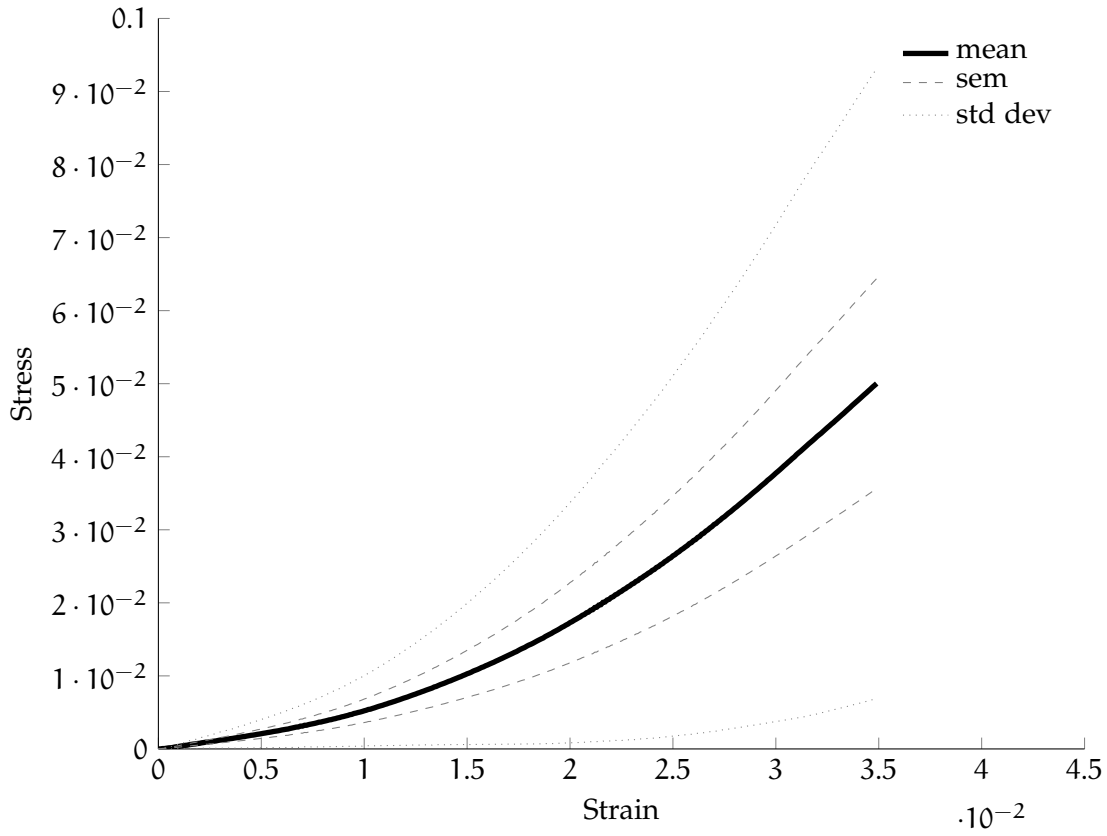



Figure 33: Mean stress strain curve for the pia mater samples with the standard error of the mean and the standard deviation.

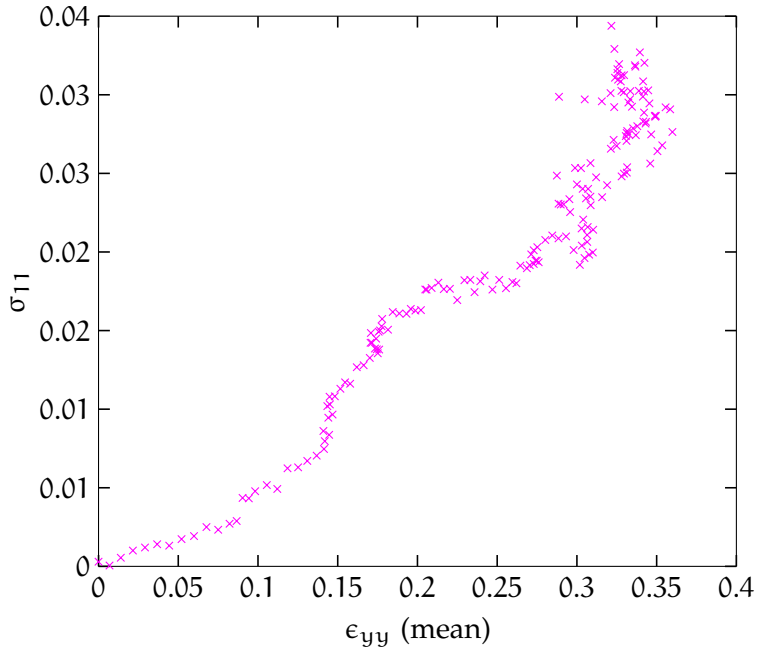



Figure 34: Mean average strain in the direction of loading for sample #11, calculated using DSP.

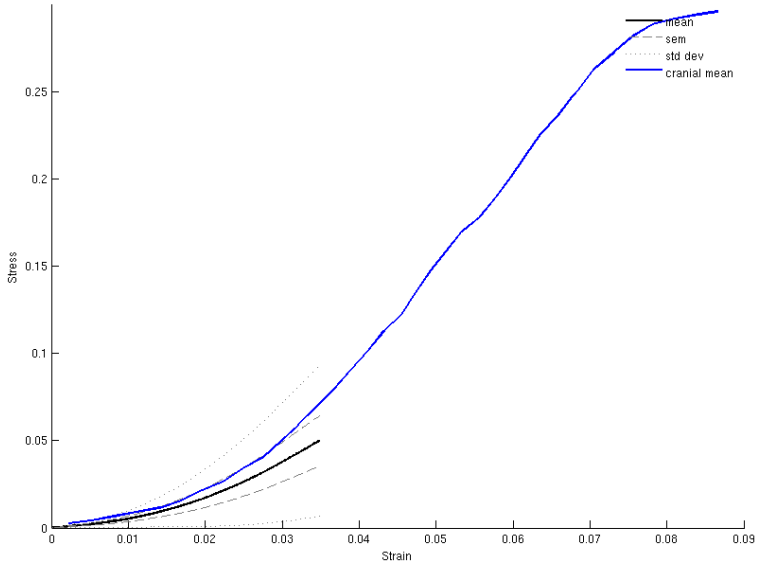



Figure 35: The mean stress vs strain for the bovine spinal pia compared to published mean result for bovine cranial pia mater [165].

dard error reflects this. The samples all broke near to the clamps, a small number began slipping in the clamps. Subsequently there is only valid stress-strain curves for small strains. It was not possible to determine the strength. The mean stress vs strain (taken from the platen separation) was compared to published results for bovine cranial pia mater [165]. For the portion of the curve available no statistically significant difference was observed between the two.

## 5.5 OGDEN MATERIAL PARAMETERS

The mean stress–strain curve was fitted to an Ogden hyperelastic model to determine the parameters, these may be used to recreate the mechanical behaviour of the pia in computational SCI models. These are shown in Table 20. A single term Ogden model was used and a good fit, however; the high $\alpha$ values may be computationally expensive due to the number of `pow()` operations required. If this is

an issue then a two term Ogden model may be able to achieve similar behaviour with lower $\alpha$ values.

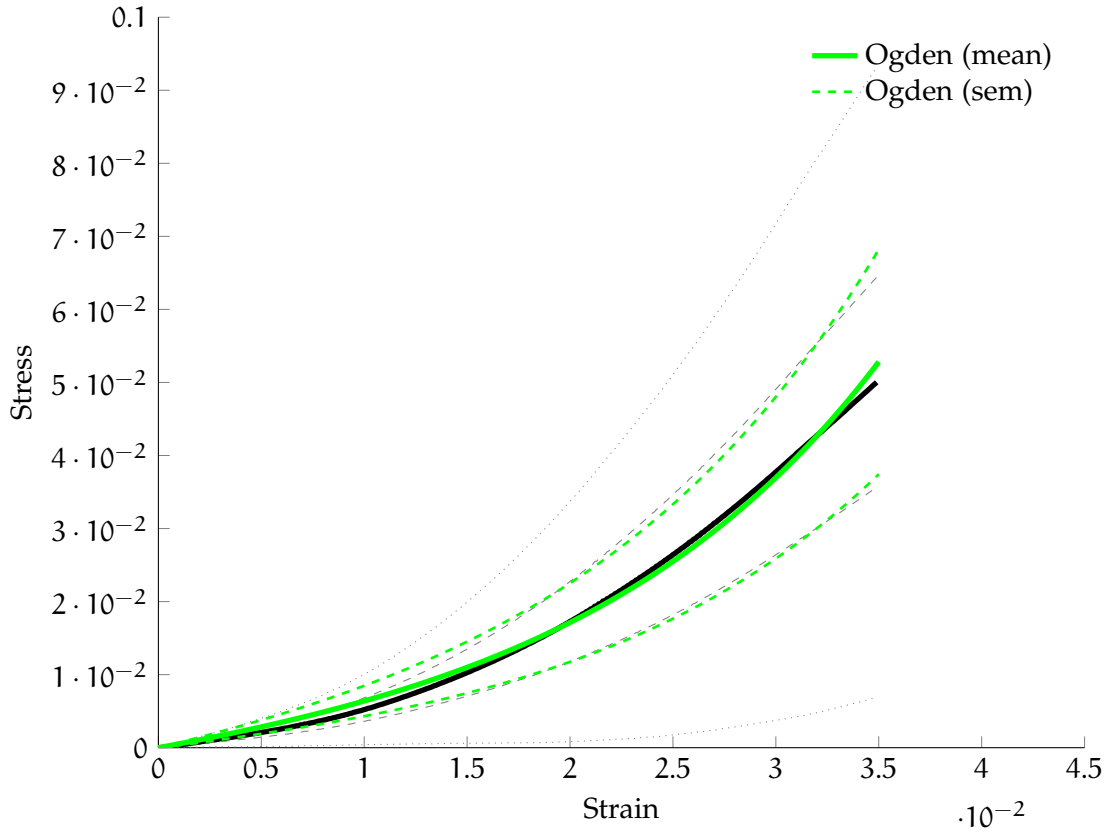



Figure 36: Mean stress strain curve fitted to the Ogden constitutive model. Curve fitting was also applied to the mean curve plus and minus the standard error of the mean. The corresponding material constants are shown in Table 20.

## 5.6 DISCUSSION

Mechanically characterising the pia mater is challenging, due in large part to the delicate nature of the tissue. Using a method adapted from Aimedieu and Grebe it was possible to mechanically test number of samples. The results of this preliminary study are of very limited use. The quality of the samples improved greatly over the course of the study due partly to improved human skills preparing the pia for testing, partly due to refinements to the method, described in Section 5.3.

| | Ogden material constants |
|---|---|
| -sem | $\mu = 0.0031$, $\alpha = 73.7509$ |
| mean | $\mu = 0.0049$, $\alpha = 70.9881$ |
| +sem | $\mu = 0.0066$, $\alpha = 69.5055$ |

Table 20: Ogden material constants determined from experimental data, corresponding curves shown in Figure 36.

Switching to a plastic backing sheet led to significant improvements, as this made the process of removing the backing sheet much easier once the sample was clamped, this is the stage at which the most samples were accidentally damaged. It was not possible to determine the material strength from these results as all the samples failed at the clamps. However, the methodology has been sufficiently improved to avoid this issue in future studies.

When clamped in place, even the best samples were not entirely flat, this was unavoidable. Small creases and folds in the tissue meant that the load was not distributed evenly over the width of the samples. This likely contributed a great deal to the samples failing at the clamps, and is also likely responsible for the partial failures visible in Figure 32. Going forward the samples should be cut into a dog bone shape to reduce the number of samples that break in the clamps. This was not done in this preliminary study due to the difficulty incurred when extracting the samples. However, this should now be possible with a more practised hand and refined technique. Small areas of creased tissue start out bearing the majority of the load, these sections fail and the load is then picked up by the surrounding tissue, which had been experiencing low strain up until that point. This issue may be overcome in part by cutting the samples into the dog-bone shape, and padding the clamps to avoid slippage and tearing. It may also be overcome using the DSP. The use of DSP proved challenging, however towards the end of the study the speckle pattern on

the samples was being fully recognised. This was one of the aspects of the methodology that simply improved with practice. The use of DSP made it possible to exclude the tissue around the clamps and focus only on the middle section. More results are needed, but these preliminary results hint that the DSP results may be more robust than the readouts from the DIC.

## 5.7 CONCLUSION AND FUTURE WORK

A preliminary study was performed to attempt to characterise the mechanical response of bovine spinal cord pia mater to uniaxial loading, with a view to using the results to calculate material constants for hyperelastic constitutive material models. The pia is a thin and delicate tissue, extracting and testing it proved very challenging. The results of this study are of limited use due to the high variability between the samples, small number of samples, and sample breakage at the clamps. Despite this the experimental methodology was refined considerably. With the improved methodology, detailed here, and increased experience it will be possible to conduct further experiments with greater control over the parameters.

# 6

# DISCUSSION

## 6.1 INTRODUCTION

The Material Point Method and MPMICE, in their Uintah implementation, were initially developed to meet the needs of simulating explosions of energetic devices [134, 156]. This work sought to apply this method in modelling Spinal Cord Injury, in particular in creating a spinal burst fracture injury simulation to be validated against existing experimental studies and computational studies using an Finite Element Modelling approach. The motivation for doing this stemmed from inherent limitations of FE modelling for tackling this particular class of problem, involving large material deformations over a very short time in the presence of a fluid structure interaction. FSI is required due to the presence of the CSF occupying the subdural space. These limitations are compounded by the additional burden of requiring complex material geometries.

In large deformation problems FE can be susceptible to mesh tangling, also known as element inversion, which occurs when elements become inverted, overlap with adjacent elements, or due to hourglassing (twisting of the elements into an hourglass shape). This must be addressed by recreating the Lagrangian mesh and/or lowering the size of the timestep. Lowering the timestep is costly in terms of increased computational expense, increasing the number of timesteps required to complete the simulation. Remeshing is costly in terms of the time, effort, and expertise required to establish the initial geometry based on the geometric parameters of the given problem, in this

case the structure and dimensions of the spinal cord tissues. Mesh tangling can be particularly frustrating as the issue may not become apparent until after many hours of computation. Incorporation of the CSF is also challenging, the Lagrangian mesh requires the tracking of boundaries between materials, at fluid-solid boundaries additional computational operations are required to facilitate the FSI. Solid materials are best suited to modelling using the Lagrangian frame of reference, whereas fluids are best suitable to the Eulerian frame. The phases are typically separated with interactions being handled by algorithms operating at the phase boundary. Due to these complexities many past studies have chosen to exclude the CSF altogether [18, 19, 53, 118], or to include it in a simplified form using either a solid material approximation [93] or by using an implied FSI [54] that omits the true, full physics, FSI.

In FE, the computational mesh defines the constitutive elements this is necessary to maintain material separation at the material boundaries. As such, handling material fracture and coalescence is not inherently suited to the FE method. These challenges are compounded by the need to include the complex geometry reflective of the spinal cord anatomy. FE studies of SCI to date have therefore utilised a highly simplified geometry, the model presented in this work also uses a simplified geometry, representing the spinal cord and surrounding tissue as elliptical cylinders. As the characteristics of SCI progresses, however; there will be a need for ever increasing levels of detail, perhaps ultimately down to a cellular level and beyond. All of effectively comes down to an issue of computational expense, and subsequently the time required to compute the solutions.

For computational SCI models to be an effective research tool it is necessary to be able to run the models to completion within a feasible timeframe. Computational expense many be met with increased computation resource, however; high performance computation facil-

ities typically rely on a high level of parallelisation. In order to benefit from such resources the methodology selected must be suited for parallel computation, with the ability to suitably divide the computational domain accordingly. Unfortunately, this is difficult using typical FE formulations due to the irregularity of the Lagrangian mesh combined with the need to track the material boundaries over the entire domain. The computation overheads requires for division and recombination quickly begin to offset the performance gained as the number of processors increases. While an explosion simulation may bear little resemblance to an SCI simulation, both involve high deformations over short timespans in the presence of both solid and fluid phases. The arbitrary Lagrangian-Eulerian approach employed by MPM and MPMICE overcomes these issues, with the benefit of also being highly amenable to parallelisation. The scalability of MPM and MPMICE for parallelisation was the key driver in selecting these methods for this work, the Uintah release used here is capable of running on up to 100,000 parallel cores [153, 154, 156].

## 6.2 MPMICE FOR SCI MODELLING

### 6.2.1 *An ALE Approach*

For efficient parallelisation an Eulerian grid is preferable, the use of a regular computational grid makes it substantially simpler to divide the computational domain across processors. FE formulations generally utilise a Lagrangian grid. The reason for this is that the use of an Eulerian gird is problematic for solid materials where the constitutive material models are dependent on time history dependent variables, such as deformation, velocity, and temperature gradients. For example, the Ogden model used in this work, described in Chapter

2, was dependent on the deformation gradient for the stress calculations. The need to transport deformation history through an Eulerian mesh is problematic, the lack thereof can lead to diffusion at material boundaries and non-physical stresses. MPM overcomes this issue by introducing particles and interpolating between these and the underlying grid [128, 134]. In effect, MPM facilitates solid material deformations on a regular Eulerian mesh using what the authors termed a "computational scratchpad" to advance the solution. This is based on the idea of the deformable mesh that resets to the original position at the end of each timestep, as discussed in Chapter 3. This also facilitates the MPMICE FSI integration, discussed in Chapter 4. MPM is described as an Arbitrary Lagrangian Eulerian (ALE) approach, incorporating the benefits from both frames of reference whilst avoiding the drawbacks.

### 6.2.2 *Advantages*

The use of MPMICE for this SCI model offered numerous advantages beyond parallel scaling. The particles (material points) replace the polyhedral elements used in FE, the particles exist at a single point in space and can be thought of as Dirac Delta functions. All solid materials, no matter their mechanical properties, are represented through the particles and their state vectors (carried on the particles themselves). The introduction of the particles with their ability to move freely between the regular grid cells makes the method highly resilient to mesh tangling. This is most likely to occur due to a large deformation over a short time, making this a significant issue for SCI simulations. In this regard the use of MPM/MPMICE was highly beneficial, demonstrating its suitability for high-rate-of-deformation dynamic loading problems.

The particles negate the need for differently shaped elements to comprise the material geometry. In addition to the benefits of avoiding mesh tangling, this greatly simplifies the process of defining the initial material geometry. Generation of a Lagrangian mesh for FEM requires the selection of element types (e.g. tetrahedral, quadrahedral, etc.) based on the desired arrangement and properties of the materials being modelled. This complicates the generation of the initial mesh and greatly increases the complexity of automating the process for generation of the initial geometry from image data, such as from spinal computerised tomography (CT) scans or magnetic resonance imaging (MRI) scans. This is particularly useful for generating parametric models from specimens, and makes it feasible to image a specimen prior to experimental testing for the creation of a complimentary computational model. Furthermore, the boundaries between materials need to be explicitly identified and defined. In contrast the initial point cloud generation in MPM is trivially simple, all materials are represented through the particles and there is no need to define material boundaries (except on the boundaries of the domain) as this is handled inherently by the method.

A simplified spinal cord geometry based on geometric shapes was used for the MPM SCI model, although it is also possible to generate arbitrarily complex geometries from image data. This can be achieved using a simple thresholding algorithm to generate material points based on pixel values that correspond to the relevant materials. While this greatly simplifies the process of creating the point cloud equivalent to the FE mesh, the drawback is that it does not permit the use of coarser or finer elements to facilitate a reduction in the overall number of elements. The underlying grid is regular, this can lead to a surplus of particles that needlessly increase the computational cost. This is mitigated by the ability to specify a material resolution, which fine tunes the placement of particles within a computational

cell. The issue will be solved almost entirely through the implementation of adaptive mesh refinement (AMR), discussed further in Section 6.2.4.1. With this in mind, MPM/MPMICE is highly advantageous in that it facilitates the rapid and straightforward generation of complex geometries, reducing the overall time and effort required to design and run a computational SCI model. As research into SCI progresses, increasingly complex geometries and increased numbers of materials will be required. As this happens the value of the MPM approach in simplifying the generation of the initial point cloud (mesh) will further increase.

The MPMICE method proved to be an effective means of incorporating the FSI between the CSF and the adjacent tissues. It enables full physics (involving strong physical interactions between solid and fluid phases) FSI, with a full Navier–Stokes representation of fluids [134]. The averaged approach means that there is no need to specify conditions on the boundaries between the solid and fluid phases, with their interaction being handled inherently by the method. As is the case with MPM, this is advantageous in simplifying the process of creating the initial problem set-up, in addition to facilitating excellent parallel scalability. MPMICE also allows the inclusion of multiple fluids with different material properties with no additional work required to handle their interaction. This may be useful for any studies that seek to include the effects of bleeding into the subdural space, where blood becomes mixed with the CSF.

### 6.2.3 *Disadvantages*

The absence of inter–material boundary conditions is one of the key strengths of MPM/MPMICE, however; this comes at the cost of a slightly fuzzy boundaries between materials. In FE the boundaries

can be defined exactly on the element faces that divide materials, the boundaries are less clear in the case of the particles used in MPM, although they are still limited to just a few cells. The GIMP shape function (discussed in Chapter 3, Section 3.3.1) spans more than two cells, the effective interface between two adjacent materials is somewhere between the two. This issue becomes decreasingly apparent as the grid resolution is increased as the effect is limited to just a few cells (Section 3.3.1).

Tracking of material boundaries is straightforward in FE as the element faces between two materials in contact are clearly defined. In the case of MPM/MPMICE tracking the position of the material boundaries is more difficult as none of the particles are explicitly defined as being part of a material boundary as this is not required for the method. Calculating the deformation for the whole spinal cord construct, and for the internal deformation of the spinal cord tissue only, required the tracking of the material boundaries. This was achieved using Matlab post processing scripts, which identified the particles of interest on the boundaries based on knowledge of the model geometry and then tracked the movement of these particles using the particle IDs. While suitable post processing scripts overcame this issue, the difficulty and extra processing required for tracking boundaries (especially in the case of complex geometries) is a drawback of MPM/MPMICE compared to FE.

### 6.2.4 *Material Behaviour*

The single term Ogden models accurately captured the hyperelastic behaviour of the spinal cord and dura mater, the linear elastic model was adequate for the impactor and backplate as these components experienced negligible deformation. These models were successfully

implemented in C++ and integrated into the UCF. The particle based approach negated the need to calculate a stiffness matrix via the constitutive models, which simplified the implementation somewhat. The lack of readily available constitutive models within the UCF was a drawback compared to commercial FEM software packages that typically include an extensive range. This reflects the fact that the methodology and software are still being developed.

All materials, including the dura mater, were modelled using isotropic material properties, this is a simplification, in reality the tissues exhibit an anisotropic response to loading. In particular, the dura mater is known to be significantly stronger and more deformable under longitudinal loading than horizontal, likely due to the longitudinal orientation of the fibres within the tissue [96]. The isotropic material properties may also mean that the model is overestimating stress in some regions on the tissue. While the model showed close agreement with experimental results, these experiments only involved a transverse impact. The use of anisotropic constitutive models for the tissues likely increased the compliance as the increased stiffness in the longitudinal direction was not reflected. It is possible to incorporate transverse anisotropy into the MPM model through the use of differently oriented fibres and the mechanical failure of these in response to traumatic loading could also be implemented [122]. This would require substantial development to the isotropic Ogden model used.

Grey and white matter were homogenised into a single material, in reality these are distinct regions of the spinal cord with different material properties, although they are similar [120]. The slight difference in mechanical properties meant that this simplification did not prevent the model from achieving convergence with the experimental results in terms of deformation over time. The difference may have a more significant effect on the internal distribution of stress. There is

evidence showing that grey matter is slightly stiffer than white matter, with greater stresses observed in grey matter during injury, grey matter may be more susceptible to damage than white matter [6, 18, 87, 104, 108]. For these reasons, a more detailed SCI model may seek to include disctinct regions of grey and white matter in the future.

Viscoelastic material properties were not used, future models may benefit from the incorporation of a viscoelastic component to the relative constitutive models. A published study by Sparrey and Keaveny suggest that a single term Ogden model (such as that used here) combined with a 3-term Prony series expansion would be suitable for this [166]. The time from first impact to maximum deformation is of greater interest as this is the time in which the majority of the neurological damage is inflicted and the recoil observed in these experiments does not occur to this extent in vivo, as the bone fragments remain lodged in the spinal canal [7, 11]. However, the omission of viscoelastic material properties suggest that the model may be overestimating the stresses within some regions of the tissue, this may also be due in part to the isotropic material models used. Aligning the model to better meet the recoil portion of the trajectory curves would indicate that the viscoelastic tissue properties were properly reflected in the model, this is a consideration for future iterations of the computational model.

The mechanical properties were based on compression testing of cat spinal cords [91], while the properties of the dura mater were based on tensile testing of bovine dura mater [96]. The properties of the animal tissues will be different to human due to the physiological differences between species. In general there is a lack of available data in the literature on the mechanical properties of the spinal cord, particularly in humans and where traumatic compressive loading is concerned. As such it is difficult the difference between the *ex vivo* animal properties used here and the *ex vivo* or *in vivo* human

properties. Human tissue testing is difficult for ethical and regulatory reasons, limiting the available data for computational modelling. The model would benefit from data for compressive loading at high rates of strain for a large animal such as bovine or porcine. Nonetheless the validation of the model against the experiments suggests that the properties used are reasonably representative of the natural tissue properties.

The behaviour of the CSF was governed according to an equation of state based on a Gibbs function reported by Thomsen and Hartka [164], this was selected as it is readily available within the UCF release. This accurately captured the behaviour of the CSF; however, a limitation of this implementation caused it to fail if the pressure dropeed below 1 atmosphere (101,325 Pa). To work around this the pressure of the computational domain and all the materials had to be increased to prevent the simulation from failing. The lowest stable pressure for the SCI model was found to be 3 atmospheres (303,975 Pa), while this did not seem to significantly affect the deformation and deformation time of the cord construct this is still suboptimal. A future iteration of the model would benefit from the implementation of an equation of state for MPMICE that is stable at 1 atmosphere.

#### 6.2.4.1 *Performance*

The solid material SCI model using MPM was run using 320 cores. Convergence with experimental and FE results was achieved using a grid resolution of $0.5mm^3$. These simulations ran to completion in less than 48 hours, which is a feasible timeframe. The FSI model using MPMICE was run using 400 cores, convergence was achieved using a grid resolution of $0.4mm^3$, these simulations ran to completion in around 96 hours. This is a feasible timeframe, although limitations in the MPMICE implementation resulted in substantial inefficiency,

| Material | Characteristics | Model |
|---|---|---|
| Cord | Isotropic compressible hyperelastic solid | Ogden |
| Dura | Isotropic compressible hyperelastic solid | Ogden Storakers |
| CSF | Almost incompressible Low viscosity fluid | Gibbs function (THWater) |
| Impactor | Isotropic compressible solid | Linear |
| Backplate | Isotropic compressible solid | Linear |
| Air | fluid | Ideal Gas |

Table 21: Mechanical characteristics of spinal cord tissues as defined in the MPM/MPMICE SCI models.

there is significant scope to improve performance and achieve the same results with less computational time and resource.

The introduction of the particles and the interpolation steps between these and the grid nodes means that MPM is more computationally expensive per timestep than FEM. Furthermore, MPM is currently offers only first order accuracy, with second order accuracy or higher being typical of FE methods. To overcome this the resolution must be increased to obtain equivalent accuracy, further increasing computational expense. Performance gains, in terms of the wall clock time required to run the simulations to completion, must result from parallelisation. In this regard MPM compares extremely favourably to FEM, however; there are some issues with the MPM model in its current form that negatively impact efficiency.

The computational domain was rectangular, this could be divided evenly into computational patches by defining a divider in the $x$, $y$, and $z$ directions. It was not possible to include differently sized patches. This was problematic given the geometry of the SCI model. For optimum efficiency it is desirable for the particles to be divided

as evenly as possible across the patches. Due to this restriction on the patch payout a number of the patches saw no particles at throughout the simulation, which resulted in substantial wasted computational resource. This is a restriction of the Uintah implementation of MPM and not the MPM methodology itself, which may theoretically work with any patch layout. It should be noted that this software is still being developed and refined, future iterations will likely improve this situation.

The lack of adaptive mesh refinement (AMR) capability also hinders the performance of the model in its current form. Ideally it would be possible to improve efficiency by using a coarser mesh in the areas of the domain that see little change, such as the backplate and empty spaces, and a finer mesh in the areas of interest, such as the tissues in the impact zone. Presently, an increase in the fineness of the mesh to accommodate one area of the simulation must also increase the fineness in all other areas, including those where it is not needed, resulting in surplus particles and incurring additional computational cost. Again this is a limitation of the Uintah software and not the MPM method itself. The software already contains a working AMR implementation for ICE and an early prototype for AMR for MPM and MPMICE, full AMR capability is under development planned for a future release [153, 154, 156]. Incorporation of AMR in the future, when this functionality becomes available, would improve the efficiency of the model.

The Uintah software contains some capability for applying dynamic load balancing (DLB), while this was investigated it was not utilised for the SCI model. The DLB implementations in Uintah do not vary the patch layout to maintain an even distribution of particles (and therefore computational load), instead the patch layout remains fixed and the balancer attempts to balance the load by assessing the computational cost of each patch and distributing them amongst the cores

in an even manner according to this [156]. This assumes that each core will potentially be handling more than one patch, however, one patch per core proved to be the optimum distribution. Computation time per timestep dropped significantly when the cores were required to handle multiple patches. This approach is likely more effective for ICE simulations, where the fluid materials are more evenly distributed across the domain. For problems involving solid deformations, such as this, an approach whereby patch size and layout was varied to maintain an even distribution of particles (using one core per patch) would be more effective. Again this is not prohibited by MPM or MPMICE and may appear in a future iteration of the software.

| Issue | Cause | Solution |
|---|---|---|
| Fuzzy Boundaries | MPM Inherent | Increase resolution |
| First Order Accurate | MPM Inherent | Increase resolution |
| Increased computational expense | MPM Inherent | More parallel cores |
| Some patches underfilled/empty | UCF Implementation | Further UCF Development |
| No AMR for MPM | UCF Implementation | Further UCF Development |
| Suboptimal load balancing | UCF Implementation | Further UCF Development |

Table 22: Performance Issues affecting the MPMICE SCI model

### 6.2.5 *Explicit vs Implicit*

MPM and MPMICE use an explicit time integration scheme, the authors note that these version of the algorithms is best suited for high-rate-of-deformation problems and dynamic loading problems. This is suitable for the SCI model, however; problems involving quasi-

static loading would benefit more from an implicit integration scheme which could better handle the propagation of stress waves through solid materials [134]. Implicit versions of MPM/MPMICE have been developed and implemented in the UCF [124, 156]. This involves an alternative MPM formulation whereby the computational grid is not reset at the end of each timestep, though interpolation between the particles and the grid nodes still occurs. This improves the fidelity of the stress fields for quasi static low-rate-of-deformation problems, though this makes the approach more prone to mesh tangling and hence less suited for high-rate-of-deformation problems. Nevertheless, the inclusion of the particles, which can move between cells, makes the implicit MPM more resilient to mesh tangling than equivalent FE formulations [124]. As the mesh is not reset at the end of each timestep it is necessary to calculate the stiffness matrix via the solid material constitutive models.

## 6.3 PIA MATER

A future iteration of this SCI model would seek to include the pia mater as a separate material to the spinal cord. In order to include the pia in a computational model it is necessary to determine the mechanical properties and calculate the material constants for the relevant constitutive material model. The lack of available data in the literature regarding the mechanical properties of pia mater prompted this investigation using tensile testing of bovine spinal pia mater. The methodology was adapted from a study of bovine cranial pia mater [165]. The methodology was partially successful, however the a number of refinements are required to the approach used in this preliminary study in order to more fully characterise the pia.

The method of adhering the exposed surface of the pia to a paper substrate was effective in allowing the grey and white matter to be peeled away leaving an isolated sheet of pia mater for testing. This also allowed the sample to be cut to shape using scissors. This method still required patience and practice to avoid damaging the delicate tissue. The pia was observed to be a thin, delicate transparent tissue with visible vasculature. The pia was noticeably stiffer than the soft and viscous grey and white matter contained within.

The process of transferring the pia samples, adhered to paper via the moisture present in the tissue, to the clamps whilst keeping them intact was extremely difficult. The initial method involved peeling the paper away once the sample was clamped in place. It was hoped that the paper, weakened by the moisture, would peel away easily, in reality removing the paper was very difficult and a number of samples were damaged beyond use in the process. This issue was overcome by switching the paper with plastic, which could be more easily peeled back from the tissue and then cut using scissors leaving the sample in the clamps.

All of the samples broke at the clamps, as a result it was not possible to test the samples to failure. This issue was largely due to the clamps themselves, the 10 N clamps used were the smallest available clamps for the machine but were nevertheless too heavy. This issue is indicative of the clamps being too tight, however when the clamps were loosened the samples slid out, it was not possible to find a point whereby the samples would neither slide nor break at the clamps. Two refinements to the method were suggested to address this. Firstly, longer samples should be cut to allow more of the sample to be clamped and reduce the risk of sliding. Secondly, the clamps should be padded to reduce and spread the pressure on the samples from the clamps whilst maintaining sufficient grip. Smaller, lighter, padded custom made clamps may improve the situation.

The pia exhibited hyperelastic material behaviour. The mean stress vs strain was compared to the published results for bovine cranial pia mater [165]. For the portion of the curve available no statistically significant difference was observed between the two. The standard deviation was large, this reflects the difficulty in isolating the samples and positioning them in the clamps. It is hoped that the refinements to the method discussed here combined with further testing using a larger number of samples will increase the confidence in the results.

## 6.4 VALIDATION

### 6.4.1 *Validation against FE*

The *bare cord* MPM models were in close agreement with the FE models for each of the three pellet types. For all pellet types the MPM model gave a slightly lower maximum deformation and a slightly lower time to maximum deformation. This small difference may be attributed to differences in the way the Ogden constitutive models used for the spinal cord tissue were implemented, a different implementation for the particle based MPM method was required compared to that for FE. The difference may also be in part due to the fuzzy boundaries in MPM, discussed in Section 6.2.3. It should also be noted that MPM is first order accurate and the FE method used for the comparison is second order accurate.

The pellet trajectories for the MPM cord/dura models gave a slightly higher maximum deformation compared to the FE model. This difference was most likely due the use of an Ogden model for the dura mater rather than the linear elastic model used in the FE model. The hyperelasticity enabled by the Ogden model would have allowed greater deformation of the dura and in turn the spinal cord construct.

The impact of this difference would have been further affected by the varying areas of the pellet faces. The dura mater is known to exhibit hyperelastic material behaviour and the Ogden model appears well suited to serve as the constitutive model.

The FE *cord/dura* trajectories each show a dip in deformation, this is also present in the *bare cord* trajectories but is difficult to see. This is due to a small gap between the cord and backplate in the FE model, this not present in the MPM model where the cord construct was positioned directly in front of the backplate. At a certain point, following the initial impact, the inertia is overcome and the whole construct then begins to move backwards. This also accounts for some of the differences in maximum deformation and time to maximum deformation between the MPM and FE results.

### 6.4.2 *Validation Against Experimental*

The MPM *bare cord* computational model overestimated the maximum deformation compared to the experimental model for all pellet types, this was also the case with the FE model. This may be the due to an increased compliance in the bovine spinal cord tissue following repeated impacts. Each impact will have damaged the extracellular matrix of the tissues, weakening the structure and reducing the ability of the tissue to withstand further impacts. In the experiments the specimens were first tested with the dura and CSF present, then with the CSF drained but the dura mater remaining, then with the dura mater removed also. As a result the specimens would have sustained between 6 and 8 impacts prior to the bare cord impact tests. Inter-animal variance and tissue degradation due to the increased time between animal slaughter and testing may have also contributed to this discrepancy.

The *cord/dura* MPM model showed a close agreement with both the experimental pellet trajectory curves, the overestimation of the maximum deformation observed in the bare cord model was not present here. In the case of Pellet 1 the maximum deformation aligned almost exactly with experimental mean, and very slightly higher than that of the FE model. For Pellet 2, the MPM model slightly overestimated the maximum deformation, although this difference was not statistically significant. In the case of Pellet 3, the peaks for the MPM model aligned very closely with the experimental mean. From this validation it is possible to state that the solid MPM model is accurately capturing the mechanical response of the spinal cord tissue and that the results are comparable to that achievable using FE modelling.

The recoil time, the time from the point of maximum deformation to the point at which the pellet leaves contact with the cord surface, is shorter for both the MPM result when compared to the experimental results for all three pellet types and all three configurations. Again, this was also the case for the FE model. The longer recoil time observed in the experiments is most likely due to the viscoelastic properties of the spinal cord tissues [14]. Some of the energy absorbed by the cord from the pellet would not have been transferred back to the pellet upon recoil due to the viscoelasticity, reducing the pellet recoil velocity. This did not occur in the computational models as viscoelastic material behaviour was omitted. The also accounts for the faster time from first impact to maximum deformation observed in the computational models relative to the experimental model.

## 6.5 BENEFICIARIES

The beneficiaries of this research will include academics, both those with an interest in SCI biomechanics and those with an interest in

high-end computational modelling. The clinical community and patient groups will also benefit from an increased understanding of the spinal cord biomechanics, which will hopefully be useful in investigating SCI treatment strategies. Additionally, animal experimentation may be reduced, by refining existing models, replacing some experiments computationally, and providing information to guide the better design of animal tests in the future.

# 7

# CONCLUSIONS AND FUTURE WORK

## 7.1 CONCLUSIONS

An burst fracture Spinal Cord Injury (SCI) simulation was created using the Material Point Method (MPM) and the Material Point Method Implicit Continuous Fluid Eulerian (MPMICE) multi phase variant. This model was validated against existing experimental and Finite Element (FE) studies.The MPMICE model produced equivalent results to FE, and achieved convergence with experimental results. Hyperelastic Ogden models were implemented for MPM and these accurately captured the hyperelastic material behaviour of the spinal tissues. MPM/MPMICE was demonstrated to be a viable alternative to FE for modelling SCI biomechanics. MPMICE demonstrates excellent parallel scalability, 560 parallel processors were used for the model and the potential exists to use many more (up to 100,000 and beyond). This parallel scalabilty is the key advantage over FE based approaches and facilitates the use of additional computational resource, enabling the development of more complex and detailed models going forward. MPMICE is less accurate (first order) than FE (typically second order), this can be overcome by increasing the grid resolution. The subsequent increase in computational requirement can be overcome by using more parallel processors. MPMICE is also more computationally expensive per timestep than FE, again this can be overcome by using more parallel processors to shorten the real time required to run the simulations. Future work will focus on taking the model further, incorporating greater anatomical detail, more types of

injury, and utilising greater numbers of parallel processors to reduce computation time. To summarize the key conclusions of this study are:

- The complexity of FE models of SCI is limited by poor scalability for parallel computing, mesh entanglement, and difficulties in incorporating the FSI between the CSF and the surrounding tissues
- MPMICE is a newer alternative method to FE that can overcome these limitations
- MPM is a viable alternative to FE for computationally modelling SCI where only solid phases are included
- MPMICE is a viable alternative to FE for computationally modelling SCI including the FSI
- MPM/MPMICE scale extremely well for parallel computation, offering greatly superior parallel scalability than FE
- MPM/MPMICE is a relatively new technique compared to FE, which is well established with a variety of software packages available
- Aspects the UCF implementation of MPM/MPMICE, which is the most sophisticated implementation currently available, restrict the performance gains of increased parallelisation. However, there is no reason these could not be overcome with further development of the software.
- MPM/MPMICE has the potential to enable more complex, detailed, anatomically relevant models of SCI in the future, overcoming the limitations of FE.

## 7.2 FUTURE WORK

Having established the model in its current form, future work will focus primarily on creating a parametric model and simulating additional injury types to the burst fracture. The creation of the parametric model is discussed in Section 7.2.1. The addition of spinal vertebrae to the model is necessary to simulate more injury types such as fracture dislocations, which is discussed in Section 7.2.2. For this reason it is logical that the parametric model be completed prior to adapting the model to simulate other kinds of injury to the burst fracture.

### 7.2.1 *Parametric Model*

Having established the model in its current form, a parametric model will be created based on human measurements, with greater anatomical detail, and at higher resolution than the current model. This is a logical next step in developing the model, as the characterisation of SCI continues increasing levels of detail will be required from computational models. The increase in resolution is necessary to capture smaller details, such as the thin meningeal layers. The initial model geometry will be generated from computerised tomography (CT) scan image data, suitable images are freely available from the Visible Human Project (United States National Library of Medicine, Bethesda, MD, USA). While the process of generating a material point cloud from an image is straightforward, some preprocessing will likely be required to limit the number of visible materials in the image to those to be included in the model. This preprocessing will be performed using commercially available software; Matlab (The MathWorks Inc., Natick, MA, USA), Photoshop (Adobe Systems Inc.,

Mountain View, CA, USA, or Simpleware (Synopsys, Inc., Mountain View, CA, USA) would all be suitable.

### 7.2.2 *Other Injury Types*

With the burst fracture injury simulation completed, the model can now be altered to simulate other types of injury also. In particular fracture dislocation injuries, both anterior and lateral, could be simulated. This would require the addition of at least two simulated vertebrae instead of the impactor, the upper section will move shearing the cord between the simulated vertebrae.

### 7.2.3 *Further Validation*

Further validation will be carried out against published experimental and computational SCI studies. More types of injury will be simulated in addition to the burst fracture, these will include other common types of spinal cord injury, most notably fracture dislocation injuries. In particular the MPMICE SCI model may be validated against an experimental rat contusion injury model reported by Lam et al. and the complimentary FE study reported by Russell et al., which also simulated dislocation injuries in addition to contusion [54, 167]. Altering the MPMICE model geometry to rat spinal dimensions is straightforward as the dimensions are known and the point cloud generation in MPM is simple. Russell et al. used a hyperelastic Ogden for the cord and dura and reported the values used, these can be used to easily alter parameters of the MPMICE to reflect the rat tissue mechanical properties.

A porcine animal model using Yucatan miniature pigs has been reported by Lee et al., this study also analysed contusion injuries

induced via a weight drop device [168]. As this is one of few animal models using larger animals it would be worthwhile to recreate this computationally using MPMICE. Again, the generation of the porcine spinal cord geometry is straightforward, however; the mechanical properties are not fully characterised. Compression testing data for porcine spinal white matter is available [166], the properties of the porcine grey matter are likely to be similar to this. Tensile testing results for porcine dura mater have been reported by Mazgajczyk et al., which may be used to calculate appropriate Ogden constants for porcine dura [169]. The benefits of creating comparable models to these studies, beyond further demonstrating the viability of the method, is that it will allow correlations to be drawn between the internal stress and strain fields observed in the MPMICE model to the biological tissue damage observed in the animal studies [54, 167, 168]. This is useful in establishing damage thresholds and increasing the characterisation and understanding of the injuries.

### 7.2.4 *Distinct Tissues*

Increasing the level of anatomical detail included in the SCI model will require the inclusion of some distinct tissues not presently included. The spinal cord tissue will be broken down into distinct regions of grey and white matter. These tissues have slightly different material properties [120], while both would still be represented with an Ogden model, the parameters will be adjusted to account for this difference. A parametric study will may also be required to quantify the effects of this difference and to fine tune the material parameters. The pia mater will also be incorporated as a distinct tissue using an Ogden model. The material parameters will be based on mechanical characterisation of bovine tissue described in Chapter 5.

### 7.2.5 *Constitutive Model Development*

The model in its current form uses isotropic, hyperelastic, non-viscoelastic constitutive models for the spinal cord tissues, these have been suitable thus far. However, as the model is further developed it would be beneficial to extend these models to more closely capture the biomechanical properties of the tissues and improve anatomical accuracy. Viscoelasticity will be incorporated into the Ogden constitutive models using a Prony series expansion, the literature suggests that this will be suitable to incorperate the viscoelastic response [19, 120]. Transverse anisotropy can be achieved through the integration of fibre mechanics into the constitutive models, a method for implementing this has been reported by [126], where layers of differently ordered fibres were used to achieve an anisotropic material response to transverse penetrating impacts [122, 126]. The dura mater is a tissue containing many longitudinal fibres and exhibits anisotropic properties [95, 96, 169]. The integration of fibres into the constitutive models would also facilitate dynamic fibre failures, whereby fibres break when a certain threshold is exceeded during the simulation and subsequently cease bearing load. Finally, the single term Ogden model implementation may be expanded into multi term Ogden model with up to 9 terms. While a single term Ogden model was suitable for the model in its current form a multi term model is capable of capturing more detail in the stress–strain response of the materials [149, 159]. This may not be required and would only be developed based on the needs of a specific study.

### 7.2.6 *Characterisation of Spinal Pia Mater*

The preliminary study, which attempted to characterize the material properties of the pia via tensile testing, was partially successful but the refinements to the method discussed in Chapter 6 Section 6.3 are required to avoid the samples breaking in the clamps and to reduce the variability between samples. Using the refined method it will be necessary to test a greater number of samples, and to test subsets of samples in both the horizontal and longitudinal directions. The fibre orientation in the tissue likely means that the mechanical response differs based on the direction of loading, and this the fibre structure is likely to be different to that of cranial pia mater. Following further testing using the refined method the Ogden material constants can once again be calculated in the same manner as before, and these may be used to include the pia mater as a distinct tissue in the SCI model.

# APPENDICES

## A.1 INPUT FILES

### A.1.1 *Bare Cord P1*

```
%[caption=dev2.9.1_ogadina.ups]
<?xml version='1.0' encoding='ISO-8859-1' ?>

<!-- with dura -->
<!-- with additional plane of symmetry -->
<!-- non linear ogden parameters for dura -->
<!-- adina based ogden model -->

<Uintah_specification>

  <SimulationComponent type = "mpm"/>

  <Time>
    <maxTime>6e-3</maxTime>
    <initTime>0</initTime>
    <delt_min>1.0e-12</delt_min>
    <delt_max>100.0e-3</delt_max>
    <delt_init>100.0e-3</delt_init>
    <timestep_multiplier>0.1</timestep_multiplier>
  </Time>

  <Grid>
    <BoundaryConditions>
      <Face side = "x-">
        <BCType id = "all" var = "Neumann" label = "Velocity">
          <value> [0.0,0.0,0.0] </value>
        </BCType>
      </Face>
      <Face side = "x+">
        <BCType id = "all" var = "symmetry" label = "Symmetric"> </BCType>
      </Face>
      <Face side = "y-">
        <BCType id = "all" var = "Dirichlet" label = "Velocity">
          <value> [0.0,0.0,0.0] </value>
        </BCType>
      </Face>
      <Face side = "y+">
        <BCType id = "all" var = "Neumann" label = "Velocity">
          <value> [0.0,0.0,0.0] </value>
        </BCType>
      </Face>
      <Face side = "z-">
        <BCType id = "all" var = "Dirichlet" label = "Velocity">
          <value> [0.0,0.0,0.0] </value>
        </BCType>
      </Face>
      <Face side = "z+">
```

```
        <BCType id = "all" var = "symmetry" label = "Symmetric"> </BCType>
      </Face>
    </BoundaryConditions>

    <Level>
      <Box label = "Domain">
        <lower>[0,0,0]</lower>
        <upper>[13e-3,45e-3,40e-3]</upper>
        <extraCells>[1,1,1]</extraCells>
        <patches>[4,1,80]</patches>
        <!--<patches>[1,1,16]</patches>-->
      </Box>
      <spacing>[0.5e-3, 0.5e-3, 0.5e-3]</spacing>
    </Level>
  </Grid>

  <DataArchiver>
    <filebase>/nobackup/mnsg/uintah_dev5_1/output/dev2.9.1_ogadina.uda</
        filebase>
    <!--<filebase>/usr/not-backed-up/mnsg-usr/UCF_Output/dev2.9.1
        _ogadina.uda</filebase>-->
    <outputInterval>0.05e-3</outputInterval>
      <save label = "p.particleID"/>
      <save label = "p.x"/>
      <save label = "p.velocity"/>
      <save label = "p.stress"/>
      <save label = "p.color"/>
      <checkpoint cycle = "2" interval = "0.0001"/>
  </DataArchiver>

  <MPM>
    <time_integrator>explicit</time_integrator>
    <interpolator>gimp</interpolator>
    <withColor>true</withColor>
    <DoPressureStabilization>false</DoPressureStabilization>
  </MPM>

  <PhysicalConstants>
    <gravity>[0,0,0]</gravity>
  </PhysicalConstants>

  <MaterialProperties>
    <MPM>
      <material name="cord_tissue">
        <!--density of SC tissue from Persson-->
        <density>1050</density>
        <!--thermal conductivity for water at 25 celcius is 0.58 W/(m.K)-->
        <thermal_conductivity>1</thermal_conductivity>
        <!--specific heat capacity for water is 4.1855 J/(g.K)-->
        <specific_heat>1</specific_heat>

        <constitutive_model type = "comp_ogden_standard">
          <bulk_modulus>45e3</bulk_modulus>
          <mu_1>2000</mu_1>
          <alpha_1>9</alpha_1>
          <equation_of_state type="default_hyper"></equation_of_state>
        </constitutive_model>

        <geom_object>
          <!--elliptical cord, dimensions from Persson et al. 2011 -->
```

```
        <ellipsoid label = "cord">
          <origin> [13e-3, 6.6e-3, 7e-3] </origin>
          <rx> 7.5e-3 </rx>
          <ry> 5e-3 </ry>
          <rz> 1e12 </rz><!--effectively creates an elliptical tube within the
               domain, 140mm long-->
        </ellipsoid>

        <res>[2,2,2]</res>
        <velocity>[0.0,0.0,0.0]</velocity>
        <temperature>310.15</temperature>
        <color>20</color>
      </geom_object>
    </material>

    <material name="impactor">
      <!--density of tufnol composite, from Persson et al. 2011-->
      <density>1360</density>
      <thermal_conductivity>1</thermal_conductivity>
      <specific_heat>1</specific_heat>

      <constitutive_model type = "comp_linear">
        <E>6.5e9</E>
        <v>0.3</v>
      </constitutive_model>

      <geom_object>
        <!-- dimensions of pellet 1, from Persson et al. 2011-->
        <union>
        <cylinder>
          <bottom> [13e-3, 13e-3, 40e-3] </bottom>
          <top> [13e-3, 43e-3, 40e-3] </top>
          <radius> 6.00e-3 </radius>
        </cylinder>
        <cylinder>
          <bottom> [13e-3, 13e-3, 40e-3] </bottom>
          <top> [13e-3, 23e-3, 40e-3] </top>
          <radius> 10.00e-3 </radius>
        </cylinder>
        </union>

        <res>[2,2,2]</res>
        <!-- Speed from from Persson et al. 2011-->
        <velocity>[0,-4.5,0]</velocity>
        <temperature>310.15</temperature>
        <color>40</color>
      </geom_object>
    </material>

    <material name="steel_posterior_plate">
      <!--From CP thesis-->
      <density>8000</density>
      <!--thermal conductivity for 1% carbon steel at 25 celcius is 43 W/(m.K)
          -->
      <thermal_conductivity>1</thermal_conductivity>
      <!--specific heat capacity for steel is 0.466 J/(g.K)-->
      <specific_heat>1</specific_heat>

      <constitutive_model type = "comp_linear">
        <E>193e9</E>
```

```
      <v>0.3</v>
    </constitutive_model>

    <geom_object>
      <box>
        <min>[0,0,0]</min>
        <max>[13e-3,1e-3,80e-3]</max>
      </box>
      <res>[2,2,2]</res>
      <velocity>[0.0,0.0,0.0]</velocity>
      <temperature>310.15</temperature>
      <color>60</color>
    </geom_object>
  </material>

  <material name = "dura_mater">
    <density>1000</density>
    <thermal_conductivity>1</thermal_conductivity>
    <specific_heat>1</specific_heat>

    <!--
    <constitutive_model type = "comp_linear">
      <E>80e6</E>
      <v>0.49</v>
    </constitutive_model>


    <constitutive_model type = "comp_ogden_storakers">
      <PR>0.49</PR>
      <mu_1>322e3</mu_1>
      <alpha_1>19</alpha_1>
    </constitutive_model>

    -->

    <constitutive_model type = "comp_ogden_standard">
      <bulk_modulus>152950e3</bulk_modulus>
      <mu_1>322e3</mu_1>
      <alpha_1>19</alpha_1>
      <equation_of_state type="default_hyper"></equation_of_state>
    </constitutive_model>


    <geom_object>
      <difference>
        <ellipsoid label = "outer_dura">
          <origin> [13e-3, 6.6e-3, 7e-3] </origin>
          <rx> 9.5e-3 </rx>
          <ry> 5.6e-3 </ry>
          <rz> 1e12 </rz>
        </ellipsoid>
        <ellipsoid label = "inner_dura">
          <origin> [13e-3, 6.6e-3, 7e-3] </origin>
          <rx> 9e-3 </rx>
          <ry> 5.1e-3 </ry>
          <rz> 1e12 </rz>
        </ellipsoid>
      </difference>

      <res>[2,2,2]</res>
```

```
          <velocity>[0,0,0]</velocity>
          <temperature>310.15</temperature>
          <color>80</color>
        </geom_object>
      </material>

    <contact>
      <type>friction</type>
      <materials>[0,1,2,3]</materials>
      <mu>0.0</mu>
    </contact>

    </MPM>
  </MaterialProperties>
</Uintah_specification>
```

### A.1.2 *Bare Cord P2*

```
%[caption=cord_v1_p2.ups]
<?xml version='1.0' encoding='ISO-8859-1' ?>

<Uintah_specification>

  <Meta>
    <title>bare cord model, pellet 2</title>
  </Meta>

  <SimulationComponent type = "mpm"/>

  <Time>
    <maxTime>     6e-3   </maxTime>
    <initTime>     0    </initTime>
    <delt_min>     1.0e-12   </delt_min>
    <delt_max>     100.0e-3  </delt_max>
    <delt_init>    100.0e-3  </delt_init>
    <timestep_multiplier>  0.1    </timestep_multiplier>
  </Time>

  <MPM>
    <time_integrator>   explicit  </time_integrator>
    <interpolator>    gimp   </interpolator>
    <withColor>     true   </withColor>
    <DoPressureStabilization> false   </DoPressureStabilization>
  </MPM>

  <PhysicalConstants>
    <gravity>     [0, 0, 0]  </gravity>
  </PhysicalConstants>

  <MaterialProperties>
    <MPM>
      <material name="cord_tissue">

        <constitutive_model type = "comp_ogden_standard">
          <bulk_modulus> 45e3 </bulk_modulus>
          <mu_1>   2000 </mu_1>
          <alpha_1>  9  </alpha_1>
          <equation_of_state type="default_hyper"></equation_of_state>
          <useModifiedEOS> true  </useModifiedEOS>
        </constitutive_model>
```

```
    <density>    1050 </density>
    <thermal_conductivity> 1  </thermal_conductivity>
    <specific_heat>   1  </specific_heat>

    <geom_object>
      <ellipsoid label = "cord">
        <origin> [13e-3, 6e-3, 7e-3]  </origin>
        <rx>  7.5e-3     </rx>
        <ry>  5e-3     </ry>
        <rz>  1e12     </rz>
      </ellipsoid>

      <res>   [2, 2, 2]    </res>
      <velocity>  [0.0, 0.0, 0.0]   </velocity>
      <temperature> 310.15     </temperature>
      <color>   20      </color>
    </geom_object>
  </material>

  <material name="tufnol">

    <constitutive_model type = "comp_linear">
      <E>     6.5e9  </E>
      <v>     0.3   </v>
      <equation_of_state type="default_hyper"></equation_of_state>
      <useModifiedEOS> true  </useModifiedEOS>
    </constitutive_model>

    <density>    1360 </density>
    <thermal_conductivity> 1  </thermal_conductivity>
    <specific_heat>   1  </specific_heat>

    <geom_object>
      <union>
        <cylinder label = "impactor_tail">
          <bottom> [13e-3, 12e-3, 40e-3] </bottom>
          <top>  [13e-3, 34e-3, 40e-3] </top>
          <radius> 7.00e-3     </radius>
        </cylinder>
        <cylinder label = "impactor_mid">
          <bottom> [13e-3, 24e-3, 40e-3] </bottom>
          <top>  [13e-3, 34e-3, 40e-3] </top>
          <radius> 10.00e-3    </radius>
        </cylinder>
        <cylinder label = "impactor_head">
          <bottom> [13e-3, 34e-3, 40e-3] </bottom>
          <top>  [13e-3, 37e-3, 40e-3] </top>
          <radius> 6.00e-3     </radius>
        </cylinder>
      </union>

      <res>   [1, 1, 1]     </res>
      <velocity>  [0, -4.5, 0]    </velocity>
      <temperature> 310.15     </temperature>
      <color>   40      </color>
    </geom_object>
  </material>

  <material name="steel">
```

```
      <constitutive_model type = "comp_linear">
        <E>     193e9  </E>
        <v>     0.3   </v>
        <equation_of_state type="default_hyper"></equation_of_state>
        <useModifiedEOS> true  </useModifiedEOS>
      </constitutive_model>

      <density>    8000 </density>
      <thermal_conductivity> 1  </thermal_conductivity>
      <specific_heat>   1  </specific_heat>

      <geom_object>
        <box label = "backplate">
          <min>   [0, 0, 0]    </min>
          <max>   [13e-3, 1e-3, 80e-3] </max>
        </box>
        <res>    [1, 1, 1]    </res>
        <velocity>   [0.0, 0.0, 0.0]   </velocity>
        <temperature>  310.15     </temperature>
        <color>    60      </color>
      </geom_object>
    </material>

    <contact>
      <type>   friction  </type>
      <materials>  [0, 1, 2]  </materials>
      <mu>   0.0    </mu>
    </contact>

  </MPM>
</MaterialProperties>

<Grid>
  <BoundaryConditions>
    <Face side = "x-">
      <BCType id = "all" var = "Neumann" label = "Velocity">
        <value> [0, 0, 0] </value>
      </BCType>
    </Face>
    <Face side = "x+">
      <BCType id = "all" var = "symmetry" label = "Symmetric"> </BCType>
    </Face>
    <Face side = "y-">
      <BCType id = "all" var = "Dirichlet" label = "Velocity">
        <value> [0, 0, 0] </value>
      </BCType>
    </Face>
    <Face side = "y+">
      <BCType id = "all" var = "Neumann" label = "Velocity">
        <value> [0, 0, 0] </value>
      </BCType>
    </Face>
    <Face side = "z-">
      <BCType id = "all" var = "Dirichlet" label = "Velocity">
        <value> [0, 0, 0] </value>
      </BCType>
    </Face>
    <Face side = "z+">
      <BCType id = "all" var = "symmetry" label = "Symmetric"> </BCType>
```

```
      </Face>
    </BoundaryConditions>

    <Level>
      <Box label = "Domain">
      <lower>   [0, 0, 0]     </lower>
      <upper>   [13e-3, 45e-3, 40e-3]  </upper>
      <extraCells> [1, 1, 1]     </extraCells>
      <patches>  [8, 1, 40]     </patches>
      <!--<patches>  [1, 1, 16]    </patches>-->
      </Box>
      <spacing>  [0.5e-3, 0.5e-3, 0.5e-3]  </spacing>
    </Level>
  </Grid>

  <DataArchiver>
    <filebase>/nobackup/mnsg/uintah_dev5_1/output/cord_v1_p2.uda</
        filebase>
    <!--<filebase>/usr/not-backed-up/UCF_output/cord_v1_p2.uda</filebase
        >-->
    <outputInterval>0.05e-3</outputInterval>
      <save label = "p.particleID"/>
      <save label = "p.x"/>
      <save label = "p.velocity"/>
      <save label = "p.stress"/>
      <save label = "p.color"/>
      <checkpoint cycle = "2" interval = "0.0001"/>
  </DataArchiver>

</Uintah_specification>
```

### A.1.3 *Bare Cord P3*

```
%[caption=cord_v1_p3.ups]
<?xml version='1.0' encoding='ISO-8859-1' ?>

<Uintah_specification>

  <Meta>
    <title>bare cord model, pellet 3</title>
  </Meta>

  <SimulationComponent type = "mpm"/>

  <Time>
    <maxTime>     6e-3   </maxTime>
    <initTime>     0    </initTime>
    <delt_min>     1.0e-12   </delt_min>
    <delt_max>     100.0e-3  </delt_max>
    <delt_init>    100.0e-3  </delt_init>
    <timestep_multiplier>  0.1    </timestep_multiplier>
  </Time>

  <MPM>
    <time_integrator>   explicit  </time_integrator>
    <interpolator>    gimp   </interpolator>
    <withColor>     true   </withColor>
    <DoPressureStabilization> false   </DoPressureStabilization>
  </MPM>
```

```
<PhysicalConstants>
  <gravity>     [0, 0, 0]  </gravity>
</PhysicalConstants>

<MaterialProperties>
  <MPM>
    <material name="cord_tissue">

      <constitutive_model type = "comp_ogden_standard">
        <bulk_modulus> 45e3 </bulk_modulus>
        <mu_1>   2000 </mu_1>
        <alpha_1>  9  </alpha_1>
        <equation_of_state type="default_hyper"></equation_of_state>
        <useModifiedEOS> true  </useModifiedEOS>
      </constitutive_model>

      <density>    1050 </density>
      <thermal_conductivity> 1  </thermal_conductivity>
      <specific_heat>   1  </specific_heat>

      <geom_object>
        <ellipsoid label = "cord">
          <origin> [13e-3, 6e-3, 7e-3]  </origin>
          <rx>  7.5e-3     </rx>
          <ry>  5e-3     </ry>
          <rz>  1e12     </rz>
        </ellipsoid>

        <res>   [2, 2, 2]    </res>
        <velocity>  [0.0, 0.0, 0.0]   </velocity>
        <temperature> 310.15     </temperature>
        <color>   20      </color>
      </geom_object>
    </material>

    <material name="tufnol">

      <constitutive_model type = "comp_linear">
        <E>     6.5e9  </E>
        <v>     0.3   </v>
        <equation_of_state type="default_hyper"></equation_of_state>
        <useModifiedEOS> true  </useModifiedEOS>
      </constitutive_model>

      <density>    1360 </density>
      <thermal_conductivity> 1  </thermal_conductivity>
      <specific_heat>   1  </specific_heat>

      <geom_object>
        <union>
          <cylinder label = "impactor_tail">
            <bottom> [13e-3, 12e-3, 40e-3] </bottom>
            <top>  [13e-3, 27e-3, 40e-3] </top>
            <radius> 5.00e-3     </radius>
          </cylinder>
          <cylinder label = "impactor_mid">
            <bottom> [13e-3, 27e-3, 40e-3] </bottom>
            <top>  [13e-3, 39e-3, 40e-3] </top>
            <radius> 10.00e-3    </radius>
          </cylinder>
```

```
            <cylinder label = "impactor_head">
              <bottom> [13e-3, 39e-3, 40e-3] </bottom>
              <top>  [13e-3, 42e-3, 40e-3] </top>
              <radius> 6.00e-3     </radius>
            </cylinder>
          </union>

          <res>   [1, 1, 1]    </res>
          <velocity>  [0, -4.5, 0]    </velocity>
          <temperature> 310.15     </temperature>
          <color>   40      </color>
        </geom_object>
      </material>

      <material name="steel">

        <constitutive_model type = "comp_linear">
          <E>     193e9  </E>
          <v>     0.3   </v>
          <equation_of_state type="default_hyper"></equation_of_state>
          <useModifiedEOS> true  </useModifiedEOS>
        </constitutive_model>

        <density>    8000 </density>
        <thermal_conductivity> 1  </thermal_conductivity>
        <specific_heat>   1  </specific_heat>

        <geom_object>
          <box label = "backplate">
            <min>   [0, 0, 0]    </min>
            <max>   [13e-3, 1e-3, 80e-3] </max>
          </box>
          <res>    [1, 1, 1]    </res>
          <velocity>   [0.0, 0.0, 0.0]   </velocity>
          <temperature>  310.15     </temperature>
          <color>    60      </color>
        </geom_object>
      </material>

      <contact>
        <type>   friction  </type>
        <materials>  [0, 1, 2]  </materials>
        <mu>   0.0    </mu>
      </contact>

    </MPM>
  </MaterialProperties>

  <Grid>
    <BoundaryConditions>
      <Face side = "x-">
        <BCType id = "all" var = "Neumann" label = "Velocity">
          <value> [0, 0, 0] </value>
        </BCType>
      </Face>
      <Face side = "x+">
        <BCType id = "all" var = "symmetry" label = "Symmetric"> </BCType>
      </Face>
      <Face side = "y-">
        <BCType id = "all" var = "Dirichlet" label = "Velocity">
```

```
            <value> [0, 0, 0] </value>
          </BCType>
        </Face>
        <Face side = "y+">
          <BCType id = "all" var = "Neumann" label = "Velocity">
            <value> [0, 0, 0] </value>
          </BCType>
        </Face>
        <Face side = "z-">
          <BCType id = "all" var = "Dirichlet" label = "Velocity">
            <value> [0, 0, 0] </value>
          </BCType>
        </Face>
        <Face side = "z+">
          <BCType id = "all" var = "symmetry" label = "Symmetric"> </BCType>
        </Face>
      </BoundaryConditions>

      <Level>
        <Box label = "Domain">
        <lower>   [0, 0, 0]     </lower>
        <upper>   [13e-3, 45e-3, 40e-3]  </upper>
        <extraCells> [1, 1, 1]     </extraCells>
        <patches>  [8, 1, 40]     </patches>
        <!--<patches>  [1, 1, 16]    </patches>-->
        </Box>
        <spacing>  [0.5e-3, 0.5e-3, 0.5e-3]  </spacing>
      </Level>
    </Grid>

    <DataArchiver>
      <filebase>/nobackup/mnsg/uintah_dev5_1/output/cord_v1_p3.uda</
          filebase>
      <!--<filebase>/usr/not-backed-up/UCF_output/cord_v1_p3.uda</filebase
          >-->
      <outputInterval>0.05e-3</outputInterval>
        <save label = "p.particleID"/>
        <save label = "p.x"/>
        <save label = "p.velocity"/>
        <save label = "p.stress"/>
        <save label = "p.color"/>
        <checkpoint cycle = "2" interval = "0.0001"/>
    </DataArchiver>

</Uintah_specification>
```

### A.1.4 *Cord/Dura P1*

```
%[caption=dura_v3_pro.4.ups]
<?xml version='1.0' encoding='ISO-8859-1' ?>

<!-- with dura -->
<!-- with additional plane of symmetry -->
<!-- non linear ogden parameters for dura -->
<!-- linear to rigid -->
<!-- dura back to ogden_storakers from ogden_standard -->
<!-- reduced PR of dura to .3 -->
<!-- increased PR of dura to .4 -->

<Uintah_specification>
```

```
<SimulationComponent type = "mpm"/>

<Time>
  <maxTime>7e-3</maxTime>
  <initTime>0</initTime>
  <delt_min>1.0e-12</delt_min>
  <delt_max>100.0e-3</delt_max>
  <delt_init>100.0e-3</delt_init>
  <timestep_multiplier>0.1</timestep_multiplier>
</Time>

<Grid>
  <BoundaryConditions>
    <Face side = "x-">
      <BCType id = "all" var = "Neumann" label = "Velocity">
        <value> [0.0,0.0,0.0] </value>
      </BCType>
    </Face>
    <Face side = "x+">
      <BCType id = "all" var = "symmetry" label = "Symmetric"> </BCType>
    </Face>
    <Face side = "y-">
      <BCType id = "all" var = "Dirichlet" label = "Velocity">
        <value> [0.0,0.0,0.0] </value>
      </BCType>
    </Face>
    <Face side = "y+">
      <BCType id = "all" var = "Neumann" label = "Velocity">
        <value> [0.0,0.0,0.0] </value>
      </BCType>
    </Face>
    <Face side = "z-">
      <BCType id = "all" var = "Dirichlet" label = "Velocity">
        <value> [0.0,0.0,0.0] </value>
      </BCType>
    </Face>
    <Face side = "z+">
      <BCType id = "all" var = "symmetry" label = "Symmetric"> </BCType>
    </Face>
  </BoundaryConditions>

  <Level>
    <Box label = "Domain">
      <lower>[0,0,0]</lower>
      <upper>[13e-3,45e-3,40e-3]</upper>
      <extraCells>[1,1,1]</extraCells>
      <patches>[4,1,80]</patches>
      <!--<patches>[1,1,16]</patches>-->
    </Box>
    <spacing>[0.5e-3, 0.5e-3, 0.5e-3]</spacing>
  </Level>
</Grid>

<DataArchiver>
  <filebase>/nobackup/mnsg/uintah_dev5_1/output/dura_v3_pro.4.uda</
      filebase>
  <!--<filebase>/usr/not-backed-up/UCF_output/dura_v3.uda</filebase>
      -->
  <outputInterval>0.05e-3</outputInterval>
```

```
      <save label = "p.particleID"/>
      <save label = "p.x"/>
      <save label = "p.velocity"/>
      <save label = "p.stress"/>
      <save label = "p.color"/>
      <checkpoint cycle = "2" interval = "0.0001"/>
</DataArchiver>

<MPM>
  <time_integrator>explicit</time_integrator>
  <interpolator>gimp</interpolator>
  <withColor>true</withColor>
  <DoPressureStabilization>false</DoPressureStabilization>
</MPM>

<PhysicalConstants>
  <gravity>[0,0,0]</gravity>
</PhysicalConstants>

<MaterialProperties>
  <MPM>
    <material name="cord_tissue">
      <!--density of SC tissue from Persson-->
      <density>1050</density>
      <!--thermal conductivity for water at 25 celcius is 0.58 W/(m.K)-->
      <thermal_conductivity>1</thermal_conductivity>
      <!--specific heat capacity for water is 4.1855 J/(g.K)-->
      <specific_heat>1</specific_heat>

      <constitutive_model type = "comp_ogden_standard">
        <bulk_modulus>45e3</bulk_modulus>
        <mu_1>2000</mu_1>
        <alpha_1>9</alpha_1>
        <equation_of_state type="default_hyper"></equation_of_state>
      </constitutive_model>

      <geom_object>
        <!--elliptical cord, dimensions from Persson et al. 2011 -->
        <ellipsoid label = "cord">
          <origin> [13e-3, 6.6e-3, 7e-3] </origin>
          <rx> 7.5e-3 </rx>
          <ry> 5e-3 </ry>
          <rz> 1e12 </rz><!--effectively creates an elliptical tube within the
              domain, 140mm long-->
        </ellipsoid>

        <res>[2,2,2]</res>
        <velocity>[0.0,0.0,0.0]</velocity>
        <temperature>310.15</temperature>
        <color>20</color>
      </geom_object>
    </material>

    <material name="impactor">
      <!--density of tufnol composite, from Persson et al. 2011-->
      <density>1360</density>
      <thermal_conductivity>1</thermal_conductivity>
      <specific_heat>1</specific_heat>

      <constitutive_model type = "comp_linear">
```

```
          <E>6.5e9</E>
          <v>0.3</v>
        </constitutive_model>
  <!--
        <constitutive_model type = "rigid"></constitutive_model>
  -->
        <geom_object>
          <!-- dimensions of pellet 1, from Persson et al. 2011-->
          <union>
          <cylinder>
            <bottom> [13e-3, 13e-3, 40e-3] </bottom>
            <top> [13e-3, 43e-3, 40e-3] </top>
            <radius> 6.00e-3 </radius>
          </cylinder>
          <cylinder>
            <bottom> [13e-3, 13e-3, 40e-3] </bottom>
            <top> [13e-3, 23e-3, 40e-3] </top>
            <radius> 10.00e-3 </radius>
          </cylinder>
          </union>

          <res>[2,2,2]</res>
          <!-- Speed from from Persson et al. 2011-->
          <velocity>[0,-4.5,0]</velocity>
          <temperature>310.15</temperature>
          <color>40</color>
        </geom_object>
      </material>

      <material name="steel_posterior_plate">
        <!--From CP thesis-->
        <density>8000</density>
        <!--thermal conductivity for 1% carbon steel at 25 celcius is 43 W/(m.K)
            -->
        <thermal_conductivity>1</thermal_conductivity>
        <!--specific heat capacity for steel is 0.466 J/(g.K)-->
        <specific_heat>1</specific_heat>

        <constitutive_model type = "comp_linear">
          <E>193e9</E>
          <v>0.3</v>
        </constitutive_model>
<!--
        <constitutive_model type = "rigid"></constitutive_model>
    -->
        <geom_object>
          <box>
            <min>[0,0,0]</min>
            <max>[13e-3,1e-3,80e-3]</max>
          </box>
          <res>[2,2,2]</res>
          <velocity>[0.0,0.0,0.0]</velocity>
          <temperature>310.15</temperature>
          <color>60</color>
        </geom_object>
      </material>

      <material name = "dura_mater">
        <density>1000</density>
        <thermal_conductivity>1</thermal_conductivity>
```

```
        <specific_heat>1</specific_heat>

 <!--
        <constitutive_model type = "comp_linear">
          <E>80e6</E>
          <v>0.49</v>
        </constitutive_model>
 -->

        <constitutive_model type = "comp_ogden_storakers">
          <PR>0.4</PR>
          <mu_1>322e3</mu_1>
          <alpha_1>19</alpha_1>
        </constitutive_model>


<!--
        <constitutive_model type = "comp_ogden_standard">
          <bulk_modulus>152950e3</bulk_modulus>
          <mu_1>322e3</mu_1>
          <alpha_1>19</alpha_1>
          <equation_of_state type="default_hyper"></equation_of_state>
        </constitutive_model>
-->

        <geom_object>
          <difference>
            <ellipsoid label = "outer_dura">
              <origin> [13e-3, 6.6e-3, 7e-3] </origin>
              <rx> 9e-3 </rx>
              <ry> 5.6e-3 </ry>
              <rz> 1e12 </rz>
            </ellipsoid>
            <ellipsoid label = "inner_dura">
              <origin> [13e-3, 6.6e-3, 7e-3] </origin>
              <rx> 8.5e-3 </rx>
              <ry> 5.1e-3 </ry>
              <rz> 1e12 </rz>
            </ellipsoid>
          </difference>

          <res>[2,2,2]</res>
          <velocity>[0,0,0]</velocity>
          <temperature>310.15</temperature>
          <color>80</color>
        </geom_object>
      </material>

    <contact>
      <type>friction</type>
      <materials>[0,1,2,3]</materials>
      <mu>0.0</mu>
    </contact>

    </MPM>
  </MaterialProperties>
</Uintah_specification>
```

### A.1.5 *Cord/Dura P2*

```
%[caption=dura_v3_p2.ups]
<?xml version='1.0' encoding='ISO-8859-1' ?>

<Uintah_specification>

  <SimulationComponent type = "mpm"/>

  <Time>
    <maxTime>7e-3</maxTime>
    <initTime>0</initTime>
    <delt_min>1.0e-12</delt_min>
    <delt_max>100.0e-3</delt_max>
    <delt_init>100.0e-3</delt_init>
    <timestep_multiplier>0.1</timestep_multiplier>
  </Time>

  <Grid>
    <BoundaryConditions>
      <Face side = "x-">
        <BCType id = "all" var = "Neumann" label = "Velocity">
          <value> [0.0,0.0,0.0] </value>
        </BCType>
      </Face>
      <Face side = "x+">
        <BCType id = "all" var = "symmetry" label = "Symmetric"> </BCType>
      </Face>
      <Face side = "y-">
        <BCType id = "all" var = "Dirichlet" label = "Velocity">
          <value> [0.0,0.0,0.0] </value>
        </BCType>
      </Face>
      <Face side = "y+">
        <BCType id = "all" var = "Neumann" label = "Velocity">
          <value> [0.0,0.0,0.0] </value>
        </BCType>
      </Face>
      <Face side = "z-">
        <BCType id = "all" var = "Dirichlet" label = "Velocity">
          <value> [0.0,0.0,0.0] </value>
        </BCType>
      </Face>
      <Face side = "z+">
        <BCType id = "all" var = "symmetry" label = "Symmetric"> </BCType>
      </Face>
    </BoundaryConditions>

    <Level>
      <Box label = "Domain">
        <lower>[0,0,0]</lower>
        <upper>[13e-3,45e-3,40e-3]</upper>
        <extraCells>[1,1,1]</extraCells>
        <patches>[4,1,80]</patches>
        <!--<patches>[1,1,16]</patches>-->
      </Box>
      <spacing>[0.5e-3, 0.5e-3, 0.5e-3]</spacing>
    </Level>
  </Grid>

  <DataArchiver>
```

```
  <filebase>/nobackup/mnsg/uintah_dev5_1/output/dura_v3_p2.uda</
      filebase>
  <!--<filebase>/usr/not-backed-up/UCF_output/dura_v3_p2.uda</filebase
      >-->
  <outputInterval>0.05e-3</outputInterval>
    <save label = "p.particleID"/>
    <save label = "p.x"/>
    <save label = "p.velocity"/>
    <save label = "p.stress"/>
    <save label = "p.color"/>
    <checkpoint cycle = "2" interval = "0.0001"/>
</DataArchiver>

<MPM>
  <time_integrator>explicit</time_integrator>
  <interpolator>gimp</interpolator>
  <withColor>true</withColor>
  <DoPressureStabilization>false</DoPressureStabilization>
</MPM>

<PhysicalConstants>
  <gravity>[0,0,0]</gravity>
</PhysicalConstants>

<MaterialProperties>
  <MPM>
    <material name="cord_tissue">
      <density>1050</density>
      <thermal_conductivity>0</thermal_conductivity>
      <specific_heat>1</specific_heat>

      <constitutive_model type = "comp_ogden_standard">
        <bulk_modulus>45e3</bulk_modulus>
        <mu_1>2000</mu_1>
        <alpha_1>9</alpha_1>
        <equation_of_state type="default_hyper"></equation_of_state>
      </constitutive_model>

      <geom_object>
        <ellipsoid label = "cord">
          <origin> [13e-3, 6.6e-3, 7e-3] </origin>
          <rx> 7.5e-3 </rx>
          <ry> 5e-3 </ry>
          <rz> 1e12 </rz>
        </ellipsoid>

        <res>[2,2,2]</res>
        <velocity>[0.0,0.0,0.0]</velocity>
        <temperature>310.15</temperature>
        <color>20</color>
      </geom_object>
    </material>

    <material name="impactor">
      <density>1360</density>
      <thermal_conductivity>0</thermal_conductivity>
      <specific_heat>1</specific_heat>

      <constitutive_model type = "comp_linear">
        <E>6.5e9</E>
```

```
        <v>0.3</v>
      </constitutive_model>
      <geom_object>
        <union>
          <cylinder label = "impactor_tail">
            <bottom> [13e-3, 13.5e-3, 40e-3] </bottom>
            <top>  [13e-3, 35.5e-3, 40e-3] </top>
            <radius> 7.00e-3     </radius>
          </cylinder>
          <cylinder label = "impactor_mid">
            <bottom> [13e-3, 25.5e-3, 40e-3] </bottom>
            <top>  [13e-3, 35.5e-3, 40e-3] </top>
            <radius> 10.00e-3    </radius>
          </cylinder>
          <cylinder label = "impactor_head">
            <bottom> [13e-3, 35.5e-3, 40e-3] </bottom>
            <top>  [13e-3, 38.5e-3, 40e-3] </top>
            <radius> 6.00e-3     </radius>
          </cylinder>
        </union>

        <res>[2,2,2]</res>
        <velocity>[0,-4.5,0]</velocity>
        <temperature>310.15</temperature>
        <color>40</color>
      </geom_object>
    </material>

    <material name="steel_posterior_plate">
      <density>8000</density>
      <thermal_conductivity>0</thermal_conductivity>
      <specific_heat>1</specific_heat>

      <constitutive_model type = "comp_linear">
        <E>193e9</E>
        <v>0.3</v>
      </constitutive_model>

      <geom_object>
        <box>
          <min>[0,0,0]</min>
          <max>[13e-3,1e-3,80e-3]</max>
        </box>
        <res>[2,2,2]</res>
        <velocity>[0.0,0.0,0.0]</velocity>
        <temperature>310.15</temperature>
        <color>60</color>
      </geom_object>
    </material>

    <material name = "dura_mater">
      <density>1000</density>
      <thermal_conductivity>0</thermal_conductivity>
      <specific_heat>1</specific_heat>

      <constitutive_model type = "comp_ogden_storakers">
        <PR>0.4</PR>
        <mu_1>322e3</mu_1>
        <alpha_1>19</alpha_1>
      </constitutive_model>
```

```
        <geom_object>
          <difference>
            <ellipsoid label = "outer_dura">
              <origin> [13e-3, 6.6e-3, 7e-3] </origin>
              <rx> 9e-3 </rx>
              <ry> 5.6e-3 </ry>
              <rz> 1e12 </rz>
            </ellipsoid>
            <ellipsoid label = "inner_dura">
              <origin> [13e-3, 6.6e-3, 7e-3] </origin>
              <rx> 8.5e-3 </rx>
              <ry> 5.1e-3 </ry>
              <rz> 1e12 </rz>
            </ellipsoid>
          </difference>

          <res>[2,2,2]</res>
          <velocity>[0,0,0]</velocity>
          <temperature>310.15</temperature>
          <color>80</color>
        </geom_object>
      </material>

    <contact>
      <type>friction</type>
      <materials>[0,1,2,3]</materials>
      <mu>0.0</mu>
    </contact>

    </MPM>
  </MaterialProperties>
</Uintah_specification>
```

### A.1.6 *Cord/Dura P3*

```
%[caption=dura_v3_p2.ups]
<?xml version='1.0' encoding='ISO-8859-1' ?>

<Uintah_specification>

  <SimulationComponent type = "mpm"/>

  <Time>
    <maxTime>7e-3</maxTime>
    <initTime>0</initTime>
    <delt_min>1.0e-12</delt_min>
    <delt_max>100.0e-3</delt_max>
    <delt_init>100.0e-3</delt_init>
    <timestep_multiplier>0.1</timestep_multiplier>
  </Time>

  <Grid>
    <BoundaryConditions>
      <Face side = "x-">
        <BCType id = "all" var = "Neumann" label = "Velocity">
          <value> [0.0,0.0,0.0] </value>
        </BCType>
      </Face>
      <Face side = "x+">
```

```
      <BCType id = "all" var = "symmetry" label = "Symmetric"> </BCType>
    </Face>
    <Face side = "y-">
      <BCType id = "all" var = "Dirichlet" label = "Velocity">
        <value> [0.0,0.0,0.0] </value>
      </BCType>
    </Face>
    <Face side = "y+">
      <BCType id = "all" var = "Neumann" label = "Velocity">
        <value> [0.0,0.0,0.0] </value>
      </BCType>
    </Face>
    <Face side = "z-">
      <BCType id = "all" var = "Dirichlet" label = "Velocity">
        <value> [0.0,0.0,0.0] </value>
      </BCType>
    </Face>
    <Face side = "z+">
      <BCType id = "all" var = "symmetry" label = "Symmetric"> </BCType>
    </Face>
  </BoundaryConditions>

  <Level>
    <Box label = "Domain">
      <lower>[0,0,0]</lower>
      <upper>[13e-3,45e-3,40e-3]</upper>
      <extraCells>[1,1,1]</extraCells>
      <patches>[4,1,80]</patches>
      <!--<patches>[1,1,16]</patches>-->
    </Box>
    <spacing>[0.5e-3, 0.5e-3, 0.5e-3]</spacing>
  </Level>
</Grid>

<DataArchiver>
  <filebase>/nobackup/mnsg/uintah_dev5_1/output/dura_v3_p3.uda</
      filebase>
  <!--<filebase>/usr/not-backed-up/UCF_output/dura_v3_p3.uda</filebase
      >-->
  <outputInterval>0.05e-3</outputInterval>
    <save label = "p.particleID"/>
    <save label = "p.x"/>
    <save label = "p.velocity"/>
    <save label = "p.stress"/>
    <save label = "p.color"/>
    <save label = "p.scalefactor"/>
    <checkpoint cycle = "2" interval = "0.0001"/>
</DataArchiver>

<MPM>
  <time_integrator>explicit</time_integrator>
  <interpolator>gimp</interpolator>
  <withColor>true</withColor>
  <DoPressureStabilization>false</DoPressureStabilization>
</MPM>

<PhysicalConstants>
  <gravity>[0,0,0]</gravity>
</PhysicalConstants>
```

```
<MaterialProperties>
  <MPM>
    <material name="cord_tissue">
      <density>1050</density>
      <thermal_conductivity>0</thermal_conductivity>
      <specific_heat>1</specific_heat>

      <constitutive_model type = "comp_ogden_standard">
        <bulk_modulus>45e3</bulk_modulus>
        <mu_1>2000</mu_1>
        <alpha_1>9</alpha_1>
        <equation_of_state type="default_hyper"></equation_of_state>
      </constitutive_model>

      <geom_object>
        <ellipsoid label = "cord">
          <origin> [13e-3, 6.6e-3, 7e-3] </origin>
          <rx> 7.5e-3 </rx>
          <ry> 5e-3 </ry>
          <rz> 1e12 </rz>
        </ellipsoid>

        <res>[2,2,2]</res>
        <velocity>[0.0,0.0,0.0]</velocity>
        <temperature>310.15</temperature>
        <color>20</color>
      </geom_object>
    </material>

    <material name="impactor">
      <density>1360</density>
      <thermal_conductivity>0</thermal_conductivity>
      <specific_heat>1</specific_heat>

      <constitutive_model type = "comp_linear">
        <E>6.5e9</E>
        <v>0.3</v>
      </constitutive_model>
      <geom_object>
        <union>
          <cylinder label = "impactor_head">
            <bottom> [13e-3, 13.5e-3, 40e-3] </bottom>
            <top>  [13e-3, 28.5e-3, 40e-3] </top>
            <radius> 5.00e-3     </radius>
          </cylinder>
          <cylinder label = "impactor_mid">
            <bottom> [13e-3, 28.5e-3, 40e-3] </bottom>
            <top>  [13e-3, 40.5e-3, 40e-3] </top>
            <radius> 10.00e-3    </radius>
          </cylinder>
          <cylinder label = "impactor_tail">
            <bottom> [13e-3, 40.5e-3, 40e-3] </bottom>
            <top>  [13e-3, 43.5e-3, 40e-3] </top>
            <radius> 6.00e-3     </radius>
          </cylinder>
        </union>

        <res>[2,2,2]</res>
        <velocity>[0,-4.5,0]</velocity>
        <temperature>310.15</temperature>
```

```
      <color>40</color>
    </geom_object>
  </material>

  <material name="steel_posterior_plate">
    <density>8000</density>
    <thermal_conductivity>0</thermal_conductivity>
    <specific_heat>1</specific_heat>

    <constitutive_model type = "comp_linear">
      <E>193e9</E>
      <v>0.3</v>
    </constitutive_model>

    <geom_object>
      <box>
        <min>[0,0,0]</min>
        <max>[13e-3,1e-3,80e-3]</max>
      </box>
      <res>[2,2,2]</res>
      <velocity>[0.0,0.0,0.0]</velocity>
      <temperature>310.15</temperature>
      <color>60</color>
    </geom_object>
  </material>

  <material name = "dura_mater">
    <density>1000</density>
    <thermal_conductivity>0</thermal_conductivity>
    <specific_heat>1</specific_heat>

    <constitutive_model type = "comp_ogden_storakers">
      <PR>0.4</PR>
      <mu_1>322e3</mu_1>
      <alpha_1>19</alpha_1>
    </constitutive_model>

    <geom_object>
      <difference>
        <ellipsoid label = "outer_dura">
          <origin> [13e-3, 6.6e-3, 7e-3] </origin>
          <rx> 9e-3 </rx>
          <ry> 5.6e-3 </ry>
          <rz> 1e12 </rz>
        </ellipsoid>
        <ellipsoid label = "inner_dura">
          <origin> [13e-3, 6.6e-3, 7e-3] </origin>
          <rx> 8.5e-3 </rx>
          <ry> 5.1e-3 </ry>
          <rz> 1e12 </rz>
        </ellipsoid>
      </difference>

      <res>[2,2,2]</res>
      <velocity>[0,0,0]</velocity>
      <temperature>310.15</temperature>
      <color>80</color>
    </geom_object>
  </material>
```

```
    <contact>
      <type>friction</type>
      <materials>[0,1,2,3]</materials>
      <mu>0.0</mu>
    </contact>

    </MPM>
  </MaterialProperties>
</Uintah_specification>
```

### A.1.7 *Cord/Dura/CSF P1*

```
<?xml version='1.0' encoding='ISO-8859-1' ?>

<Uintah_specification>

  <Meta>
    <title>CSF SCI</title>
  </Meta>

  <SimulationComponent type="mpmice" />

  <Time>
    <maxTime>  8e-3  </maxTime>
    <initTime>  0.0   </initTime>
    <delt_min>  1e-12  </delt_min>
    <delt_max>  1.0   </delt_max>
    <max_delt_increase>  1.0   </max_delt_increase>
    <timestep_multiplier> 0.2   </timestep_multiplier>
  </Time>

  <DataArchiver>

    <filebase>/nobackup/mnsg/udev5_3/output/csf1.7.1_3D_3ATM.uda</
        filebase>
    <!--
    <filebase>/usr/not-backed-up/UCF_output1/csf1.7.1_3D_3ATM.uda</
        filebase>
    -->
    <outputInterval>0.02e-3</outputInterval>
    <checkpoint cycle = "2" interval = "0.02e-3"/>
    <!-- MPM -->
    <save label="p.x"/>
    <save label="p.velocity"/>
    <save label="p.volume"/>
    <save label="p.size"/>
    <save label="p.temperature"/>
      <save label="p.deformationMeasure"/>
    <save label="p.stress"/>
    <save label="p.color"/>
    <save label="p.scalefactor"/>
    <save label="p.particleID"/>
    <save label="g.mass"/>
    <save label="g.velocity"/>
    <save label="g.velocity_star"/>
    <!-- ICE -->
    <save label="press_equil_CC"/>
    <save label="vol_frac_CC"/>
    <save label="sp_vol_CC"/>
    <save label="speedSound_CC"/>
```

```
    <save label="vel_CC"/>
    <save label="rho_CC"/>
    <save label="temp_CC"/>
    <save label="mom_source_CC"/>
  </DataArchiver>

  <PhysicalConstants>
    <gravity>    [0, 0, 0] </gravity>
    <reference_pressure> 303975 </reference_pressure>
  </PhysicalConstants>

  <MPM>
        <time_integrator>  explicit  </time_integrator>
        <testForNegTemps_mpm>  false  </testForNegTemps_mpm>
        <interpolator>  gimp  </interpolator>
    <withColor>    true  </withColor>
  </MPM>

  <CFD>
    <cfl>0.1</cfl>
    <ICE>
      <advection type = "SecondOrder" />
      <TimeStepControl>
        <Scheme_for_delT_calc>   aggressive </Scheme_for_delT_calc>
        <knob_for_speedSound>   1   </knob_for_speedSound>
      </TimeStepControl>

      <applyHydrostaticPressure>   false  </applyHydrostaticPressure>
      <ClampSpecificVolume>    true  </ClampSpecificVolume>
<!--
      <ImplicitSolver>
        <max_outer_iterations>   10   </max_outer_iterations>
        <outer_iteration_tolerance>  1e-8  </outer_iteration_tolerance>
        <iters_before_timestep_restart> 10   </iters_before_timestep_restart>

        <Parameters variable="implicitPressure">
          <tolerance>    1.e-20  </tolerance>
          <solver>    cg   </solver>
          <preconditioner>    pfmg  </preconditioner>
          <maxiterations>    7500  </maxiterations>
          <npre>    1   </npre>
          <npost>    1   </npost>
          <skip>    1   </skip>
          <jump>    0   </jump>
          <relax_type>    2   </relax_type>
        </Parameters>
      </ImplicitSolver>
-->
    </ICE>
  </CFD>

  <MaterialProperties>
    <MPM>

      <material name="cord_tissue">

        <constitutive_model type = "comp_ogden_standard">
          <bulk_modulus>   45e3    </bulk_modulus>
          <mu_1>     2000    </mu_1>
          <alpha_1>    9     </alpha_1>
```

```
      <useModifiedEOS>  false    </useModifiedEOS>
      <equation_of_state type="default_hyper"> </equation_of_state>
    </constitutive_model>
    <density>     1054.5033333333333303 </density>
    <thermal_conductivity>  0      </thermal_conductivity>
    <!-- isochoric specific heat of water at 20atm 300 kelvin -->
    <specific_heat>    4129 </specific_heat>

    <geom_object>
      <ellipsoid label = "cord">
        <origin>   [15.2e-3, 7.2e-3, 40.4e-3] </origin>
        <rx>    7.5e-3     </rx>
        <ry>    5e-3     </ry>
        <rz>    1e12     </rz>
      </ellipsoid>

      <res>     [2, 2, 1]    </res>
      <velocity>    [0, 0, 0]    </velocity>
      <temperature>   310.15     </temperature>
      <color>     20      </color>
    </geom_object>
  </material>

  <material name = "dura_mater">

    <constitutive_model type = "comp_ogden_storakers">
      <PR>     0.4      </PR>
      <mu_1>     322e3     </mu_1>
      <alpha_1>    19      </alpha_1>
      <useModifiedEOS>  false     </useModifiedEOS>
      <equation_of_state type="default_hyper">  </equation_of_state>
    </constitutive_model>

    <density>     1000.0000001348602154 </density>
    <thermal_conductivity>  0      </thermal_conductivity>
    <specific_heat>    4129     </specific_heat>


    <geom_object>
      <difference>
        <ellipsoid label = "outer_dura">
          <origin>  [15.2e-3, 7.2e-3, 40.4e-3] </origin>
          <rx>    9.5e-3     </rx>
          <ry>   7e-3     </ry>
          <rz>   1e12     </rz>
        </ellipsoid>
        <ellipsoid label = "inner_dura">
          <origin>  [15.28e-3, 7.2e-3, 40.4e-3] </origin>
          <rx>   9e-3     </rx>
          <ry>   6.5e-3     </ry>
          <rz>   1e12     </rz>
        </ellipsoid>
      </difference>

      <res>     [2, 2, 1]    </res>
      <velocity>    [0, 0, 0]    </velocity>
      <temperature>   310.15      </temperature>
      <color>     80      </color>
    </geom_object>
  </material>
```

```
<material name="tufnol">

  <constitutive_model type = "comp_linear">
    <E>      6.5e9     </E>
    <v>      0.3       </v>
    <useModifiedEOS>  true     </useModifiedEOS>
    <equation_of_state type="default_hyper">  </equation_of_state>
  </constitutive_model>

  <density>     1360.0000374123076199 </density>
  <thermal_conductivity>  0      </thermal_conductivity>
  <specific_heat>    1500     </specific_heat>

  <geom_object>
  <!--
  <cylinder label = "short_impactor">
       <bottom> [14e-3, 15e-3, 40.4e-3] </bottom>
       <top> [14e-3, 25e-3, 40.4e-3] </top>
       <radius> 10.00e-3 </radius>
  </cylinder>
  -->
    <union>
      <cylinder label = "impactor_tail">
        <bottom> [15.2e-3, 16.4e-3, 0e-3] </bottom>
        <top> [15.2e-3, 46.4e-3, 0e-3] </top>
        <radius> 6.00e-3 </radius>
      </cylinder>
      <cylinder label = "impactor_head">
        <bottom> [15.2e-3, 16.4e-3, 0e-3] </bottom>
        <top> [15.28e-3, 26.4e-3, 0e-3] </top>
        <radius> 10.00e-3 </radius>
      </cylinder>
    </union>

    <res>     [2, 2, 1]    </res>
    <velocity>    [0, -4.5, 0]   </velocity>
    <temperature>   310.15     </temperature>
    <color>     40      </color>
  </geom_object>
</material>

<material name="backplate">

  <constitutive_model type = "comp_linear">
    <E>      193e9     </E>
    <v>      0.3       </v>
    <useModifiedEOS>  true     </useModifiedEOS>
    <equation_of_state type="default_hyper">  </equation_of_state>
  </constitutive_model>

  <!--<density>     8000.0000025199997253 </density>-->
  <density>     1360.0000374123076199 </density>
  <thermal_conductivity>  0      </thermal_conductivity>
  <!--<specific_heat>    4130     </specific_heat>-->
  <specific_heat>    1500     </specific_heat>

  <geom_object>
    <box label = "backplate">
      <min>    [0,  0,  0]    </min>
```

```
          <max>    [15.2e-3, 0.2e-3, 40e-3] </max>
        </box>
        <res>    [2, 2, 1]   </res>
        <velocity>    [0.0, 0.0, 0.0]   </velocity>
        <temperature>   300     </temperature>
        <color>     60     </color>
      </geom_object>
    </material>

    <contact>
      <type>     friction    </type>
      <materials>     [0, 1, 2, 3]   </materials>
      <mu>     0.0     </mu>
    </contact>
  </MPM>

  <ICE>

    <material name = "air">
      <EOS type = "ideal_gas">    </EOS>
      <dynamic_viscosity>   0.0    </dynamic_viscosity>
      <thermal_conductivity>  0      </thermal_conductivity>
      <!-- isochoric specific heat of air at 310.15K at 5 atm pressure -->
      <!--<specific_heat>    719.6    </specific_heat>-->
      <!-- isochoric specific heat of air at 310.15K at 3 atm pressure -->
      <specific_heat>    719.1    </specific_heat>
      <gamma>     1.4   </gamma>

      <geom_object>
        <difference>
          <box label="whole_domain">
            <min>  [0e-3,   0e-3,  0e-3] </min>
            <max>  [15.2e-3, 50e-3, 40e-3] </max>
            <!--<max>[13.94e-3, 46.92e-3, 0.34e-3]</max>-->
          </box>
          <union>
            <box label = "backplate"></box>
            <cylinder label = "impactor_tail"></cylinder>
            <cylinder label = "impactor_head"></cylinder>
            <ellipsoid label = "outer_dura"></ellipsoid>
          </union>
        </difference>

        <res>    [2, 2, 1]    </res>
        <velocity>    [0.0, 0.0, 0.0]   </velocity>
        <temperature>   310.15     </temperature>
                <!--<density>   5.6749715055   </density> -->
                <!-- density at 3 atm  -->
                <density>   3.4142587173926797  </density>
        <pressure>    0    </pressure>
      </geom_object>
    </material>

  <material name = "csf">

      <EOS type="Thomsen_Hartka_water">
        <a>     2.0e-7     </a>
        <b>     2.6     </b>
        <co>     4205.7     </co>
        <ko>     5.0e-10     </ko>
```

```
        <To>    277.0    </To>
        <L>     8.0e-6   </L>
        <vo>    1.00008e-3   </vo>
      </EOS>

      <dynamic_viscosity>  0.000    </dynamic_viscosity>
      <!-- THWAter Script 3 atm -->
            <gamma>  1.0000392285636475  </gamma>
      <thermal_conductivity> 0     </thermal_conductivity>
      <!-- THWAter Script 3 atm -->
      <specific_heat>    4117.8398968398732904 </specific_heat>

      <geom_object>

        <difference>
          <ellipsoid label = "inner_dura"></ellipsoid>
          <ellipsoid label = "cord"></ellipsoid>
        </difference>
        <res>  [2, 2, 1]   </res>
        <velocity>  [0.0, 0.0, 0.0]   </velocity>
        <!-- THWAter Script 3 atm -->
        <density>   991.3235777144317353 </density>
        <pressure>  0     </pressure>
        <temperature>  310.15     </temperature>
        <color>    100     </color>
      </geom_object>
    </material>

  </ICE>

  <exchange_properties>
    <exchange_coefficients>
      <!--
      0) cord_tissue
      1) dura_mater
      2) tufnol
      3) backplate
      4) air
      5) csf

      0->1, 0->2, 0->3, 0->4, 0->5,
           1->2, 1->3, 1->4, 1->5,
                2->3, 2->4, 2->5,
                     3->4, 3->5,
                          4->5,

            0->1,  0->2,  0->3,  0->4,  0->5,  1->2,  1->3,  1->4, 1->5,  2->3,
                  2->4,  2->5,  3->4,  3->5,  4->5,
      -->
      <momentum> [1e10.6,  1e10.6,  1e10.6,  1e10.6,  1e10.6,  1e10.6,  1e10.6,  1e1,
            1e10.6,  1e10.6,  1e10.6,  1e10.6,  1e10.6,  1e10.6,  1e10.6] </
          momentum>
      <!--<momentum> [1e10.6,  0,    0,   0,    1e10.6,  1e10.6,  1e10.6,  1e1,  1e10
          .6,  0,    1e10.6,  0,    1e1,   0,    0] </momentum>-->
      <heat>  [0,   0,    0,    0,    0,    0,    0,    0,  0,   0,    0,    0,    0,   0,   0 ] </heat
          >
    </exchange_coefficients>
  </exchange_properties>
</MaterialProperties>
```

```
<Grid>
  <BoundaryConditions>
    <Face side = "x−">
      <BCType id = "0" label = "Pressure" var = "Neumann">
        <value> 0.0 </value>
      </BCType>
      <BCType id = "all" label = "Velocity" var = "Dirichlet">
        <value> [0, 0, 0] </value>
      </BCType>
      <BCType id = "all" label = "Temperature" var = "Neumann">
        <value> 0.0 </value>
      </BCType>
      <BCType id = "all" label = "Density" var = "Neumann">
        <value> 0.0 </value>
      </BCType>
    </Face>

    <Face side = "x+">
      <BCType id = "all" var = "symmetry" label = "Symmetric"> </BCType>
      <!−−<BCType id = "0" label = "Pressure" var = "Neumann">
        <value> 0.0 </value>
      </BCType>
      <BCType id = "all" label = "Velocity" var = "Dirichlet">
        <value> [0, 0, 0] </value>
      </BCType>
      <BCType id = "all" label = "Temperature" var = "Neumann">
        <value> 0.0 </value>
      </BCType>
      <BCType id = "all" label = "Density" var = "Neumann">
        <value> 0.0 </value>
      </BCType>
      −−>
    </Face>

    <Face side = "y−">
      <BCType id = "0" label = "Pressure" var = "Neumann">
        <value> 0.0 </value>
      </BCType>
      <BCType id = "all" label = "Velocity" var = "Dirichlet">
        <value> [0, 0, 0] </value>
      </BCType>
      <BCType id = "all" label = "Temperature" var = "Neumann">
        <value> 0.0 </value>
      </BCType>
      <BCType id = "all" label = "Density" var = "Neumann">
        <value> 0.0 </value>
      </BCType>
    </Face>

    <Face side = "y+">
      <BCType id = "0" label = "Pressure" var = "Neumann">
        <value> 0.0 </value>
      </BCType>

      <BCType id = "all" label = "Velocity" var = "Dirichlet">
        <value> [0, 0, 0] </value>
      </BCType>
      <!−−
      <BCType id = "all" label = "Velocity" var = "Neumann">
        <value> [0, 0, 0] </value>
```

```
        </BCType> -->
        <BCType id = "all" label = "Temperature" var = "Neumann">
          <value> 0.0 </value>
        </BCType>
        <BCType id = "all" label = "Density" var = "Neumann">
          <value> 0.0 </value>
        </BCType>
      </Face>

      <Face side = "z-">
        <BCType id = "all" var = "symmetry" label = "Symmetric"> </BCType>
      </Face>

      <Face side = "z+">
        <!--
        <BCType id = "all" var = "symmetry" label = "Symmetric"> </BCType>
        -->
        <BCType id = "o" label = "Pressure" var = "Neumann">
          <value> 0.0 </value>
        </BCType>
        <BCType id = "all" label = "Velocity" var = "Dirichlet">
          <value> [0, 0, 0] </value>
        </BCType>
        <BCType id = "all" label = "Temperature" var = "Neumann">
          <value> 0.0 </value>
        </BCType>
        <BCType id = "all" label = "Density" var = "Neumann">
          <value> 0.0 </value>
        </BCType>
      </Face>
    </BoundaryConditions>

    <Level>
      <!--
      <Box label = "Domain">
        <lower>   [0e-3,   0e-3,  0e-3]   </lower>
        <upper>   [13.94e-3, 46.92e-3, 0.34e-3]  </upper>
        <extraCells> [1, 1, 1]      </extraCells>
        <patches>  [12, 1, 1]      </patches>
      </Box>
      <spacing>   [0.34e-3, 0.34e-3, 0.34e-3]  </spacing>
      -->
      <Box label = "Domain">
        <lower>   [0e-3,   0e-3,  0e-3] </lower>
        <upper>   [15.2e-3, 50e-3,  40e-3] </upper>
        <extraCells> [1, 1, 1]      </extraCells>
        <patches>  [10, 1, 40]      </patches>
      </Box>
      <spacing>   [0.4e-3, 0.4e-3, 0.4e-3]  </spacing>

    </Level>
  </Grid>
</Uintah_specification>

<!--
mpirun -np 12 /usr/not-backed-up/udev5_3/opt/StandAlone/sus -mpi /usr/
    not-backed-up/csf1.7.1_3D_3ATM.ups 2>&1 | tee /usr/not-backed-up/
    csf1.7.1_3D_3ATM.txt
-->
```

### A.1.8 *Cord/Dura/CSF P2*

```
<?xml version='1.0' encoding='ISO-8859-1' ?>

<Uintah_specification>

  <Meta>
    <title>CSF SCI</title>
  </Meta>

  <SimulationComponent type="mpmice" />

  <Time>
    <maxTime>  8e-3  </maxTime>
    <initTime>  0.0   </initTime>
    <delt_min>  1e-12  </delt_min>
    <delt_max>  1.0   </delt_max>
    <max_delt_increase>  1.0   </max_delt_increase>
    <timestep_multiplier> 0.2   </timestep_multiplier>
  </Time>

  <DataArchiver>

    <filebase>/nobackup/mnsg/udev5_3/output/csf1.7.1_3D_3ATM_P2.uda</
        filebase>
    <!--
    <filebase>/usr/not-backed-up/UCF_output1/csf1.7.1_3D_3ATM_P2.uda</
        filebase>
    -->
    <outputInterval>0.05e-3</outputInterval>
    <checkpoint cycle = "2" interval = "0.02e-3"/>
    <!-- MPM -->
    <save label="p.x"/>
    <save label="p.velocity"/>
    <save label="p.volume"/>
    <save label="p.size"/>
    <save label="p.temperature"/>
      <save label="p.deformationMeasure"/>
    <save label="p.stress"/>
    <save label="p.color"/>
    <save label="p.scalefactor"/>
    <save label="p.particleID"/>
    <save label="g.mass"/>
    <save label="g.velocity"/>
    <save label="g.velocity_star"/>
    <!-- ICE -->
    <save label="press_equil_CC"/>
    <save label="vol_frac_CC"/>
    <save label="sp_vol_CC"/>
    <save label="speedSound_CC"/>
    <save label="vel_CC"/>
    <save label="rho_CC"/>
    <save label="temp_CC"/>
    <save label="mom_source_CC"/>
  </DataArchiver>

  <PhysicalConstants>
    <gravity>    [0, 0, 0] </gravity>
    <reference_pressure> 303975 </reference_pressure>
```

```
  </PhysicalConstants>

  <MPM>
        <time_integrator>  explicit  </time_integrator>
        <testForNegTemps_mpm>  false  </testForNegTemps_mpm>
        <interpolator>  gimp  </interpolator>
    <withColor>    true  </withColor>
  </MPM>

  <CFD>
    <cfl>0.1</cfl>
    <ICE>
      <advection type = "SecondOrder" />
      <TimeStepControl>
        <Scheme_for_delT_calc>   aggressive </Scheme_for_delT_calc>
        <knob_for_speedSound>   1   </knob_for_speedSound>
      </TimeStepControl>

      <applyHydrostaticPressure>   false  </applyHydrostaticPressure>
      <ClampSpecificVolume>    true  </ClampSpecificVolume>
<!--
      <ImplicitSolver>
        <max_outer_iterations>   10   </max_outer_iterations>
        <outer_iteration_tolerance>  1e-8  </outer_iteration_tolerance>
        <iters_before_timestep_restart> 10   </iters_before_timestep_restart>

        <Parameters variable="implicitPressure">
          <tolerance>    1.e-20  </tolerance>
          <solver>    cg   </solver>
          <preconditioner>    pfmg  </preconditioner>
          <maxiterations>    7500  </maxiterations>
          <npre>    1   </npre>
          <npost>    1   </npost>
          <skip>    1   </skip>
          <jump>    0   </jump>
          <relax_type>    2   </relax_type>
        </Parameters>
      </ImplicitSolver>
-->
    </ICE>
  </CFD>

  <MaterialProperties>
    <MPM>

      <material name="cord_tissue">

        <constitutive_model type = "comp_ogden_standard">
          <bulk_modulus>   45e3    </bulk_modulus>
          <mu_1>     2000    </mu_1>
          <alpha_1>    9     </alpha_1>
          <useModifiedEOS>  false    </useModifiedEOS>
          <equation_of_state type="default_hyper"> </equation_of_state>
        </constitutive_model>
        <density>     1054.5033333333333303 </density>
        <thermal_conductivity>  0      </thermal_conductivity>
        <!-- isochoric specific heat of water at 20atm 300 kelvin -->
        <specific_heat>    4129 </specific_heat>

        <geom_object>
```

```
      <ellipsoid label = "cord">
        <origin>  [12.8e-3, 7.2e-3, 40.4e-3] </origin>
        <rx>   7.5e-3    </rx>
        <ry>   5e-3    </ry>
        <rz>   1e12    </rz>
      </ellipsoid>

      <res>    [2, 2, 1]   </res>
      <velocity>   [0, 0, 0]   </velocity>
      <temperature>  310.15    </temperature>
      <color>    20     </color>
    </geom_object>
  </material>

  <material name = "dura_mater">

    <constitutive_model type = "comp_ogden_storakers">
      <PR>    0.4     </PR>
      <mu_1>    322e3    </mu_1>
      <alpha_1>   19     </alpha_1>
      <useModifiedEOS>  false    </useModifiedEOS>
      <equation_of_state type="default_hyper">  </equation_of_state>
    </constitutive_model>

    <density>    1000.0000001348602154 </density>
    <thermal_conductivity>  0     </thermal_conductivity>
    <specific_heat>   4129    </specific_heat>


    <geom_object>
      <difference>
        <ellipsoid label = "outer_dura">
          <origin>  [12.8e-3, 7.2e-3, 40.4e-3] </origin>
          <rx>   9.5e-3    </rx>
          <ry>   7e-3    </ry>
          <rz>   1e12    </rz>
        </ellipsoid>
        <ellipsoid label = "inner_dura">
          <origin>  [12.8e-3, 7.2e-3, 40.4e-3] </origin>
          <rx>   9e-3    </rx>
          <ry>   6.5e-3    </ry>
          <rz>   1e12    </rz>
        </ellipsoid>
      </difference>

      <res>    [2, 2, 1]   </res>
      <velocity>   [0, 0, 0]   </velocity>
      <temperature>  310.15    </temperature>
      <color>    80     </color>
    </geom_object>
  </material>

  <material name="tufnol">

    <constitutive_model type = "comp_linear">
      <E>     6.5e9    </E>
      <v>     0.3     </v>
      <useModifiedEOS>  true    </useModifiedEOS>
      <equation_of_state type="default_hyper">  </equation_of_state>
    </constitutive_model>
```

```
    <density>     1360.0000374123076199 </density>
    <thermal_conductivity>  0      </thermal_conductivity>
    <specific_heat>    1500     </specific_heat>

    <geom_object>
    <!--
    <cylinder label = "short_impactor">
        <bottom> [14e-3, 15e-3, 40.4e-3] </bottom>
        <top> [14e-3, 25e-3, 40.4e-3] </top>
        <radius> 10.00e-3 </radius>
    </cylinder>
    -->
      <union>
        <cylinder label = "impactor_tail">
          <bottom> [12.8e-3, 28.4e-3, 0e-3] </bottom>
          <top> [12.8e-3, 40.4e-3, 0e-3] </top>
          <radius> 7e-3 </radius>
        </cylinder>
        <cylinder label = "impactor_mid">
          <bottom> [12.8e-3, 18.4e-3, 0e-3] </bottom>
          <top> [12.8e-3, 28.4e-3, 0e-3] </top>
          <radius> 10.00e-3 </radius>
        </cylinder>
        <cylinder label = "impactor_head">
          <bottom> [12.8e-3, 15.4e-3, 0e-3] </bottom>
          <top> [12.8e-3, 18.4e-3, 0e-3] </top>
          <radius> 6e-3 </radius>
        </cylinder>
      </union>

      <res>     [2, 2, 1]    </res>
      <velocity>    [0, -4.5, 0]   </velocity>
      <temperature>   310.15     </temperature>
      <color>     40      </color>
    </geom_object>
  </material>

  <material name="backplate">

    <constitutive_model type = "comp_linear">
      <E>      193e9     </E>
      <v>      0.3      </v>
      <useModifiedEOS>  true     </useModifiedEOS>
      <equation_of_state type="default_hyper">  </equation_of_state>
    </constitutive_model>

    <!--<density>     8000.0000025199997253 </density>-->
    <density>     1360.0000374123076199 </density>
    <thermal_conductivity>  0      </thermal_conductivity>
    <!--<specific_heat>    4130     </specific_heat>-->
    <specific_heat>    1500     </specific_heat>

    <geom_object>
      <box label = "backplate">
        <min>    [0,  0,  0]    </min>
        <max>    [12.8e-3, 0.2e-3, 40e-3] </max>
      </box>
      <res>     [2, 2, 1]    </res>
      <velocity>    [0.0, 0.0, 0.0]   </velocity>
```

```
        <temperature>   300     </temperature>
        <color>     60     </color>
      </geom_object>
    </material>

    <contact>
      <type>     friction    </type>
      <materials>    [0, 1, 2, 3]   </materials>
      <mu>     0.0     </mu>
    </contact>
  </MPM>

  <ICE>

    <material name = "air">
      <EOS type = "ideal_gas">    </EOS>
      <dynamic_viscosity>   0.0    </dynamic_viscosity>
      <thermal_conductivity>  0      </thermal_conductivity>
      <!-- isochoric specific heat of air at 310.15K at 5 atm pressure -->
      <!--<specific_heat>    719.6    </specific_heat>-->
      <!-- isochoric specific heat of air at 310.15K at 3 atm pressure -->
      <specific_heat>    719.1    </specific_heat>
      <gamma>     1.4   </gamma>

      <geom_object>
        <difference>
          <box label="whole_domain">
            <min>  [0e-3,   0e-3,  0e-3] </min>
            <max>  [12.8e-3, 50e-3, 40e-3] </max>
            <!--<max>[13.94e-3, 46.92e-3, 0.34e-3]</max>-->
          </box>
          <union>
            <box label = "backplate"></box>
            <cylinder label = "impactor_tail"></cylinder>
            <cylinder label = "impactor_head"></cylinder>
            <cylinder label = "impactor_mid"></cylinder>
            <ellipsoid label = "outer_dura"></ellipsoid>
          </union>
        </difference>

        <res>     [2, 2, 1]    </res>
        <velocity>    [0.0, 0.0, 0.0]   </velocity>
        <temperature>   310.15     </temperature>
                <!--<density>   5.6749715055   </density> -->
                <!-- density at 3 atm  -->
                <density>   3.4142587173926797  </density>
        <pressure>    0    </pressure>
      </geom_object>
    </material>

  <material name = "csf">

      <EOS type="Thomsen_Hartka_water">
        <a>      2.0e-7     </a>
        <b>      2.6      </b>
        <co>     4205.7     </co>
        <ko>     5.0e-10     </ko>
        <To>     277.0     </To>
        <L>      8.0e-6     </L>
        <vo>     1.00008e-3    </vo>
```

```
      </EOS>

      <dynamic_viscosity>   0.000     </dynamic_viscosity>
      <!-- THWAter Script 3 atm -->
            <gamma>  1.0000392285636475  </gamma>
      <thermal_conductivity>  0      </thermal_conductivity>
      <!-- THWAter Script 3 atm -->
      <specific_heat>    4117.8398968398732904 </specific_heat>

      <geom_object>

        <difference>
          <ellipsoid label = "inner_dura"></ellipsoid>
          <ellipsoid label = "cord"></ellipsoid>
        </difference>
        <res>   [2, 2, 1]    </res>
        <velocity>  [0.0, 0.0, 0.0]   </velocity>
        <!-- THWAter Script 3 atm -->
        <density>   991.3235777144317353 </density>
        <pressure>  0      </pressure>
        <temperature>  310.15      </temperature>
        <color>     100      </color>
      </geom_object>
    </material>

  </ICE>

  <exchange_properties>
    <exchange_coefficients>
      <!--
      0) cord_tissue
      1) dura_mater
      2) tufnol
      3) backplate
      4) air
      5) csf

      0->1, 0->2, 0->3, 0->4, 0->5,
           1->2, 1->3, 1->4, 1->5,
                2->3, 2->4, 2->5,
                     3->4, 3->5,
                          4->5,

              0->1,  0->2,  0->3,  0->4,  0->5,  1->2,  1->3,  1->4, 1->5,  2->3,
                    2->4,  2->5,  3->4,  3->5,  4->5,
      -->
      <momentum> [1e10.6,  1e10.6,  1e10.6,  1e10.6,  1e10.6,  1e10.6,  1e10.6,  1e1,
            1e10.6,  1e10.6,  1e10.6,  1e10.6,  1e10.6,  1e10.6,  1e10.6] </
          momentum>
      <!--<momentum> [1e10.6,  0,    0,   0,    1e10.6,  1e10.6,  1e10.6,  1e1,  1e10
          .6,  0,    1e10.6,  0,    1e1,   0,    0] </momentum>-->
      <heat>  [0,   0,    0,    0,    0,    0,    0,    0, 0,   0,    0,    0,    0,   0,   0 ] </heat
          >
    </exchange_coefficients>
  </exchange_properties>
</MaterialProperties>

<Grid>
  <BoundaryConditions>
    <Face side = "x-">
```

```
    <BCType id = "o" label = "Pressure" var = "Neumann">
      <value> 0.0 </value>
    </BCType>
    <BCType id = "all" label = "Velocity" var = "Dirichlet">
      <value> [0, 0, 0] </value>
    </BCType>
    <BCType id = "all" label = "Temperature" var = "Neumann">
      <value> 0.0 </value>
    </BCType>
    <BCType id = "all" label = "Density" var = "Neumann">
      <value> 0.0 </value>
    </BCType>
  </Face>

  <Face side = "x+">
    <BCType id = "all" var = "symmetry" label = "Symmetric"> </BCType>
    <!--<BCType id = "o" label = "Pressure" var = "Neumann">
      <value> 0.0 </value>
    </BCType>
    <BCType id = "all" label = "Velocity" var = "Dirichlet">
      <value> [0, 0, 0] </value>
    </BCType>
    <BCType id = "all" label = "Temperature" var = "Neumann">
      <value> 0.0 </value>
    </BCType>
    <BCType id = "all" label = "Density" var = "Neumann">
      <value> 0.0 </value>
    </BCType>
    -->
  </Face>

  <Face side = "y-">
    <BCType id = "o" label = "Pressure" var = "Neumann">
      <value> 0.0 </value>
    </BCType>
    <BCType id = "all" label = "Velocity" var = "Dirichlet">
      <value> [0, 0, 0] </value>
    </BCType>
    <BCType id = "all" label = "Temperature" var = "Neumann">
      <value> 0.0 </value>
    </BCType>
    <BCType id = "all" label = "Density" var = "Neumann">
      <value> 0.0 </value>
    </BCType>
  </Face>

  <Face side = "y+">
    <BCType id = "o" label = "Pressure" var = "Neumann">
      <value> 0.0 </value>
    </BCType>

    <BCType id = "all" label = "Velocity" var = "Dirichlet">
      <value> [0, 0, 0] </value>
    </BCType>
    <!--
    <BCType id = "all" label = "Velocity" var = "Neumann">
      <value> [0, 0, 0] </value>
    </BCType> -->
    <BCType id = "all" label = "Temperature" var = "Neumann">
      <value> 0.0 </value>
```

```
        </BCType>
        <BCType id = "all" label = "Density" var = "Neumann">
          <value> 0.0 </value>
        </BCType>
      </Face>

      <Face side = "z-">
        <BCType id = "all" var = "symmetry" label = "Symmetric"> </BCType>
      </Face>

      <Face side = "z+">
        <!--
        <BCType id = "all" var = "symmetry" label = "Symmetric"> </BCType>
        -->
        <BCType id = "0" label = "Pressure" var = "Neumann">
          <value> 0.0 </value>
        </BCType>
        <BCType id = "all" label = "Velocity" var = "Dirichlet">
          <value> [0, 0, 0] </value>
        </BCType>
        <BCType id = "all" label = "Temperature" var = "Neumann">
          <value> 0.0 </value>
        </BCType>
        <BCType id = "all" label = "Density" var = "Neumann">
          <value> 0.0 </value>
        </BCType>
      </Face>
    </BoundaryConditions>

    <Level>
      <!--
      <Box label = "Domain">
        <lower>   [0e-3,   0e-3,  0e-3]   </lower>
        <upper>   [13.94e-3, 46.92e-3, 0.34e-3]  </upper>
        <extraCells> [1, 1, 1]      </extraCells>
        <patches>  [12, 1, 1]      </patches>
      </Box>
      <spacing>   [0.34e-3, 0.34e-3, 0.34e-3]  </spacing>
      -->
      <Box label = "Domain">
        <lower>   [0e-3,   0e-3,  0e-3] </lower>
        <upper>   [12.8e-3, 44e-3,  40e-3] </upper>
        <extraCells> [1, 1, 1]      </extraCells>
        <patches>  [16, 1, 40]      </patches>
      </Box>
      <spacing>   [0.4e-3, 0.4e-3, 0.4e-3]  </spacing>

    </Level>
  </Grid>
</Uintah_specification>

<!--
mpirun -np 12 /usr/not-backed-up/udev5_3/opt/StandAlone/sus -mpi /usr/
    not-backed-up/csf1.7.1_3D_3ATM_P2.ups 2>&1 | tee /usr/not-backed-up
    /csf1.7.1_3D_3ATM_P2.txt
-->
```

## A.2 CONSTITUTIVE MATERIAL MODEL CODE

### A.2.1 *Ogden Model (Cord)*

Listing 1: CompOgdenStandard.h

```
#ifndef __COMP_OGDEN_STANDARD_CONSTITUTIVE_MODEL_H__
  #define __COMP_OGDEN_STANDARD_CONSTITUTIVE_MODEL_H__

#include <CCA/Components/MPM/ConstitutiveModel/ConstitutiveModel.h>
#include "../PlasticityModels/MPMEquationOfState.h"
#include <Core/Math/Matrix3.h>
#include <Core/ProblemSpec/ProblemSpecP.h>
#include <CCA/Ports/DataWarehouseP.h>

#include <cmath>

namespace Uintah {

  class MPMLabel;
  class MPMFlags;

  class CompOgdenStandard : public ConstitutiveModel {

    private:

      bool d_useModifiedEOS; // use modified equation of state

    protected:

      bool d_useInitialStress; // Initial stress state
      double d_init_pressure; // Initial pressure

      MPMEquationOfState* d_eos; // Equation of State factory

    public:

      struct CMData {
        double K; // Bulk Modulus
        double alpha; // material constant
        double mu; // material constant
      };

      const VarLabel* bElBarLabel;
      const VarLabel* bElBarLabel_preReloc;

    private:

      CMData d_initialData;

      // Prevent copying of this class
      CompOgdenStandard& operator=(const CompOgdenStandard &cm);

      // Calculate the bulk modulus based on the Ogden material constants
      double CalculateBulkModulus();

    public:
```

```
    // Constructor
    CompOgdenStandard(ProblemSpecP& ps, MPMFlags* flag);

    // Copy constructor
    CompOgdenStandard(const CompOgdenStandard* cm);

    // Destructor
    virtual ~CompOgdenStandard();

    // Clone
    CompOgdenStandard* clone();

    virtual void outputProblemSpec(ProblemSpecP& ps,bool output_cm_tag =
        true);

    // initialize each particle's constitutive model data
    virtual void initializeCMData(const Patch* patch, const MPMMaterial* matl,
        DataWarehouse* new_dw);

    // Keeps track of the particles and the related variables as particles move
        from patch to patch
    virtual void addParticleState(std::vector<const VarLabel*>& from, std::vector
        <const VarLabel*>& to);

    // Tells the scheduler what data needs to be available at the time
        computeStressTensor(...) is called
    virtual void addComputesAndRequires(Task* task, const MPMMaterial* matl
        , const PatchSet* patches) const;

    virtual void addComputesAndRequires(Task* task, const MPMMaterial* matl
        , const PatchSet* patches, const bool recurse, const bool SchedParent)
        const;

    virtual void addInitialComputesAndRequires(Task* task, const MPMMaterial
        * matl, const PatchSet*) const;

    // Carry forward constitutive model data for RigidMPM
    virtual void carryForward(const PatchSubset* patches, const MPMMaterial*
        matl, DataWarehouse* old_dw, DataWarehouse* new_dw);

    // Calculate the initial timestep, after this the timestep (dT) will be
        calculated by computeStressTensor(...)
    virtual void computeStableTimestep(const Patch* patch, const MPMMaterial*
         matl, DataWarehouse* new_dw);

    // Compute cauchy stress for each particle in the patch
    virtual void computeStressTensor(const PatchSubset* patches, const
        MPMMaterial* matl, DataWarehouse* old_dw, DataWarehouse*
        new_dw);

    virtual double computeRhoMicroCM(double pressure, const double p_ref,
        const MPMMaterial* matl, double temperature, double rho_guess);

    virtual void computePressEOSCM(double rho_m, double& press_eos, double
         p_ref, double& dp_drho, double& ss_new, const MPMMaterial* matl,
        double temperature);

    virtual double getCompressibility();
};
} // End namespace Uintah
```

```
#endif // __COMP_OGDEN_STANDARD_CONSTITUTIVE_MODEL_H__
```

Listing 2: CompOgdenStandard.cc

```
/*
 * The MIT License
 *
 * Copyright (c) 1997–2012 The University of Utah
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 * of this software and associated documentation files (the "Software"), to
 * deal in the Software without restriction, including without limitation the
 * rights to use, copy, modify, merge, publish, distribute, sublicense, and/or
 * sell copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY
      KIND, EXPRESS OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
      MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO
      EVENT SHALL THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM,
      DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR
      OTHERWISE, ARISING
 * FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE
      OR OTHER DEALINGS
 * IN THE SOFTWARE.
 */

#include <CCA/Components/MPM/ConstitutiveModel/UoL/
    CompOgdenStandard.h>
#include <CCA/Components/MPM/ConstitutiveModel/MPMMaterial.h>
#include <CCA/Components/MPM/ConstitutiveModel/PlasticityModels/
    MPMEquationOfStateFactory.h>
#include <CCA/Components/MPM/MPMFlags.h>
#include <Core/Math/Matrix3.h>
#include <CCA/Ports/DataWarehouse.h>
#include <Core/Grid/Variables/VarLabel.h>
#include <Core/Grid/Variables/ParticleVariable.h>
#include <Core/Grid/Variables/NCVariable.h>
#include <Core/Grid/Patch.h>
#include <Core/Grid/Variables/VarTypes.h>
#include <Core/Labels/MPMLabel.h>
#include <Core/Math/FastMatrix.h>
#include <Core/Exceptions/ParameterNotFound.h>
#include <Core/Exceptions/InvalidValue.h>
#include <Core/Exceptions/ConvergenceFailure.h>
#include <Core/Malloc/Allocator.h>
#include <cmath>
#include <iostream>
#include <stdio.h>
#include <iomanip>

using namespace Uintah;
```

```
using namespace std;

CompOgdenStandard::CompOgdenStandard(ProblemSpecP& ps, MPMFlags*
      Mflag) : ConstitutiveModel(Mflag)
{
  // required - material parameters for the constitutive model
  ps->require("bulk_modulus", d_initialData.K);
  ps->require("alpha_1", d_initialData.alpha);
  ps->require("mu_1", d_initialData.mu);

  // optional - use modified equation of state (off by default)
  d_useModifiedEOS = false;
  ps->get("useModifiedEOS",d_useModifiedEOS); // no negative pressure for
       solids

  // Initial stress
  // Fix: Need to make it more general. Add gravity turn-on option and
  // read from file option etc.
  ps->getWithDefault("useInitialStress", d_useInitialStress, false);
  d_init_pressure = 0.0;
  if (d_useInitialStress) {
    ps->getWithDefault("initial_pressure", d_init_pressure, 0.0);
  }

  // Equation of state factory for pressure (default is DefaultHyperEOS)
  d_eos = MPMEquationOfStateFactory::create(ps);

  d_eos->setBulkModulus(d_initialData.K);

  if(!d_eos){
    ostringstream desc;
    desc << "An error occured in the MPM EquationOfStateFactory that has \n"
    << " slipped through the existing bullet proofing. Please check and correct." <<
         endl;
    throw ParameterNotFound(desc.str(), __FILE__, __LINE__);
  }

  // Universal Labels
  bElBarLabel = VarLabel::create("p.bElBar", ParticleVariable<Matrix3>::
       getTypeDescription());
  bElBarLabel_preReloc = VarLabel::create("p.bElBar+", ParticleVariable<Matrix3>::
       getTypeDescription());

}

CompOgdenStandard::CompOgdenStandard(const CompOgdenStandard* cm) :
      ConstitutiveModel(cm)
{
  d_initialData.K = cm->d_initialData.K;
  d_initialData.alpha = cm->d_initialData.alpha;
  d_initialData.mu = cm->d_initialData.mu;
  d_useModifiedEOS = cm->d_useModifiedEOS;

  // Initial stress
  d_useInitialStress = cm->d_useInitialStress;
  if (d_useInitialStress) {
    d_init_pressure = cm->d_init_pressure;
  }

  // Equation of state factory for pressure (default is DefaultHyperEOS)
```

```
  d_eos = MPMEquationOfStateFactory::createCopy(cm->d_eos);

  d_eos->setBulkModulus(d_initialData.K);

  if(!d_eos){
    ostringstream desc;
    desc << "An error occured in the MPM EquationOfStateFactory that has \n"
    << " slipped through the existing bullet proofing. Please check and correct." <<
         endl;
    throw ParameterNotFound(desc.str(), __FILE__, __LINE__);
  }

  // Universal Labels
  bElBarLabel = VarLabel::create("p.bElBar", ParticleVariable<Matrix3>::
       getTypeDescription());
  bElBarLabel_preReloc = VarLabel::create("p.bElBar+", ParticleVariable<Matrix3>::
       getTypeDescription());
}

CompOgdenStandard::~CompOgdenStandard()
{
  // Universal Deletes
  VarLabel::destroy(bElBarLabel);
  VarLabel::destroy(bElBarLabel_preReloc);
}

CompOgdenStandard* CompOgdenStandard::clone()
{
  return scinew CompOgdenStandard(*this);
}

void CompOgdenStandard::outputProblemSpec(ProblemSpecP& ps, bool
     output_cm_tag)
{
  ProblemSpecP cm_ps = ps;
  if (output_cm_tag) {
    cm_ps = ps->appendChild("constitutive_model");
    cm_ps->setAttribute("type","comp_ogden_standard");
  }
  cm_ps->appendElement("bulk_modulus", d_initialData.K);
  cm_ps->appendElement("alpha_1", d_initialData.alpha);
  cm_ps->appendElement("mu_1", d_initialData.mu);
  cm_ps->appendElement("useModifiedEOS", d_useModifiedEOS);
}

void CompOgdenStandard::initializeCMData(const Patch* patch, const
     MPMMaterial* matl, DataWarehouse* new_dw)
{
  // Put stuff in here to initialize each particle's
  // constitutive model parameters and deformationMeasure
  Matrix3 Identity;
  Identity.Identity();
  Matrix3 zero(0.0);

  ParticleSubset* pset = new_dw->getParticleSubset(matl->getDWIndex(), patch);

  ParticleSubset::iterator iterUniv = pset->begin();

  // Initialize the variables shared by all constitutive models
  // This method is defined in the ConstitutiveModel base class.
```

```
  initSharedDataForExplicit(patch, matl, new_dw);

  // calcualte the initial timestep, after this the timestep will be calculated by
       computeStressTensor
  computeStableTimestep(patch, matl, new_dw);


  // Universal
  ParticleVariable<Matrix3> deformationGradient, pstress, bElBar;

  new_dw->allocateAndPut(bElBar, bElBarLabel, pset);

  for(;iterUniv != pset->end(); iterUniv++){
  bElBar[*iterUniv] = Identity;
  }
}

void CompOgdenStandard::addParticleState(std::vector<const VarLabel*>& from,
      std::vector<const VarLabel*>& to)
{
  // Keeps track of the particles and the related variables as particles move from
       patch to patch (each CM adds its own state variables, e.g. failure, damage,
       plasticity, etc.)
  // Universal
  from.push_back(bElBarLabel);
  to.push_back(bElBarLabel_preReloc);
}

void CompOgdenStandard::addInitialComputesAndRequires(Task* task, const
      MPMMaterial* matl, const PatchSet*) const
{
  const MaterialSubset* matlset = matl->thisMaterial();
  // Universal
  task->computes(bElBarLabel, matlset);
}

// Tells the scheduler what data needs to be available at the time
      computeStressTensor(...) is called
void CompOgdenStandard::addComputesAndRequires(Task* task, const
      MPMMaterial* matl, const PatchSet* patches) const
{
  const MaterialSubset* matlset = matl->thisMaterial();

  // Add the computes and requires that are common to all explicit constitutive
       models.
  // This method is defined in the ConstitutiveModel base class.
  addSharedCRForExplicit(task, matlset, patches);

  // Other constitutive model and input dependent computes and requires
  Ghost::GhostType gnone = Ghost::None;

    task->requires(Task::OldDW, lb->pParticleIDLabel, matlset, gnone);

  if(flag->d_with_color) {
    task->requires(Task::OldDW, lb->pColorLabel, Ghost::None);
  }

  // Universal
  task->requires(Task::OldDW, bElBarLabel, matlset, gnone);
  task->computes(bElBarLabel_preReloc,  matlset);
```

```
}

void CompOgdenStandard::addComputesAndRequires(Task* task, const
      MPMMaterial* matl, const PatchSet* patches, const bool recurse, const bool
      SchedParent) const
{
  const MaterialSubset* matlset = matl->thisMaterial();

  Ghost::GhostType gnone = Ghost::None;

  if(SchedParent){
    task->requires(Task::ParentOldDW, bElBarLabel, matlset, gnone);
  }
  else{
    task->requires(Task::OldDW, bElBarLabel, matlset, gnone);
  }
}

// Carry forward CM data for RigidMPM
void CompOgdenStandard::carryForward(const PatchSubset* patches, const
      MPMMaterial* matl, DataWarehouse* old_dw, DataWarehouse* new_dw)
{
#if 0
  for(int p=0; p<patches->size(); p++){
    const Patch* patch = patches->get(p);
    int dwi = matl->getDWIndex();
    ParticleSubset* pset = old_dw->getParticleSubset(dwi, patch);

    // Carry forward the data common to all constitutive models when using
         RigidMPM.
    // This method is defined in the ConstitutiveModel base class.
    carryForwardSharedData(pset, old_dw, new_dw, matl);

    // Carry forward the data local to this constitutive model
    new_dw->put(delt_vartype(1.e10), lb->delTLabel, patch->getLevel());
    if (flag->d_reductionVars->accStrainEnergy || flag->d_reductionVars->
         strainEnergy) {
      new_dw->put(sum_vartype(0.0), lb->StrainEnergyLabel);
    }
  }
#endif
  for(int p=0;p<patches->size();p++){
    const Patch* patch = patches->get(p);
    int dwi = matl->getDWIndex();
    ParticleSubset* pset = old_dw->getParticleSubset(dwi, patch);

    // Carry forward the data common to all constitutive models
    // when using RigidMPM.
    // This method is defined in the ConstitutiveModel base class.
    carryForwardSharedData(pset, old_dw, new_dw, matl);

    // Carry forward the data local to this constitutive model
    // Universal
    ParticleVariable<Matrix3> bElBar_new;
    constParticleVariable<Matrix3> bElBar;
    old_dw->get(bElBar, bElBarLabel, pset);
    new_dw->allocateAndPut(bElBar_new, bElBarLabel_preReloc, pset);
    for(ParticleSubset::iterator iter = pset->begin();
      iter != pset->end(); iter++){
      particleIndex idx = *iter;
```

```
      bElBar_new[idx] = bElBar[idx];
    }
    new_dw->put(delt_vartype(1.e10), lb->delTLabel, patch->getLevel());
    if (flag->d_reductionVars->accStrainEnergy ||
      flag->d_reductionVars->strainEnergy) {
      new_dw->put(sum_vartype(0.), lb->StrainEnergyLabel);
    }
  } // End Particle Loop

}

// Calculate the initial timestep, after this the timestep (dT) will be calculated by
      computeStressTensor(...)
// The size of the timestep depends on cell spacing, velocity of the particle, and
      the material wavespeed (c_dil) at each particle
// A reduction over all dTs from every patch is performed, the smallest dT is used
void CompOgdenStandard::computeStableTimestep(const Patch* patch, const
      MPMMaterial* matl, DataWarehouse* new_dw)
{
  Vector dx = patch->dCell();
  int dwi = matl->getDWIndex();

  // Retrieve the array of constitutive parameters
  ParticleSubset* pset = new_dw->getParticleSubset(dwi, patch);
  constParticleVariable<double> pmass, pvolume;
  constParticleVariable<Vector> pvelocity;

  new_dw->get(pmass, lb->pMassLabel, pset);
  new_dw->get(pvolume, lb->pVolumeLabel, pset);
  new_dw->get(pvelocity, lb->pVelocityLabel, pset);

  double c_dil = 0.0; // local speed of sound
  Matrix3 F; // Deformation gradient
  Matrix3 B; // Left Cauchy Green deformation tensor
  Matrix3 Identity;
  Identity.Identity(); // 3x3 Identity matrix
  Vector WaveSpeed(1.e-12, 1.e-12, 1.e-12);
  Vector eVals(0, 0, 0); // Eigenvalues
  Matrix3 eVecs; // Eigenvectors
  double a = d_initialData.alpha; // Ogden material constant
  double mu = d_initialData.mu; // Ogden material constant
  double l1; // first principal stretch

  // Assuming a stress free reference configuration F = Identity
  F = Identity;

  // Compute the left Cauchy-Green deformation tensor
    B = F*F.Transpose();

  // The principal stretches can be calculated from the square roots of the
       Eigenvlaues of B
  B.eigen(eVals, eVecs);

  // Only the first principal stretch (the biggest) is needed
  l1 = sqrt(eVals.x());

  // Compute wave speed at each particle, store the maximum
  for(ParticleSubset::iterator iter = pset->begin(); iter != pset->end(); iter++){
    particleIndex idx = *iter;
```

```
    // Compute c_dil, the local speed of sound
    c_dil = sqrt(
      mu*(a * pow(l1, a - 1.0) + 0.5 * a * pow(l1, -0.5 * a - 1.0))/(pmass[idx]/
          pvolume[idx])
    );

    // Compute wave speed + particle velocity at each particle, then store the
        maximum
    WaveSpeed=Vector(Max(c_dil+fabs(pvelocity[idx].x()), WaveSpeed.x()),
                 Max(c_dil+fabs(pvelocity[idx].y()), WaveSpeed.y()),
               Max(c_dil+fabs(pvelocity[idx].z()), WaveSpeed.z()));
  }

  WaveSpeed = dx/WaveSpeed;
  double delT_new = WaveSpeed.minComponent();
  new_dw->put(delt_vartype(delT_new), lb->delTLabel, patch->getLevel());
}

// Computes the Cauchy stress for each particle for the given material
// Called once per timestep (for each material)
void CompOgdenStandard::computeStressTensor(const PatchSubset* patches, const
      MPMMaterial* matl, DataWarehouse* old_dw, DataWarehouse* new_dw)
{
  // Loop over each patch
  for(int patch_idx = 0; patch_idx < patches->size(); patch_idx++){
    const Patch* patch = patches->get(patch_idx);

    Matrix3 F; // Deformation gradient
    Matrix3 B; // Left Cauchy Green deformation tensor
    Matrix3 Identity; Identity.Identity(); // 3x3 Identity matrix
    Vector WaveSpeed(1.e-12, 1.e-12, 1.e-12);
    Vector eVals(0, 0, 0); // Eigenvalues
    Matrix3 eVecs; // Eigenvectors
    Matrix3 n1, n2, n3; // Outer product of Eigenvectors
    double c_dil = 0.0; // Local speed of sound
    double se = 0.0; // Accumulated strain energy for all particles
    double J; // Jacobian of deformation gradient
    double l1, l2, l3; // The principl stretches, lambda_1, lambda_2, lambda_3
    double a = 0.0; // Ogden material constant
    double mu = 0.0; // Ogden material constant
    double K; // Bulk modulus (material constant)
    double rho_current; // Current density
    double rho_orig; // Original density
    double athird, pow_l1l2_athird, pow_l1l3_athird, pow_l2l3_athird; // Temp
        variables for stress calculation

      Matrix3 bElBarTrial(0.0), pDefGradInc(0.0), fBar(0.0); ///
    double Jinc = 0.0;

    ParticleInterpolator* interpolator = flag->d_interpolator->clone(patch);
    vector<IntVector> ni(interpolator->size());
    vector<Vector> d_S(interpolator->size());
    vector<double> S(interpolator->size());

    Vector dx = patch->dCell();

    // DataWarehouse index
    int dwi = matl->getDWIndex();

    // Create array for the particle position
```

```
ParticleSubset* pset = old_dw->getParticleSubset(dwi, patch);
constParticleVariable<Point> px;
constParticleVariable<Matrix3> deformationGradient_new;
constParticleVariable<Matrix3> deformationGradient;
ParticleVariable<Matrix3> pstress;
constParticleVariable<double> pmass;
constParticleVariable<double> pvolume_new;
constParticleVariable<Vector> pvelocity;
constParticleVariable<Matrix3> velGrad;
constParticleVariable<Matrix3> psize;
ParticleVariable<double> pdTdt,p_q;

constParticleVariable<Matrix3> pDefGrad, bElBar; ///
ParticleVariable<Matrix3> pStress, bElBar_new; ///

delt_vartype delT;
old_dw->get(delT,       lb->delTLabel, getLevel(patches));
old_dw->get(pvelocity,   lb->pVelocityLabel, pset);
old_dw->get(deformationGradient,   lb->pDeformationMeasureLabel, pset);
old_dw->get(bElBar,      bElBarLabel, pset); ///

new_dw->allocateAndPut(pstress, lb->pStressLabel_preReloc, pset);
new_dw->allocateAndPut(pdTdt, lb->pdTdtLabel, pset);
new_dw->allocateAndPut(p_q, lb->p_qLabel_preReloc, pset);

new_dw->get(pvolume_new,    lb->pVolumeLabel_preReloc, pset);
new_dw->get(velGrad,    lb->pVelGradLabel_preReloc, pset);
new_dw->get(deformationGradient_new, lb->
      pDeformationMeasureLabel_preReloc, pset);

new_dw->allocateAndPut(bElBar_new,  bElBarLabel_preReloc, pset); ///

// Get initial data
a = d_initialData.alpha;
mu = d_initialData.mu;
K = d_initialData.K;
rho_orig = matl->getInitialDensity();
athird = 0.33333333333333333 * a;

for(ParticleSubset::iterator iter = pset->begin();iter!=pset->end();iter++){
  particleIndex idx = *iter;

  // Assign zero internal heating by default - modify if necessary.
  pdTdt[idx] = 0.0;

  // Get deformation gradient
  F = deformationGradient_new[idx];

  ///
  // Calculate bElBar (needed for interaction with RigidMPM)
  pDefGradInc = deformationGradient_new[idx]*deformationGradient[idx].
        Inverse();
  Jinc = pDefGradInc.Determinant();

  // Get the volume preserving part of the deformation gradient increment
  fBar = pDefGradInc/cbrt(Jinc);

  // Compute the elastic part of the volume preserving
  // part of the left Cauchy-Green deformation tensor
  bElBar_new[idx] = fBar*bElBar[idx]*fBar.Transpose();
```

```
    // get the original volumetric part of the deformation
    J = F.Determinant();

    // Compute the left Cauchy-Green deformation tensor
        B = F*F.Transpose();

    // Error checking
    if (J < 0.0) {
      throw InvalidValue("Negative Jacobian of deformation gradient", __FILE__,
            __LINE__);
    }

    // The Ogden model is typically expressed in terms of the principal
        stretches rather than the invariants
    // The principal stretches can be calculated from the square roots of the
        Eigenvlaues of B
    B.eigen(eVals, eVecs);

    // Calcualte the principal stretches
    l1 = sqrt(eVals.x());
    l2 = sqrt(eVals.y());
    l3 = sqrt(eVals.z());

    // Put the Eigenvectors in matrix form for convenience
    n1.set(0,0,eVecs(0,0));
    n1.set(1,0,eVecs(1,0));
    n1.set(2,0,eVecs(2,0));
    n1.set(0,1,0);
    n1.set(1,1,0);
    n1.set(2,1,0);
    n1.set(0,2,0);
    n1.set(1,2,0);
    n1.set(2,2,0);
    n2.set(0,0,eVecs(0,1));
    n2.set(1,0,eVecs(1,1));
    n2.set(2,0,eVecs(2,1));
    n2.set(0,1,0);
    n2.set(1,1,0);
    n2.set(2,1,0);
    n2.set(0,2,0);
    n2.set(1,2,0);
    n2.set(2,2,0);
    n3.set(0,0,eVecs(0,2));
    n3.set(1,0,eVecs(1,2));
    n3.set(2,0,eVecs(2,2));
    n3.set(0,1,0);
    n3.set(1,1,0);
    n3.set(2,1,0);
    n3.set(0,2,0);
    n3.set(1,2,0);
    n3.set(2,2,0);

    // Calculate the outer product of the Eigenvectors, results in a 3x3 matrix
    Matrix3 N1 = n1*n1.Transpose();
    Matrix3 N2 = n2*n2.Transpose();
    Matrix3 N3 = n3*n3.Transpose();
/*
    double dW_dl1, dW_dl2, dW_dl3; // The strain energy function partially
        differentiated by lambda_1, lambda_2, lambda_3
```

```
        double scalar1, scalar2, scalar3;
        double J1_3 = pow(J, -0.1e1 / 0.3e1);
        double J4_3 = pow(J, -0.4e1 / 0.3e1);
        dW_dl1 = mu / a * (pow(l1 * J1_3, a) * a * (J1_3 - l1 * J4_3 * l2 * l3 / 0.3e1) /
            l1 * pow(J, 0.1e1 / 0.3e1) - pow(l2 * J1_3, a) * a / l1 / 0.3e1 - pow(l3 *
            J1_3, a) * a / l1 / 0.3e1) + K * (J - 0.1e1) * l2 * l3;
        dW_dl2 = mu / a * (-pow(l1 * J1_3, a) * a / l2 / 0.3e1 + pow(l2 * J1_3, a) * a
            * (J1_3 - l1 * J4_3 * l2 * l3 / 0.3e1) / l2 * pow(J, 0.1e1 / 0.3e1) - pow(l3 *
             J1_3, a) * a / l2 / 0.3e1) + K * (J - 0.1e1) * l1 * l3;
        dW_dl3 = mu / a * (-pow(l1 * J1_3, a) * a / l3 / 0.3e1 - pow(l2 * J1_3, a) * a
            / l3 / 0.3e1 + pow(l3 * J1_3, a) * a * (J1_3 - l1 * J4_3 * l2 * l3 / 0.3e1) / l3
             * pow(J, 0.1e1 / 0.3e1)) + K * (J - 0.1e1) * l1 * l2;
        scalar1 = (l1 / J) * dW_dl1;
        scalar2 = (l2 / J) * dW_dl2;
        scalar3 = (l3 / J) * dW_dl3;
        // Calculate Cauchy stress for particle
        pstress[idx] = scalar1*N1 + scalar2*N2 + scalar3*N3;
*/

        // Calculate Cauchy stress for particle
        // Better optimised for performance
        pow_l1l2_athird = pow(l1/l2,athird);
        pow_l1l3_athird = pow(l1/l3,athird);
        pow_l2l3_athird = pow_l1l3_athird/pow_l1l2_athird;
        pstress[idx] = mu/(3.0*J)*( (N1+N1-N2-N3)* pow_l1l2_athird*
            pow_l1l3_athird
                + (N2+N2-N3-N1)* pow_l2l3_athird/pow_l1l2_athird
                + (N3+N3-N1-N2)/(pow_l1l3_athird*pow_l2l3_athird))
                + K*(J-1.0)*(N1+N2+N3);

        // Calculate total strain energy over all particles
        //se = mu / a * (pow(l1 * pow(J, 0.1e1 / 0.3e1), a) + pow(l2 * pow(J, 0.1e1 /
            0.3e1), a) + pow(l3 * pow(J, 0.1e1 / 0.3e1), a) - 0.3e1) + K * pow(J - 0.1
            e1, 0.2e1) / 0.2e1;
        se = 0;

        // Get the current density
        rho_current = rho_orig/J;

        // Compute c_dil, the local speed of sound, this came from Ogden's book:
            Non-Linear Elastic Deformations
        c_dil = sqrt(
          mu*(a * pow(l1, a - 1.0) + 0.5 * a * pow(l1, - 0.5 * a - 1.0))/rho_current
        );

        // Compute wave speed + particle velocity at each particle, then store the
            maximum
        WaveSpeed=Vector(Max(c_dil+fabs(pvelocity[idx].x()), WaveSpeed.x()),
                         Max(c_dil+fabs(pvelocity[idx].y()), WaveSpeed.y()),
                        Max(c_dil+fabs(pvelocity[idx].z()), WaveSpeed.z()));

        // Compute artificial viscosity term
        if (flag->d_artificial_viscosity) {
          throw InvalidValue("Artificial viscosity has not been implemented for this
              constitutive model", __FILE__, __LINE__);
          // To be added later
        }
        else{
          p_q[idx] = 0.0;
        }
```

```
    } // end loop over particles

    WaveSpeed = dx/WaveSpeed;
    double delT_new = WaveSpeed.minComponent();

    new_dw->put(delt_vartype(delT_new), lb->delTLabel, patch->getLevel());

    if (flag->d_reductionVars->accStrainEnergy || flag->d_reductionVars->
        strainEnergy) {
      new_dw->put(sum_vartype(se), lb->StrainEnergyLabel);
    }
    delete interpolator;
  }
}

double CompOgdenStandard::computeRhoMicroCM(double pressure, const
     double p_ref, const MPMMaterial* matl, double temperature, double
     rho_guess)
{
  double p_gauge = pressure - p_ref;
  double rho_orig = matl->getInitialDensity();
  double rho_cur = -1.0;
  bool error = false;

  if (d_useModifiedEOS && p_gauge < 0.0) { // Modified EOS

    double K = d_initialData.K;
    double A = p_ref;
    double n = p_ref/K;
    rho_cur = rho_orig*pow(pressure/A,n);

  } else { // Standard EOS

    try {
      rho_cur = d_eos->computeDensity(rho_orig, -p_gauge);
    } catch (ConvergenceFailure& e) {
      cout << e.message() << endl;
      error = true;
    }
    if (error || rho_cur < 0.0 || isnan(rho_cur)) {
      ostringstream desc;
      desc << "rho_cur = " << rho_cur << " pressure = " << -p_gauge
           << " p_ref = " << p_ref << " 1/sp_vol_CC = " << rho_guess << endl;
      throw InvalidValue(desc.str(), __FILE__, __LINE__);
    }
  }

    return rho_cur;
}

void CompOgdenStandard::computePressEOSCM(const double rho_cur,double&
     pressure, const double p_ref, double& dp_drho, double& cSquared, const
     MPMMaterial* matl, double temperature)
{
  double rho_orig = matl->getInitialDensity();

  if (d_useModifiedEOS && rho_cur < rho_orig) { // Modified EOS

    double K = d_initialData.K;
    double A = p_ref;
```

```
    double n = K/p_ref;
    double rho_rat_to_the_n = pow(rho_cur/rho_orig,n);
    pressure = A*rho_rat_to_the_n;
    dp_drho = (K/rho_cur)*rho_rat_to_the_n;
    cSquared = dp_drho; // Speed of sound squared

  } else { // Standard EOS

    double p = 0.0;
    d_eos->computePressure(rho_orig, rho_cur, p, dp_drho, cSquared);
    pressure = -p + p_ref;
    dp_drho = -dp_drho;
  }

}

double CompOgdenStandard::getCompressibility()
{
  return 1.0/d_initialData.K;
}
```

### A.2.2 *Ogden Model (Dura)*

Listing 3: CompOgdenStorakers.h

```
#ifndef __COMP_OGDEN_STORAKERS_CONSTITUTIVE_MODEL_H__
  #define __COMP_OGDEN_STORAKERS_CONSTITUTIVE_MODEL_H__

#include <CCA/Components/MPM/ConstitutiveModel/ConstitutiveModel.h>
#include "../PlasticityModels/MPMEquationOfState.h"
#include <Core/Math/Matrix3.h>
#include <Core/ProblemSpec/ProblemSpecP.h>
#include <CCA/Ports/DataWarehouseP.h>

#include <cmath>

namespace Uintah {

  class MPMLabel;
  class MPMFlags;

  class CompOgdenStorakers : public ConstitutiveModel {

    private:

      bool d_useModifiedEOS; // use modified equation of state

    protected:

      bool d_useInitialStress; // Initial stress state
      double d_init_pressure; // Initial pressure

      MPMEquationOfState* d_eos; // Equation of State factory

    public:

      struct CMData {
        double PR; //Poisson's Ratio
```

```
    double alpha; // material constant
    double mu; // material constant
  };

  const VarLabel* bElBarLabel;
  const VarLabel* bElBarLabel_preReloc;

private:

  CMData d_initialData;

  // Prevent copying of this class
  CompOgdenStorakers& operator=(const CompOgdenStorakers &cm);

  // Calculate the bulk modulus based on the Ogden material constants
  double CalculateBulkModulus();

public:

  // Constructor
  CompOgdenStorakers(ProblemSpecP& ps, MPMFlags* flag);

  // Copy constructor
  CompOgdenStorakers(const CompOgdenStorakers* cm);

  // Destructor
  virtual ~CompOgdenStorakers();

  // Clone
  CompOgdenStorakers* clone();

  virtual void outputProblemSpec(ProblemSpecP& ps,bool output_cm_tag =
       true);

  // initialize each particle's constitutive model data
  virtual void initializeCMData(const Patch* patch, const MPMMaterial* matl,
       DataWarehouse* new_dw);

  // Keeps track of the particles and the related variables as particles move
       from patch to patch
  virtual void addParticleState(std::vector<const VarLabel*>& from, std::vector
       <const VarLabel*>& to);

  // Tells the scheduler what data needs to be available at the time
       computeStressTensor(...) is called
  virtual void addComputesAndRequires(Task* task, const MPMMaterial* matl
       , const PatchSet* patches) const;

  virtual void addComputesAndRequires(Task* task, const MPMMaterial* matl
       , const PatchSet* patches, const bool recurse, const bool SchedParent)
       const;

  virtual void addInitialComputesAndRequires(Task* task, const MPMMaterial
       * matl, const PatchSet*) const;

  // Carry forward constitutive model data for RigidMPM
  virtual void carryForward(const PatchSubset* patches, const MPMMaterial*
       matl, DataWarehouse* old_dw, DataWarehouse* new_dw);
```

```
        // Calculate the initial timestep, after this the timestep (dT) will be
            calculated by computeStressTensor(...)
        virtual void computeStableTimestep(const Patch* patch, const MPMMaterial*
            matl, DataWarehouse* new_dw);

        // Compute cauchy stress for each particle in the patch
        virtual void computeStressTensor(const PatchSubset* patches, const
            MPMMaterial* matl, DataWarehouse* old_dw, DataWarehouse*
            new_dw);

        virtual double computeRhoMicroCM(double pressure, const double p_ref,
            const MPMMaterial* matl, double temperature, double rho_guess);

        virtual void computePressEOSCM(double rho_m, double& press_eos, double
            p_ref, double& dp_drho, double& ss_new, const MPMMaterial* matl,
            double temperature);

        virtual double getCompressibility();

  };
} // End namespace Uintah

#endif // __COMP_OGDEN_STORAKERS_CONSTITUTIVE_MODEL_H__
```

Listing 4: CompOgdenStorakers.cc

```
/*
 * The MIT License
 *
 * Copyright (c) 1997–2012 The University of Utah
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 * of this software and associated documentation files (the "Software"), to
 * deal in the Software without restriction, including without limitation the
 * rights to use, copy, modify, merge, publish, distribute, sublicense, and/or
 * sell copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY
      KIND, EXPRESS OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
      MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO
      EVENT SHALL THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM,
      DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR
      OTHERWISE, ARISING
 * FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE
      OR OTHER DEALINGS
 * IN THE SOFTWARE.
 */

#include <CCA/Components/MPM/ConstitutiveModel/UoL/
    CompOgdenStorakers.h>
#include <CCA/Components/MPM/ConstitutiveModel/MPMMaterial.h>
```

```
#include <CCA/Components/MPM/ConstitutiveModel/PlasticityModels/
    MPMEquationOfStateFactory.h>
#include <CCA/Components/MPM/MPMFlags.h>
#include <Core/Math/Matrix3.h>
#include <CCA/Ports/DataWarehouse.h>
#include <Core/Grid/Variables/VarLabel.h>
#include <Core/Grid/Variables/ParticleVariable.h>
#include <Core/Grid/Variables/NCVariable.h>
#include <Core/Grid/Patch.h>
#include <Core/Grid/Variables/VarTypes.h>
#include <Core/Labels/MPMLabel.h>
#include <Core/Math/FastMatrix.h>
#include <Core/Exceptions/ParameterNotFound.h>
#include <Core/Exceptions/InvalidValue.h>
#include <Core/Exceptions/ConvergenceFailure.h>
#include <Core/Malloc/Allocator.h>
#include <cmath>
#include <iostream>
#include <stdio.h>
#include <iomanip>

using namespace Uintah;
using namespace std;

CompOgdenStorakers::CompOgdenStorakers(ProblemSpecP& ps, MPMFlags*
    Mflag) : ConstitutiveModel(Mflag)
{
  // required - material parameters for the constitutive model
  ps->require("PR", d_initialData.PR);
  ps->require("alpha_1", d_initialData.alpha);
  ps->require("mu_1", d_initialData.mu);

  // optional - use modified equation of state (off by default)
  d_useModifiedEOS = false;
  ps->get("useModifiedEOS",d_useModifiedEOS); // no negative pressure for
      solids

  // Initial stress
  // Fix: Need to make it more general. Add gravity turn-on option and
  // read from file option etc.
  ps->getWithDefault("useInitialStress", d_useInitialStress, false);
  d_init_pressure = 0.0;
  if (d_useInitialStress) {
    ps->getWithDefault("initial_pressure", d_init_pressure, 0.0);
  }

  // Equation of state factory for pressure (default is DefaultHyperEOS)
  d_eos = MPMEquationOfStateFactory::create(ps);

  d_eos->setBulkModulus(CalculateBulkModulus());

  if(!d_eos){
    ostringstream desc;
    desc << "An error occured in the MPM EquationOfStateFactory that has \n"
    << " slipped through the existing bullet proofing. Please check and correct." <<
        endl;
    throw ParameterNotFound(desc.str(), __FILE__, __LINE__);
  }

  // Universal Labels
```

```
  bElBarLabel = VarLabel::create("p.bElBar", ParticleVariable<Matrix3>::
        getTypeDescription());
  bElBarLabel_preReloc = VarLabel::create("p.bElBar+", ParticleVariable<Matrix3>::
        getTypeDescription());
}

CompOgdenStorakers::CompOgdenStorakers(const CompOgdenStorakers* cm) :
      ConstitutiveModel(cm)
{
  d_initialData.PR = cm->d_initialData.PR;
  d_initialData.alpha = cm->d_initialData.alpha;
  d_initialData.mu = cm->d_initialData.mu;
  d_useModifiedEOS = cm->d_useModifiedEOS;

  // Initial stress
  d_useInitialStress = cm->d_useInitialStress;
  if (d_useInitialStress) {
    d_init_pressure = cm->d_init_pressure;
  }

  // Equation of state factory for pressure (default is DefaultHyperEOS)
  d_eos = MPMEquationOfStateFactory::createCopy(cm->d_eos);

  d_eos->setBulkModulus(CalculateBulkModulus());

  if(!d_eos){
    ostringstream desc;
    desc << "An error occured in the MPM EquationOfStateFactory that has \n"
    << " slipped through the existing bullet proofing. Please check and correct." <<
          endl;
    throw ParameterNotFound(desc.str(), __FILE__, __LINE__);
  }

  // Universal Labels
  bElBarLabel = VarLabel::create("p.bElBar", ParticleVariable<Matrix3>::
        getTypeDescription());
  bElBarLabel_preReloc = VarLabel::create("p.bElBar+", ParticleVariable<Matrix3>::
        getTypeDescription());
}

CompOgdenStorakers::~CompOgdenStorakers()
{
  // Universal Deletes
  VarLabel::destroy(bElBarLabel);
  VarLabel::destroy(bElBarLabel_preReloc);
}

CompOgdenStorakers* CompOgdenStorakers::clone()
{
  return scinew CompOgdenStorakers(*this);
}

void CompOgdenStorakers::outputProblemSpec(ProblemSpecP& ps, bool
      output_cm_tag)
{
  ProblemSpecP cm_ps = ps;
  if (output_cm_tag) {
    cm_ps = ps->appendChild("constitutive_model");
    cm_ps->setAttribute("type","comp_ogden_storakers");
  }
```

```
  cm_ps->appendElement("PR", d_initialData.PR);
  cm_ps->appendElement("alpha_1", d_initialData.alpha);
  cm_ps->appendElement("mu_1", d_initialData.mu);
  cm_ps->appendElement("useModifiedEOS",d_useModifiedEOS);
}

void CompOgdenStorakers::initializeCMData(const Patch* patch, const
      MPMMaterial* matl, DataWarehouse* new_dw)
{
  // Put stuff in here to initialize each particle's
  // constitutive model parameters and deformationMeasure
  Matrix3 Identity;
  Identity.Identity();
  Matrix3 zero(0.0);

  ParticleSubset* pset = new_dw->getParticleSubset(matl->getDWIndex(), patch);

  ParticleSubset::iterator iterUniv = pset->begin();

  // Initialize the variables shared by all constitutive models
  // This method is defined in the ConstitutiveModel base class.
  initSharedDataForExplicit(patch, matl, new_dw);

  // calcualte the initial timestep, after this the timestep will be calculated by
        computeStressTensor
  computeStableTimestep(patch, matl, new_dw);

  // Universal
  ParticleVariable<Matrix3> deformationGradient, pstress, bElBar;

  new_dw->allocateAndPut(bElBar, bElBarLabel, pset);

  for(;iterUniv != pset->end(); iterUniv++){
    bElBar[*iterUniv] = Identity;
  }
}

void CompOgdenStorakers::addParticleState(std::vector<const VarLabel*>& from,
      std::vector<const VarLabel*>& to)
{
  // Keeps track of the particles and the related variables as particles move from
        patch to patch (each CM adds its own state variables, e.g. failure, damage,
        plasticity, etc.)
  // Universal
  from.push_back(bElBarLabel);
  to.push_back(bElBarLabel_preReloc);
}

void CompOgdenStorakers::addInitialComputesAndRequires(Task* task, const
      MPMMaterial* matl, const PatchSet*) const
{
  const MaterialSubset* matlset = matl->thisMaterial();
  // Universal
  task->computes(bElBarLabel, matlset);
}

// Tells the scheduler what data needs to be available at the time
      computeStressTensor(...) is called
void CompOgdenStorakers::addComputesAndRequires(Task* task, const
      MPMMaterial* matl, const PatchSet* patches) const
```

```
{
  const MaterialSubset* matlset = matl->thisMaterial();

  // Add the computes and requires that are common to all explicit constitutive
       models.
  // This method is defined in the ConstitutiveModel base class.
  addSharedCRForExplicit(task, matlset, patches);

  // Other constitutive model and input dependent computes and requires
  Ghost::GhostType gnone = Ghost::None;

    task->requires(Task::OldDW, lb->pParticleIDLabel, matlset, gnone);

  if(flag->d_with_color) {
    task->requires(Task::OldDW, lb->pColorLabel, Ghost::None);
  }

  // Universal
  task->requires(Task::OldDW, bElBarLabel, matlset, gnone);
  task->computes(bElBarLabel_preReloc,  matlset);
}

void CompOgdenStorakers::addComputesAndRequires(Task* task, const
      MPMMaterial* matl, const PatchSet* patches, const bool recurse, const bool
      SchedParent) const
{
  const MaterialSubset* matlset = matl->thisMaterial();

  Ghost::GhostType gnone = Ghost::None;

  if(SchedParent){
    task->requires(Task::ParentOldDW, bElBarLabel, matlset, gnone);
  }
  else{
    task->requires(Task::OldDW, bElBarLabel, matlset, gnone);
  }
}

// Carry forward CM data for RigidMPM
void CompOgdenStorakers::carryForward(const PatchSubset* patches, const
      MPMMaterial* matl, DataWarehouse* old_dw, DataWarehouse* new_dw)
{
#if 0
  for(int p=0; p<patches->size(); p++){
    const Patch* patch = patches->get(p);
    int dwi = matl->getDWIndex();
    ParticleSubset* pset = old_dw->getParticleSubset(dwi, patch);

    // Carry forward the data common to all constitutive models when using
         RigidMPM.
    // This method is defined in the ConstitutiveModel base class.
    carryForwardSharedData(pset, old_dw, new_dw, matl);

    // Carry forward the data local to this constitutive model
    new_dw->put(delt_vartype(1.e10), lb->delTLabel, patch->getLevel());
    if (flag->d_reductionVars->accStrainEnergy || flag->d_reductionVars->
         strainEnergy) {
      new_dw->put(sum_vartype(0.0), lb->StrainEnergyLabel);
    }
  }
```

```
#endif
  for(int p=0;p<patches->size();p++){
    const Patch* patch = patches->get(p);
    int dwi = matl->getDWIndex();
    ParticleSubset* pset = old_dw->getParticleSubset(dwi, patch);

    // Carry forward the data common to all constitutive models
    // when using RigidMPM.
    // This method is defined in the ConstitutiveModel base class.
    carryForwardSharedData(pset, old_dw, new_dw, matl);

    // Carry forward the data local to this constitutive model
    // Universal
    ParticleVariable<Matrix3> bElBar_new;
    constParticleVariable<Matrix3> bElBar;
    old_dw->get(bElBar, bElBarLabel, pset);
    new_dw->allocateAndPut(bElBar_new, bElBarLabel_preReloc, pset);
    for(ParticleSubset::iterator iter = pset->begin();
      iter != pset->end(); iter++){
      particleIndex idx = *iter;
      bElBar_new[idx] = bElBar[idx];
    }
    new_dw->put(delt_vartype(1.e10), lb->delTLabel, patch->getLevel());
    if (flag->d_reductionVars->accStrainEnergy ||
      flag->d_reductionVars->strainEnergy) {
      new_dw->put(sum_vartype(0.), lb->StrainEnergyLabel);
    }
  } // End Particle Loop

}

// Calculate the initial timestep, after this the timestep (dT) will be calculated by
      computeStressTensor(...)
// The size of the timestep depends on cell spacing, velocity of the particle, and
      the material wavespeed (c_dil) at each particle
// A reduction over all dTs from every patch is performed, the smallest dT is used
void CompOgdenStorakers::computeStableTimestep(const Patch* patch, const
      MPMMaterial* matl, DataWarehouse* new_dw)
{
  Vector dx = patch->dCell();
  int dwi = matl->getDWIndex();

  // Retrieve the array of constitutive parameters
  ParticleSubset* pset = new_dw->getParticleSubset(dwi, patch);
  constParticleVariable<double> pmass, pvolume;
  constParticleVariable<Vector> pvelocity;

  new_dw->get(pmass, lb->pMassLabel, pset);
  new_dw->get(pvolume, lb->pVolumeLabel, pset);
  new_dw->get(pvelocity, lb->pVelocityLabel, pset);

  double c_dil = 0.0; // local speed of sound
  Matrix3 F; // Deformation gradient
  Matrix3 B; // Left Cauchy Green deformation tensor
  Matrix3 Identity;
  Identity.Identity(); // 3x3 Identity matrix
  Vector WaveSpeed(1.e-12, 1.e-12, 1.e-12);
  Vector eVals(0, 0, 0); // Eigenvalues
  Matrix3 eVecs; // Eigenvectors
  double a = d_initialData.alpha; // Ogden material constant
```

```
  double mu = d_initialData.mu; // Ogden material constant
  double l1; // first principal stretch

  // Assuming a stress free reference configuration F = Identity
  F = Identity;

  // Compute the left Cauchy-Green deformation tensor
    B = F*F.Transpose();

  // The principal stretches can be calculated from the square roots of the
       Eigenvlaues of B
  B.eigen(eVals, eVecs);

  // Only the first principal stretch (the biggest) is needed
  l1 = sqrt(eVals.x());

  // Compute wave speed at each particle, store the maximum
  for(ParticleSubset::iterator iter = pset->begin(); iter != pset->end(); iter++){
    particleIndex idx = *iter;

    // Compute c_dil, the local speed of sound
    c_dil = sqrt(
      mu*(a * pow(l1, a - 1.0) + 0.5 * a * pow(l1, -0.5 * a - 1.0))/(pmass[idx]/
           pvolume[idx])
    );

    // Compute wave speed + particle velocity at each particle, then store the
         maximum
    WaveSpeed=Vector(Max(c_dil+fabs(pvelocity[idx].x()), WaveSpeed.x()),
                     Max(c_dil+fabs(pvelocity[idx].y()), WaveSpeed.y()),
                   Max(c_dil+fabs(pvelocity[idx].z()), WaveSpeed.z()));
  }

  WaveSpeed = dx/WaveSpeed;
  double delT_new = WaveSpeed.minComponent();
  new_dw->put(delt_vartype(delT_new), lb->delTLabel, patch->getLevel());
}

// Computes the Cauchy stress for each particle for the given material
// Called once per timestep (for each material)
void CompOgdenStorakers::computeStressTensor(const PatchSubset* patches, const
      MPMMaterial* matl, DataWarehouse* old_dw, DataWarehouse* new_dw)
{
  // Loop over each patch
  for(int patch_idx = 0; patch_idx < patches->size(); patch_idx++){
    const Patch* patch = patches->get(patch_idx);

    Matrix3 F; // Deformation gradient
    Matrix3 B; // Left Cauchy Green deformation tensor
    Matrix3 Identity; Identity.Identity(); // 3x3 Identity matrix
    Vector WaveSpeed(1.e-12, 1.e-12, 1.e-12);
    Vector eVals(0, 0, 0); // Eigenvalues
    Matrix3 eVecs; // Eigenvectors
    Matrix3 n1, n2, n3; // Outer product of Eigenvectors
    double c_dil = 0.0; // Local speed of sound
    double se = 0.0; // Accumulated strain energy for all particles
    double J; // Jacobian of deformation gradient
    double l1, l2, l3; // The principl stretches, lambda_1, lambda_2, lambda_3
    double dW_dl1, dW_dl2, dW_dl3; // The strain energy function partially
         differentiated by lambda_1, lambda_2, lambda_3
```

```
double scalar1, scalar2, scalar3;
double a = 0.0; // Ogden material constant
double mu = 0.0; // Ogden material constant
double v; // Poisson's ratio (material constant)
double rho_current; // Current density
double rho_orig; // Original density

  Matrix3 bElBarTrial(0.0), pDefGradInc(0.0), fBar(0.0); ///
double Jinc = 0.0;

ParticleInterpolator* interpolator = flag->d_interpolator->clone(patch);
vector<IntVector> ni(interpolator->size());
vector<Vector> d_S(interpolator->size());
vector<double> S(interpolator->size());

Vector dx = patch->dCell();

// DataWarehouse index
int dwi = matl->getDWIndex();

// Create array for the particle position
ParticleSubset* pset = old_dw->getParticleSubset(dwi, patch);
constParticleVariable<Point> px;
constParticleVariable<Matrix3> deformationGradient_new;
constParticleVariable<Matrix3> deformationGradient;
ParticleVariable<Matrix3> pstress;
constParticleVariable<double> pmass;
constParticleVariable<double> pvolume_new;
constParticleVariable<Vector> pvelocity;
constParticleVariable<Matrix3> velGrad;
constParticleVariable<Matrix3> psize;
ParticleVariable<double> pdTdt,p_q;

constParticleVariable<Matrix3> pDefGrad, bElBar; ///
ParticleVariable<Matrix3> pStress, bElBar_new; ///

delt_vartype delT;
old_dw->get(delT,      lb->delTLabel, getLevel(patches));
old_dw->get(pvelocity,   lb->pVelocityLabel, pset);
old_dw->get(deformationGradient,   lb->pDeformationMeasureLabel, pset);
old_dw->get(bElBar,     bElBarLabel, pset); ///

new_dw->allocateAndPut(pstress, lb->pStressLabel_preReloc, pset);
new_dw->allocateAndPut(pdTdt, lb->pdTdtLabel, pset);
new_dw->allocateAndPut(p_q, lb->p_qLabel_preReloc, pset);

new_dw->get(pvolume_new,    lb->pVolumeLabel_preReloc, pset);
new_dw->get(velGrad,    lb->pVelGradLabel_preReloc, pset);
new_dw->get(deformationGradient_new, lb->
      pDeformationMeasureLabel_preReloc, pset);

new_dw->allocateAndPut(bElBar_new,  bElBarLabel_preReloc, pset); ///

// Get initial data
a = d_initialData.alpha;
mu = d_initialData.mu;
v = d_initialData.PR;
rho_orig = matl->getInitialDensity();

for(ParticleSubset::iterator iter = pset->begin();iter!=pset->end();iter++){
```

```
particleIndex idx = *iter;

// Assign zero internal heating by default - modify if necessary.
pdTdt[idx] = 0.0;

// Get deformation gradient
F = deformationGradient_new[idx];

///
// Calculate bElBar (needed for interaction with RigidMPM)
pDefGradInc = deformationGradient_new[idx]*deformationGradient[idx].
      Inverse();
Jinc = pDefGradInc.Determinant();

// Get the volume preserving part of the deformation gradient increment
fBar = pDefGradInc/cbrt(Jinc);

// Compute the elastic part of the volume preserving
// part of the left Cauchy-Green deformation tensor
bElBar_new[idx] = fBar*bElBar[idx]*fBar.Transpose();

// Compute the left Cauchy-Green deformation tensor
     B = F*F.Transpose();

// get the original volumetric part of the deformation
J = F.Determinant();

// Error checking
if (J < 0.0) {
  throw InvalidValue("Negative Jacobian of deformation gradient", __FILE__,
        __LINE__);
}

// The Ogden model is typically expressed in terms of the principal
      stretches rather than the invariants
// The principal stretches can be calculated from the square roots of the
      Eigenvlaues of B
B.eigen(eVals, eVecs);

// Calcualte the principal stretches
l1 = sqrt(eVals.x());
l2 = sqrt(eVals.y());
l3 = sqrt(eVals.z());

// Put the Eigenvectors in matrix form for convenience
n1.set(0,0,eVecs(0,0));
n1.set(1,0,eVecs(1,0));
n1.set(2,0,eVecs(2,0));
n1.set(0,1,0);
n1.set(1,1,0);
n1.set(2,1,0);
n1.set(0,2,0);
n1.set(1,2,0);
n1.set(2,2,0);
n2.set(0,0,eVecs(0,1));
n2.set(1,0,eVecs(1,1));
n2.set(2,0,eVecs(2,1));
n2.set(0,1,0);
n2.set(1,1,0);
n2.set(2,1,0);
```

```
    n2.set(0,2,0);
    n2.set(1,2,0);
    n2.set(2,2,0);
    n3.set(0,0,eVecs(0,2));
    n3.set(1,0,eVecs(1,2));
    n3.set(2,0,eVecs(2,2));
    n3.set(0,1,0);
    n3.set(1,1,0);
    n3.set(2,1,0);
    n3.set(0,2,0);
    n3.set(1,2,0);
    n3.set(2,2,0);

    // Calculate the outer product of the Eigenvectors, results in a 3x3 matrix
    n1 = n1*n1.Transpose();
    n2 = n2*n2.Transpose();
    n3 = n3*n3.Transpose();

    // Calculate scalars based on Ogden strain energy density function
    dW_dl1 = 2.0 * mu * (pow(l1, a) * a / l1 - pow(J, -a * (double) v / (double)
        (1.0 - 2.0 * v)) * a / l1) * pow(a, -2.0);
    dW_dl2 = 2.0 * mu * (pow(l2, a) * a / l2 - pow(J, -a * (double) v / (double)
        (1.0 - 2.0 * v)) * a / l2) * pow(a, -2.0);
    dW_dl3 = 2.0 * mu * (pow(l3, a) * a / l3 - pow(J, -a * (double) v / (double)
        (1.0 - 2.0 * v)) * a / l3) * pow(a, -2.0);
    scalar1 = (l1 / J) * dW_dl1;
    scalar2 = (l2 / J) * dW_dl2;
    scalar3 = (l3 / J) * dW_dl3;

    // Calculate Cauchy stress for particle
    pstress[idx] = scalar1*n1 + scalar2*n2 + scalar3*n3;

    // Calculate total strain energy over all particles
    se += 2.0 * mu * (pow(l1, a) + pow(l2, a) + pow(l3, a) - 3.0 + (pow(J, -a * (
        double) v / (double) (1.0 - 2.0 * v)) - 1.0) / (double) v * (double) (1.0 -
         2.0 * v)) * pow(a, -2.0);

    // Get the current density
    rho_current = rho_orig/J;

    // Compute c_dil, the local speed of sound, this came from Ogden's book (
        Non-Linear Elastic Deformations)
    c_dil = sqrt(
      mu*(a * pow(l1, a - 1.0) + 0.5 * a * pow(l1, - 0.5 * a - 1.0))/rho_current
    );

    // Compute wave speed + particle velocity at each particle, then store the
        maximum
    WaveSpeed=Vector(Max(c_dil+fabs(pvelocity[idx].x()), WaveSpeed.x()),
                     Max(c_dil+fabs(pvelocity[idx].y()), WaveSpeed.y()),
                    Max(c_dil+fabs(pvelocity[idx].z()), WaveSpeed.z()));

    // Compute artificial viscosity term
    if (flag->d_artificial_viscosity) {
      throw InvalidValue("Artificial viscosity has not been implemented for this
          constitutive model", __FILE__, __LINE__);
      // To be added later
    }
    else{
      p_q[idx] = 0.0;
```

```
      }
    } // end loop over particles

    WaveSpeed = dx/WaveSpeed;
    double delT_new = WaveSpeed.minComponent();

    new_dw->put(delt_vartype(delT_new), lb->delTLabel, patch->getLevel());

    if (flag->d_reductionVars->accStrainEnergy || flag->d_reductionVars->
        strainEnergy) {
      new_dw->put(sum_vartype(se), lb->StrainEnergyLabel);
    }
    delete interpolator;
  }
}

double CompOgdenStorakers::computeRhoMicroCM(double pressure, const
    double p_ref, const MPMMaterial* matl, double temperature, double
    rho_guess)
{
  double p_gauge = pressure - p_ref;
  double rho_orig = matl->getInitialDensity();
  double rho_cur = -1.0;
  bool error = false;

  if (d_useModifiedEOS && p_gauge < 0.0) { // Modified EOS

    double K = CalculateBulkModulus();
    double A = p_ref;
    double n = p_ref/K;
    rho_cur = rho_orig*pow(pressure/A,n);

  } else { // Standard EOS

    try {
      rho_cur = d_eos->computeDensity(rho_orig, -p_gauge);
    } catch (ConvergenceFailure& e) {
      cout << e.message() << endl;
      error = true;
    }
    if (error || rho_cur < 0.0 || isnan(rho_cur)) {
      ostringstream desc;
      desc << "rho_cur = " << rho_cur << " pressure = " << -p_gauge
           << " p_ref = " << p_ref << " 1/sp_vol_CC = " << rho_guess << endl;
      throw InvalidValue(desc.str(), __FILE__, __LINE__);
    }
  }

    return rho_cur;
}

void CompOgdenStorakers::computePressEOSCM(const double rho_cur,double&
    pressure, const double p_ref, double& dp_drho, double& cSquared, const
    MPMMaterial* matl, double temperature)
{
  double rho_orig = matl->getInitialDensity();

  if (d_useModifiedEOS && rho_cur < rho_orig) { // Modified EOS

    double K = CalculateBulkModulus();
```

```
    double A = p_ref;
    double n = K/p_ref;
    double rho_rat_to_the_n = pow(rho_cur/rho_orig,n);
    pressure = A*rho_rat_to_the_n;
    dp_drho = (K/rho_cur)*rho_rat_to_the_n;
    cSquared = dp_drho; // Speed of sound squared

  } else { // Standard EOS

    double p = 0.0;
    d_eos->computePressure(rho_orig, rho_cur, p, dp_drho, cSquared);
    pressure = -p + p_ref;
    dp_drho = -dp_drho;
  }

}

double CompOgdenStorakers::getCompressibility()
{
  return 1.0/CalculateBulkModulus();
}

double CompOgdenStorakers::CalculateBulkModulus()
{
  double K; // Bulk modulus
  //double E; // Elastic modulus

  // Get initial data
  //double a = d_initialData.alpha;
  double mu = d_initialData.mu;
  double v = d_initialData.PR;

  // Calculate elastic modulus
  //E = (3.0/2.0)*mu*a;

  // Calculate bulk modulus
  //K = E/(3.0*(1.0 - 2.0*v));
  K = 2.0 * mu * ((double) (v / (1.0 - 2.0 * v)) + 1.0 / 3.0);

  return K;
}
```

### A.2.3 *Linear Model (Impactor/Backplate)*

Listing 5: CompLinear.h

```
#ifndef __COMP_OGDEN_STORAKERS_CONSTITUTIVE_MODEL_H__
  #define __COMP_OGDEN_STORAKERS_CONSTITUTIVE_MODEL_H__

#include <CCA/Components/MPM/ConstitutiveModel/ConstitutiveModel.h>
#include "../PlasticityModels/MPMEquationOfState.h"
#include <Core/Math/Matrix3.h>
#include <Core/ProblemSpec/ProblemSpecP.h>
#include <CCA/Ports/DataWarehouseP.h>

#include <cmath>

namespace Uintah {
```

```
class MPMLabel;
class MPMFlags;

class CompOgdenStorakers : public ConstitutiveModel {

  private:

    bool d_useModifiedEOS; // use modified equation of state

  protected:

    bool d_useInitialStress; // Initial stress state
    double d_init_pressure; // Initial pressure

    MPMEquationOfState* d_eos; // Equation of State factory

  public:

    struct CMData {
      double PR; //Poisson's Ratio
      double alpha; // material constant
      double mu; // material constant
    };

    const VarLabel* bElBarLabel;
    const VarLabel* bElBarLabel_preReloc;

  private:

    CMData d_initialData;

    // Prevent copying of this class
    CompOgdenStorakers& operator=(const CompOgdenStorakers &cm);

    // Calculate the bulk modulus based on the Ogden material constants
    double CalculateBulkModulus();

  public:

    // Constructor
    CompOgdenStorakers(ProblemSpecP& ps, MPMFlags* flag);

    // Copy constructor
    CompOgdenStorakers(const CompOgdenStorakers* cm);

    // Destructor
    virtual ~CompOgdenStorakers();

    // Clone
    CompOgdenStorakers* clone();

    virtual void outputProblemSpec(ProblemSpecP& ps,bool output_cm_tag =
          true);

    // initialize each particle's constitutive model data
    virtual void initializeCMData(const Patch* patch, const MPMMaterial* matl,
          DataWarehouse* new_dw);
```

```
      // Keeps track of the particles and the related variables as particles move
            from patch to patch
      virtual void addParticleState(std::vector<const VarLabel*>& from, std::vector
            <const VarLabel*>& to);

      // Tells the scheduler what data needs to be available at the time
            computeStressTensor(...) is called
      virtual void addComputesAndRequires(Task* task, const MPMMaterial* matl
            , const PatchSet* patches) const;

      virtual void addComputesAndRequires(Task* task, const MPMMaterial* matl
            , const PatchSet* patches, const bool recurse, const bool SchedParent)
            const;

      virtual void addInitialComputesAndRequires(Task* task, const MPMMaterial
            * matl, const PatchSet*) const;

      // Carry forward constitutive model data for RigidMPM
      virtual void carryForward(const PatchSubset* patches, const MPMMaterial*
            matl, DataWarehouse* old_dw, DataWarehouse* new_dw);

      // Calculate the initial timestep, after this the timestep (dT) will be
            calculated by computeStressTensor(...)
      virtual void computeStableTimestep(const Patch* patch, const MPMMaterial*
             matl, DataWarehouse* new_dw);

      // Compute cauchy stress for each particle in the patch
      virtual void computeStressTensor(const PatchSubset* patches, const
            MPMMaterial* matl, DataWarehouse* old_dw, DataWarehouse*
            new_dw);

      virtual double computeRhoMicroCM(double pressure, const double p_ref,
            const MPMMaterial* matl, double temperature, double rho_guess);

      virtual void computePressEOSCM(double rho_m, double& press_eos, double
             p_ref, double& dp_drho, double& ss_new, const MPMMaterial* matl,
            double temperature);

      virtual double getCompressibility();

  };
} // End namespace Uintah

#endif // __COMP_OGDEN_STORAKERS_CONSTITUTIVE_MODEL_H__
```

Listing 6: CompLinear.cc

```
/*
 * The MIT License
 *
 * Copyright (c) 1997–2012 The University of Utah
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 * of this software and associated documentation files (the "Software"), to
 * deal in the Software without restriction, including without limitation the
 * rights to use, copy, modify, merge, publish, distribute, sublicense, and/or
 * sell copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included in
```

```
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY
       KIND, EXPRESS OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
       MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO
       EVENT SHALL THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM,
       DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR
       OTHERWISE, ARISING
 * FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE
       OR OTHER DEALINGS
 * IN THE SOFTWARE.
 */

#include <CCA/Components/MPM/ConstitutiveModel/UoL/CompLinear.h>
#include <CCA/Components/MPM/ConstitutiveModel/MPMMaterial.h>
#include <CCA/Components/MPM/ConstitutiveModel/PlasticityModels/
      MPMEquationOfStateFactory.h>
#include <CCA/Components/MPM/MPMFlags.h>
#include <Core/Math/Matrix3.h>
#include <CCA/Ports/DataWarehouse.h>
#include <Core/Grid/Variables/VarLabel.h>
#include <Core/Grid/Variables/ParticleVariable.h>
#include <Core/Grid/Variables/NCVariable.h>
#include <Core/Grid/Patch.h>
#include <Core/Grid/Variables/VarTypes.h>
#include <Core/Labels/MPMLabel.h>
#include <Core/Math/FastMatrix.h>
#include <Core/Exceptions/ParameterNotFound.h>
#include <Core/Exceptions/InvalidValue.h>
#include <Core/Exceptions/ConvergenceFailure.h>
#include <Core/Malloc/Allocator.h>
#include <cmath>
#include <iostream>
#include <stdio.h>
#include <iomanip>

using namespace Uintah;
using namespace std;

CompLinear::CompLinear(ProblemSpecP& ps, MPMFlags* Mflag) :
      ConstitutiveModel(Mflag)
{
  // required - material parameters for the constitutive model
  ps->require("E", d_initialData.E);
  ps->require("v", d_initialData.v);

  // optional - use modified equation of state (off by default)
  d_useModifiedEOS = false;
  ps->get("useModifiedEOS",d_useModifiedEOS); // no negative pressure for
       solids

  // Initial stress
  // Fix: Need to make it more general. Add gravity turn-on option and
  // read from file option etc.
  ps->getWithDefault("useInitialStress", d_useInitialStress, false);
  d_init_pressure = 0.0;
```

```
  if (d_useInitialStress) {
    ps->getWithDefault("initial_pressure", d_init_pressure, 0.0);
  }

  // Equation of state factory for pressure (default is DefaultHyperEOS)
  d_eos = MPMEquationOfStateFactory::create(ps);

  d_eos->setBulkModulus(CalculateBulkModulus());

  if(!d_eos){
    ostringstream desc;
    desc << "An error occured in the MPM EquationOfStateFactory that has \n"
    << " slipped through the existing bullet proofing. Please check and correct." <<
        endl;
    throw ParameterNotFound(desc.str(), __FILE__, __LINE__);
  }
}

CompLinear::CompLinear(const CompLinear* cm) : ConstitutiveModel(cm)
{
  d_initialData.E = cm->d_initialData.E;
  d_initialData.v = cm->d_initialData.v;
  d_useModifiedEOS = cm->d_useModifiedEOS;

  // Initial stress
  d_useInitialStress = cm->d_useInitialStress;
  if (d_useInitialStress) {
    d_init_pressure = cm->d_init_pressure;
  }

  // Equation of state factory for pressure (default is DefaultHyperEOS)
  d_eos = MPMEquationOfStateFactory::createCopy(cm->d_eos);

  d_eos->setBulkModulus(CalculateBulkModulus());

  if(!d_eos){
    ostringstream desc;
    desc << "An error occured in the MPM EquationOfStateFactory that has \n"
    << " slipped through the existing bullet proofing. Please check and correct." <<
        endl;
    throw ParameterNotFound(desc.str(), __FILE__, __LINE__);
  }
}

CompLinear::~CompLinear()
{
}

CompLinear* CompLinear::clone()
{
  return scinew CompLinear(*this);
}

void CompLinear::outputProblemSpec(ProblemSpecP& ps, bool output_cm_tag)
{
  ProblemSpecP cm_ps = ps;
  if (output_cm_tag) {
    cm_ps = ps->appendChild("constitutive_model");
    cm_ps->setAttribute("type","comp_linear");
  }
```

```
  cm_ps->appendElement("E", d_initialData.E);
  cm_ps->appendElement("v", d_initialData.v);
  cm_ps->appendElement("useModifiedEOS",d_useModifiedEOS);
}

void CompLinear::initializeCMData(const Patch* patch, const MPMMaterial* matl,
      DataWarehouse* new_dw)
{
  // Initialize the variables shared by all constitutive models
  // This method is defined in the ConstitutiveModel base class.
  initSharedDataForExplicit(patch, matl, new_dw);

  // calcualte the initial timestep, after this the timestep will be calculated by
       computeStressTensor
  computeStableTimestep(patch, matl, new_dw);
}

void CompLinear::addParticleState(std::vector<const VarLabel*>& from, std::vector
      <const VarLabel*>& to)
{
  // Keeps track of the particles and the related variables as particles move from
       patch to patch (each CM adds its own state variables, e.g. failure, damage,
       plasticity, etc.)
  // Nothing to do for this CM
}

// Tells the scheduler what data needs to be available at the time
      computeStressTensor(...) is called
void CompLinear::addComputesAndRequires(Task* task, const MPMMaterial*
      matl, const PatchSet* patches) const
{
  const MaterialSubset* matlset = matl->thisMaterial();

  // Add the computes and requires that are common to all explicit constitutive
       models.
  // This method is defined in the ConstitutiveModel base class.
  addSharedCRForExplicit(task, matlset, patches);
}

void CompLinear::addComputesAndRequires(Task*, const MPMMaterial*, const
      PatchSet*, const bool) const
{
  // Nothing to do for this CM
}

// Carry forward CM data for RigidMPM - RigidMPM contains a very reduced
      level of functionality, and is used solely in conjunction with the MPMArches
      component.
void CompLinear::carryForward(const PatchSubset* patches, const MPMMaterial*
      matl, DataWarehouse* old_dw, DataWarehouse* new_dw)
{
  for(int p=0; p<patches->size(); p++){
    const Patch* patch = patches->get(p);
    int dwi = matl->getDWIndex();
    ParticleSubset* pset = old_dw->getParticleSubset(dwi, patch);

    // Carry forward the data common to all constitutive models when using
         RigidMPM.
    // This method is defined in the ConstitutiveModel base class.
    carryForwardSharedData(pset, old_dw, new_dw, matl);
```

```
    // Carry forward the data local to this constitutive model
    new_dw->put(delt_vartype(1.e10), lb->delTLabel, patch->getLevel());
    if (flag->d_reductionVars->accStrainEnergy || flag->d_reductionVars->
        strainEnergy) {
      new_dw->put(sum_vartype(0.0), lb->StrainEnergyLabel);
    }
  }
}

// Calculate the initial timestep, after this the timestep (dT) will be calculated by
    computeStressTensor(...)
// The size of the timestep depends on cell spacing, velocity of the particle, and
    the material wavespeed (c_dil) at each particle
// A reduction over all dTs from every patch is performed, the smallest dT is used
void CompLinear::computeStableTimestep(const Patch* patch, const MPMMaterial*
    matl, DataWarehouse* new_dw)
{
  Vector dx = patch->dCell();
  int dwi = matl->getDWIndex();

  // Retrieve the array of constitutive parameters
  ParticleSubset* pset = new_dw->getParticleSubset(dwi, patch);
  constParticleVariable<double> pmass;
  constParticleVariable<Vector> pvelocity;

  new_dw->get(pmass, lb->pMassLabel, pset);
  new_dw->get(pvelocity, lb->pVelocityLabel, pset);

  Matrix3 F; // Deformation gradient
  Matrix3 Identity;
  Identity.Identity(); // 3x3 Identity matrix
  Vector WaveSpeed(1.e-12, 1.e-12, 1.e-12);
  double c_dil = 0.0; // Local speed of sound
  double J; // Jacobian of deformation gradient
  double E; // Elastic modulus
  double v; // Poisson's ratio
  double k; // Bulk modulus
  double rho_current; // Current density
  double rho_orig; // Original density

  // Get initial data
  E = d_initialData.E;
  v = d_initialData.v;
  rho_orig = matl->getInitialDensity();

  // Assuming a stress free reference configuration F = Identity
  F = Identity;
  J = F.Determinant();

  // Compute wave speed at each particle, store the maximum
  for(ParticleSubset::iterator iter = pset->begin(); iter != pset->end(); iter++){
    particleIndex idx = *iter;

    //Compute c_dil, the local speed of sound
    k = E / (3.0 * (1.0 - 2.0 * v));
    rho_current = rho_orig/J;
    c_dil = sqrt(k/rho_current); // Newton Laplace equation
```

```
    // Compute wave speed + particle velocity at each particle, then store the
        maximum
    WaveSpeed=Vector(Max(c_dil+fabs(pvelocity[idx].x()), WaveSpeed.x()),
                     Max(c_dil+fabs(pvelocity[idx].y()), WaveSpeed.y()),
                    Max(c_dil+fabs(pvelocity[idx].z()), WaveSpeed.z()));
  }

  WaveSpeed = dx/WaveSpeed;
  double delT_new = WaveSpeed.minComponent();
  new_dw->put(delt_vartype(delT_new), lb->delTLabel, patch->getLevel());
}

// Computes the Cauchy stress for each particle for the given material
// Called once per timestep (for each material)
void CompLinear::computeStressTensor(const PatchSubset* patches, const
     MPMMaterial* matl, DataWarehouse* old_dw, DataWarehouse* new_dw)
{
  // Loop over each patch
  for(int patch_idx = 0; patch_idx < patches->size(); patch_idx++){
    const Patch* patch = patches->get(patch_idx);

    Matrix3 F; // Deformation gradient
    Matrix3 Identity; Identity.Identity(); // 3x3 Identity matrix
    Vector WaveSpeed(1.e-12, 1.e-12, 1.e-12);
    Matrix3 LST; // Lagrange strain tensor
    double c_dil = 0.0; // Local speed of sound
    double se= 0.0; // Accumulated strain energy for all particles
    double J; // Jacobian of deformation gradient
    double E; // Elastic modulus
    double v; // Poisson's ratio
    double k; // Bulk modulus
    double rho_current; // Current density
    double rho_orig; // Original density

    ParticleInterpolator* interpolator = flag->d_interpolator->clone(patch);
    vector<IntVector> ni(interpolator->size());
    vector<Vector> d_S(interpolator->size());
    vector<double> S(interpolator->size());

    Vector dx = patch->dCell();

    // DataWarehouse index
    int dwi = matl->getDWIndex();

    // Create array for the particle position
    ParticleSubset* pset = old_dw->getParticleSubset(dwi, patch);
    constParticleVariable<Point> px;
    constParticleVariable<Matrix3> deformationGradient_new;
    constParticleVariable<Matrix3> deformationGradient;
    ParticleVariable<Matrix3> pstress;
    constParticleVariable<double> pmass;
    constParticleVariable<double> pvolume_new;
    constParticleVariable<Vector> pvelocity;
    constParticleVariable<Matrix3> velGrad;
    constParticleVariable<Matrix3> psize;
    ParticleVariable<double> pdTdt,p_q;

    delt_vartype delT;
    old_dw->get(delT,      lb->delTLabel, getLevel(patches));
    old_dw->get(pvelocity,   lb->pVelocityLabel, pset);
```

```
old_dw->get(deformationGradient,   lb->pDeformationMeasureLabel, pset);

new_dw->allocateAndPut(pstress, lb->pStressLabel_preReloc, pset);
new_dw->allocateAndPut(pdTdt, lb->pdTdtLabel, pset);
new_dw->allocateAndPut(p_q, lb->p_qLabel_preReloc, pset);
new_dw->get(pvolume_new,    lb->pVolumeLabel_preReloc, pset);
new_dw->get(velGrad,    lb->pVelGradLabel_preReloc, pset);
new_dw->get(deformationGradient_new, lb->
      pDeformationMeasureLabel_preReloc, pset);

// Get initial data
E = d_initialData.E;
v = d_initialData.v;
rho_orig = matl->getInitialDensity();

for(ParticleSubset::iterator iter = pset->begin();iter!=pset->end();iter++){
  particleIndex idx = *iter;

  // Assign zero internal heating by default - modify if necessary.
  pdTdt[idx] = 0.0;

  // Get deformation gradient
  F = deformationGradient_new[idx];

  // get the original volumetric part of the deformation
  J = F.Determinant();

  // Error checking
  if (J < 0.0) {
    throw InvalidValue("Negative Jacobian of deformation gradient", __FILE__,
          __LINE__);
  }

  // Calculate the Lagrange strain tensor
  //LST = F.Transpose() * F - Identity;
  //LST = LST / 2.0;

  // Calculate Cauchy stress for particle
  //pstress[idx] = LST + (v / (1.0 - 2.0 * v)) * (LST(0,0) + LST(1,1) + LST(2,2))
       * Identity;
  //pstress[idx] = pstress[idx] * (E / 1.0 + v);

  //nope
  //Matrix3 L1, L2;
  //L1 = 0.5 * (F.Transpose() + F + F.Transpose() * F);
  //L2 = (L1(0,0) + L1(1,1) + L1(2,2)) * Identity;
  //pstress[idx] = E / (1.0 + v) * (L1 + v / (1.0 - 2.0 * v) * L2);

  //nope
  //LST = 0.5 * (F.Transpose() + F + F.Transpose() * F);
  //pstress[idx] = E / (1.0 + v) * (LST + v / (1.0 - 2.0 * v) * (LST(0,0) + LST
       (1,1) + LST(2,2)) * Identity);

  LST = F.Transpose() * F - Identity;
  LST = LST / 2.0;
  pstress[idx] = (E / (1.0 + v)) * (LST + (v / (1.0 - 2.0 * v)) * (LST(0,0) + LST
       (1,1) + LST(2,2)) * Identity);

  //ok
  //LST = F.Transpose() * F - Identity;
```

```
      //LST = LST / 2.0;
      //pstress[idx] = LST + (v / (1.0 - 2.0 * v)) * (LST(0,0) + LST(1,1) + LST(2,2))
          * Identity;
      //pstress[idx] = pstress[idx] * (E / 1.0 + v);

      // Update the total strain energy for all of the particles
      // todo - how do you calculate linear strain energy in three dimensions?
      se += 0;

      //Compute c_dil, the local speed of sound
      k = E / (3.0 * (1.0 - 2.0 * v));
      rho_current = rho_orig/J;
      c_dil = sqrt(k/rho_current); // Newton Laplace equation

      // Compute wave speed + particle velocity at each particle, then store the
          maximum
      WaveSpeed=Vector(Max(c_dil+fabs(pvelocity[idx].x()), WaveSpeed.x()),
                       Max(c_dil+fabs(pvelocity[idx].y()), WaveSpeed.y()),
                     Max(c_dil+fabs(pvelocity[idx].z()), WaveSpeed.z()));

      // Compute artificial viscosity term
      if (flag->d_artificial_viscosity) {
        throw InvalidValue("Artificial viscosity has not been implemented for this
            constitutive model", __FILE__, __LINE__);
        // To be added later
      }
      else{
        p_q[idx] = 0.0;
      }
    } // end loop over particles

    WaveSpeed = dx/WaveSpeed;
    double delT_new = WaveSpeed.minComponent();

    new_dw->put(delt_vartype(delT_new), lb->delTLabel, patch->getLevel());

    if (flag->d_reductionVars->accStrainEnergy || flag->d_reductionVars->
        strainEnergy) {
      new_dw->put(sum_vartype(se), lb->StrainEnergyLabel);
    }
    delete interpolator;
  }
}

double CompLinear::computeRhoMicroCM(double pressure, const double p_ref,
      const MPMMaterial* matl, double temperature, double rho_guess)
{
  double p_gauge = pressure - p_ref;
  double rho_orig = matl->getInitialDensity();
  double rho_cur = -1.0;
  bool error = false;

  if (d_useModifiedEOS && p_gauge < 0.0) { // Modified EOS

    double K = CalculateBulkModulus();
    double A = p_ref;
    double n = p_ref/K;
    rho_cur = rho_orig*pow(pressure/A,n);

  } else { // Standard EOS
```

```
    try {
      rho_cur = d_eos->computeDensity(rho_orig, -p_gauge);
    } catch (ConvergenceFailure& e) {
      cout << e.message() << endl;
      error = true;
    }
    if (error || rho_cur < 0.0 || isnan(rho_cur)) {
      ostringstream desc;
      desc << "rho_cur = " << rho_cur << " pressure = " << -p_gauge
           << " p_ref = " << p_ref << " 1/sp_vol_CC = " << rho_guess << endl;
      throw InvalidValue(desc.str(), __FILE__, __LINE__);
    }
  }

    return rho_cur;
}

void CompLinear::computePressEOSCM(const double rho_cur,double& pressure,
      const double p_ref, double& dp_drho, double& cSquared, const MPMMaterial
      * matl, double temperature)
{
  double rho_orig = matl->getInitialDensity();

  if (d_useModifiedEOS && rho_cur < rho_orig) { // Modified EOS

    double K = CalculateBulkModulus();
    double A = p_ref;
    double n = K/p_ref;
    double rho_rat_to_the_n = pow(rho_cur/rho_orig,n);
    pressure = A*rho_rat_to_the_n;
    dp_drho = (K/rho_cur)*rho_rat_to_the_n;
    cSquared = dp_drho; // Speed of sound squared

  } else { // Standard EOS

    double p = 0.0;
    d_eos->computePressure(rho_orig, rho_cur, p, dp_drho, cSquared);
    pressure = -p + p_ref;
    dp_drho = -dp_drho;
  }
}

double CompLinear::getCompressibility()
{
  return 1.0/CalculateBulkModulus();
}

double CompLinear::CalculateBulkModulus()
{
  // Get initial data
  double E = d_initialData.E;
  double v = d_initialData.v;

  // Calculate bulk modulus
  double K = E/(3*(1 - 2*v));

  return K;
}
```

## A.3 OGDEN MODEL ALGEBRA

Bower provides the following strain energy function for a compressible Ogden model [148]:

$$W = \frac{2\mu}{\alpha^2}(\bar{\lambda}_1^{\alpha} + \bar{\lambda}_2^{\alpha} + \bar{\lambda}_3^{\alpha} - 3) + \frac{1}{2}k(J-1)^2 \tag{147}$$

where

$$\bar{\lambda}_i = \lambda_i J^{\frac{1}{3}} \tag{148}$$

This is similar to that used in ADINA, this function was again used to obtain an equation for Cauchy stress. Given that:

$$J = \lambda_1\lambda_2\lambda_3 \tag{149}$$

Equation (147) may be written as:

$$W = \frac{2\mu}{\alpha^2}((\lambda_1(\lambda_1\lambda_2\lambda_3)^{\frac{1}{3}})^{\alpha} + (\lambda_2(\lambda_1\lambda_2\lambda_3)^{\frac{1}{3}})^{\alpha} + (\lambda_3(\lambda_1\lambda_2\lambda_3)^{\frac{1}{3}})^{\alpha} - 3) + \frac{1}{2}\kappa(\lambda_1\lambda_2\lambda_3 - 1)^2 \tag{150}$$

$$W = \frac{2\mu}{\alpha^2}\lambda_1^{\frac{4}{3}\alpha}\lambda_2^{\frac{1}{3}\alpha}\lambda_3^{\frac{1}{3}\alpha} + \frac{2\mu}{\alpha^2}\lambda_1^{\frac{1}{3}\alpha}\lambda_2^{\frac{4}{3}\alpha}\lambda_3^{\frac{1}{3}\alpha} + \frac{2\mu}{\alpha^2}\lambda_1^{\frac{1}{3}\alpha}\lambda_2^{\frac{1}{3}\alpha}\lambda_3^{\frac{4}{3}\alpha} - \frac{6\mu}{\alpha^2} + \frac{1}{2}\kappa(\lambda_1\lambda_2\lambda_3 - 1)^2 \tag{151}$$

The Cauchy stress may then be calculated as follows:

$$\begin{aligned}\frac{\partial W}{\partial \lambda_1} &= \frac{8\mu}{3\alpha}\lambda_1^{\frac{1}{3}\alpha}\lambda_2^{\frac{1}{3}\alpha}\lambda_3^{\frac{1}{3}\alpha} + \frac{2\mu}{3\alpha}\lambda_1^{-\frac{2}{3}\alpha}\lambda_2^{\frac{4}{3}\alpha}\lambda_3^{\frac{1}{3}\alpha} + \frac{2\mu}{3\alpha}\lambda_1^{-\frac{2}{3}\alpha}\lambda_2^{\frac{1}{3}\alpha}\lambda_3^{\frac{4}{3}\alpha} + \kappa(\lambda_1\lambda_2\lambda_3 - 1)\lambda_2\lambda_3 \\ &= \frac{8\mu}{3\alpha}J^{\frac{1}{3}\alpha} + \frac{2\mu}{3\alpha}\lambda_1^{-\frac{2}{3}\alpha}\lambda_2^{\frac{4}{3}\alpha}\lambda_3^{\frac{1}{3}\alpha} + \frac{2\mu}{3\alpha}\lambda_1^{-\frac{2}{3}\alpha}\lambda_2^{\frac{1}{3}\alpha}\lambda_3^{\frac{4}{3}\alpha} + \kappa(J-1)\lambda_2\lambda_3\end{aligned} \tag{152}$$

$$\begin{aligned}\frac{\partial W}{\partial \lambda_2} &= \frac{2\mu}{3\alpha}\lambda_1^{\frac{4}{3}\alpha}\lambda_2^{-\frac{2}{3}\alpha}\lambda_3^{\frac{1}{3}\alpha} + \frac{8\mu}{3\alpha}\lambda_1^{\frac{1}{3}\alpha}\lambda_2^{\frac{1}{3}\alpha}\lambda_3^{\frac{1}{3}\alpha} + \frac{2\mu}{3\alpha}\lambda_1^{\frac{1}{3}\alpha}\lambda_2^{-\frac{2}{3}\alpha}\lambda_3^{\frac{4}{3}\alpha} + \kappa(\lambda_1\lambda_2\lambda_3 - 1)\lambda_1\lambda_3 \\ &= \frac{2\mu}{3\alpha}\lambda_1^{\frac{4}{3}\alpha}\lambda_2^{-\frac{2}{3}\alpha}\lambda_3^{\frac{1}{3}\alpha} + \frac{8\mu}{3\alpha}J^{\frac{1}{3}\alpha} + \frac{2\mu}{3\alpha}\lambda_1^{\frac{1}{3}\alpha}\lambda_2^{-\frac{2}{3}\alpha}\lambda_3^{\frac{4}{3}\alpha} + \kappa(J-1)\lambda_1\lambda_3\end{aligned} \tag{153}$$

$$\frac{\partial W}{\partial \lambda_3} = \frac{2\mu}{3\alpha}\lambda_1^{\frac{4}{3}\alpha}\lambda_2^{\frac{1}{3}\alpha}\lambda_3^{-\frac{2}{3}\alpha} + \frac{2\mu}{3\alpha}\lambda_1^{\frac{1}{3}\alpha}\lambda_2^{\frac{4}{3}\alpha}\lambda_3^{-\frac{2}{3}\alpha} + \frac{8\mu}{3\alpha}\lambda_1^{\frac{1}{3}\alpha}\lambda_2^{\frac{1}{3}\alpha}\lambda_3^{\frac{1}{3}\alpha} + \kappa(\lambda_1\lambda_2\lambda_3 - 1)\lambda_1\lambda_2$$
$$= \frac{2\mu}{3\alpha}\lambda_1^{\frac{4}{3}\alpha}\lambda_2^{\frac{1}{3}\alpha}\lambda_3^{-\frac{2}{3}\alpha} + \frac{2\mu}{3\alpha}\lambda_1^{\frac{1}{3}\alpha}\lambda_2^{\frac{4}{3}\alpha}\lambda_3^{-\frac{2}{3}\alpha} + \frac{8\mu}{3\alpha}J^{\frac{1}{3}\alpha} + \kappa(J-1)\lambda_1\lambda_2 \tag{154}$$

$$\begin{aligned}\boldsymbol{\sigma} = \frac{1}{\lambda_1\lambda_2\lambda_3}[&\lambda_1(\frac{8\mu}{3\alpha}J^{\frac{1}{3}\alpha} + \frac{2\mu}{3\alpha}\lambda_1^{-\frac{2}{3}\alpha}\lambda_2^{\frac{4}{3}\alpha}\lambda_3^{\frac{1}{3}\alpha} + \frac{2\mu}{3\alpha}\lambda_1^{-\frac{2}{3}\alpha}\lambda_2^{\frac{1}{3}\alpha}\lambda_3^{\frac{4}{3}\alpha} + \kappa(J-1)\lambda_2\lambda_3)\mathbf{n}_1\otimes\mathbf{n}_1\\ &+\lambda_2(\frac{2\mu}{3\alpha}\lambda_1^{\frac{4}{3}\alpha}\lambda_2^{-\frac{2}{3}\alpha}\lambda_3^{\frac{1}{3}\alpha} + \frac{8\mu}{3\alpha}J^{\frac{1}{3}\alpha} + \frac{2\mu}{3\alpha}\lambda_1^{\frac{1}{3}\alpha}\lambda_2^{-\frac{2}{3}\alpha}\lambda_3^{\frac{4}{3}\alpha} + \kappa(J-1)\lambda_1\lambda_3)\mathbf{n}_2\otimes\mathbf{n}_2\\ &+\lambda_3(\frac{2\mu}{3\alpha}\lambda_1^{\frac{4}{3}\alpha}\lambda_2^{\frac{1}{3}\alpha}\lambda_3^{-\frac{2}{3}\alpha} + \frac{2\mu}{3\alpha}\lambda_1^{\frac{1}{3}\alpha}\lambda_2^{\frac{4}{3}\alpha}\lambda_3^{-\frac{2}{3}\alpha} + \frac{8\mu}{3\alpha}J^{\frac{1}{3}\alpha} + \kappa(J-1)\lambda_1\lambda_2)\mathbf{n}_3\otimes\mathbf{n}_3]\end{aligned} \tag{155}$$

## A.4 MOONEY–RIVLIN USING PRINCIPAL STRETCHES

Having determined that the principal stretches, principal directions, and Jacobian were all being correctly calculated; two versions of a standard compressible Mooney-Rivlin model were implemented. The first was in terms of the invariants and the second was implemented in terms of the principal stretches, which is atypical. The Mooney-Rivlin model using the principal stretches was implmented in a way very similar to the Ogden model described previously. The only differences were the inputs for the material constants and the strain energy density function used. The colliding disks test was run for both versions, the results are equivalent. Both versions worked correctly, the unusual pulsating behaviour observed with the Ogden model was not observed here. At this point, having systematically rules out all alternatives, it was possible to conclude that the problem must lie with the Ogden strain energy density function itself.

The strain energy for a compressible Mooney-Rivlin hyperelastic model is given by:

$$W = C_1(\bar{I_1} - 3) + C_2(\bar{I_2} - 3) + D(J-1)^2 \tag{156}$$

where:

$$\frac{\partial W}{\partial \bar{I_1}} = C_1, \quad \frac{\partial W}{\partial \bar{I_2}} = C_2, \quad \frac{\partial W}{\partial J} = -2D(J-1) \tag{157}$$

The bulk and shear moduli relate to the constants as follows:

$$\kappa = 2D \tag{158}$$

$$\mu = 2(C_1 + C_2) \tag{159}$$

The Cauchy stress is given by:

$$\boldsymbol{\sigma} = \frac{2}{J}[\frac{1}{J^{\frac{2}{3}}}(C_1 + \bar{I_1}C_2)\mathbf{B} - \frac{1}{J^{\frac{4}{3}}}C_2\mathbf{B}\cdot\mathbf{B}] + [2D(J-1) - \frac{2}{3J}(C_1\bar{I_1} + 2C_2\bar{I_2})]\mathbf{1} \tag{160}$$

The Mooney-Rivlin model is typically expressed in terms of the invariants, although it is also possible to write it in terms of the principal stretches. Given that:

$$\bar{I_1} = J^{-\frac{2}{3}}I_1 = J^{-\frac{2}{3}}(\lambda_1^2 + \lambda_2^2 + \lambda_3^2) \tag{161}$$

$$\bar{I_2} = J^{-\frac{4}{3}}I_2 = J^{-\frac{4}{3}}(\lambda_1^2\lambda_2^2 + \lambda_2^2\lambda_3^2 + \lambda_3^2\lambda_1^2) \tag{162}$$

it is possible to rewrite the strain energy as:

$$\begin{aligned} W &= C_1(J^{-\frac{2}{3}}I_1 - 3) + C_2(J^{-\frac{4}{3}}I_2 - 3) + D(J-1)^2 \\ &= C_1(J^{-\frac{2}{3}}(\lambda_1^2 + \lambda_2^2 + \lambda_3^2) - 3) + C_2(J^{-\frac{4}{3}}(\lambda_1^2\lambda_2^2 + \lambda_2^2\lambda_3^2 + \lambda_3^2\lambda_1^2) - 3) + D(J-1)^2 \end{aligned} \tag{163}$$

which may be rewritten as:

$$\begin{aligned} W = C_1(\lambda_1^4\lambda_2^{-2}\lambda_3^{-2} + \lambda_1^{-2}\lambda_2^4\lambda_3^{-2} + \lambda_1^{-2}\lambda_2^{-2}\lambda_3^4 - 3)^{\frac{1}{3}} \\ + C_2(\lambda_1^2\lambda_2^2\lambda_3^{-4} + \lambda_1^{-4}\lambda_2^2\lambda_3^2 + \lambda_1^2\lambda_2^{-4}\lambda_3^2 - 3)^{\frac{1}{3}} \\ + D\lambda_1^2\lambda_2^2\lambda_3^2 - 2D\lambda_1\lambda_2\lambda_3 + D \end{aligned} \tag{164}$$

Cauchy stress may then again be determined

$$\begin{aligned} \frac{\partial W}{\partial \lambda_1} = C_1(4\lambda_1^3\lambda_2^{-2}\lambda_3^{-2} - 2\lambda_1^{-3}\lambda_2^4\lambda_3^{-2} - 2\lambda_1^{-3}\lambda_2^{-2}\lambda_3^4)^{\frac{1}{3}} \\ + C_2(2\lambda_1\lambda_2^2\lambda_3^{-4} - 4\lambda_1^{-5}\lambda_2^2\lambda_3^2 + 2\lambda_1\lambda_2^{-4}\lambda_3^2)^{\frac{1}{3}} \\ + 2D\lambda_1\lambda_2^2\lambda_3^2 - 2D\lambda_2\lambda_3 \end{aligned} \tag{165}$$

$$\begin{aligned} \frac{\partial W}{\partial \lambda_2} = C_1(-2\lambda_1^4\lambda_2^{-3}\lambda_3^{-2} + 4\lambda_1^{-2}\lambda_2^3\lambda_3^{-2} - 2\lambda_1^{-2}\lambda_2^{-3}\lambda_3^4)^{\frac{1}{3}} \\ + C_2(2\lambda_1^2\lambda_2\lambda_3^{-4} + 2\lambda_1^{-4}\lambda_2\lambda_3^2 - 4\lambda_1^2\lambda_2^{-5}\lambda_3^2)^{\frac{1}{3}} \\ + 2D\lambda_1^2\lambda_2\lambda_3^2 - 2D\lambda_1\lambda_3 \end{aligned} \tag{166}$$

$$\begin{aligned}\frac{\partial W}{\partial \lambda_3} = C_1(&-2\lambda_1^4\lambda_2^{-2}\lambda_3^{-3} - 2\lambda_1^{-2}\lambda_2^4\lambda_3^{-3} + 4\lambda_1^{-2}\lambda_2^{-2}\lambda_3^3)^{\frac{1}{3}} \\ &+C_2(-4\lambda_1^2\lambda_2^2\lambda_3^{-5} + 2\lambda_1^{-4}\lambda_2^2\lambda_3 + 2\lambda_1^2\lambda_2^{-4}\lambda_3)^{\frac{1}{3}} \\ &+2D\lambda_1^2\lambda_2^2\lambda_3 - 2D\lambda_1\lambda_2\end{aligned} \tag{167}$$

$$\begin{aligned}\boldsymbol{\sigma} = \frac{1}{\lambda_1\lambda_2\lambda_3}[&\lambda_1(C_1(4\lambda_1^3\lambda_2^{-2}\lambda_3^{-2} - 2\lambda_1^{-3}\lambda_2^4\lambda_3^{-2} - 2\lambda_1^{-3}\lambda_2^{-2}\lambda_3^4)^{\frac{1}{3}} \\ &+C_2(2\lambda_1\lambda_2^2\lambda_3^{-4} - 4\lambda_1^{-5}\lambda_2^2\lambda_3^2 + 2\lambda_1\lambda_2^{-4}\lambda_3^2)^{\frac{1}{3}} \\ &+2D\lambda_1\lambda_2^2\lambda_3^2 - 2D\lambda_2\lambda_3)\mathbf{n}_1 \otimes \mathbf{n}_1 \\ &+\lambda_2(C_1(-2\lambda_1^4\lambda_2^{-3}\lambda_3^{-2} + 4\lambda_1^{-2}\lambda_2^3\lambda_3^{-2} - 2\lambda_1^{-2}\lambda_2^{-3}\lambda_3^4)^{\frac{1}{3}} \\ &+C_2(2\lambda_1^2\lambda_2\lambda_3^{-4} + 2\lambda_1^{-4}\lambda_2\lambda_3^2 - 4\lambda_1^2\lambda_2^{-5}\lambda_3^2)^{\frac{1}{3}} \\ &+2D\lambda_1^2\lambda_2\lambda_3^2 - 2D\lambda_1\lambda_3)\mathbf{n}_2 \otimes \mathbf{n}_2 \\ &+\lambda_3(C_1(-2\lambda_1^4\lambda_2^{-2}\lambda_3^{-3} - 2\lambda_1^{-2}\lambda_2^4\lambda_3^{-3} + 4\lambda_1^{-2}\lambda_2^{-2}\lambda_3^3)^{\frac{1}{3}} \\ &+C_2(-4\lambda_1^2\lambda_2^2\lambda_3^{-5} + 2\lambda_1^{-4}\lambda_2^2\lambda_3 + 2\lambda_1^2\lambda_2^{-4}\lambda_3)^{\frac{1}{3}} \\ &+2D\lambda_1^2\lambda_2^2\lambda_3 - 2D\lambda_1\lambda_2)\mathbf{n}_3 \otimes \mathbf{n}_3]\end{aligned} \tag{168}$$

[1] L. H. S. Sekhon and M. G. Fehlings. "Epidemiology, demographics, and pathophysiology of acute spinal cord injury." In: *Spine* 26.24 (2001). Times Cited: 192 S, S2–S12.

[2] B Gardner and P. J. Kluger. "(iv) The overall care of the spinal cord injured patient." In: *Current Orthopaedics* 18.1 (2004), pp. 33 –48.

[3] F. M. Maynard et al. "International standards for neurological and functional classification of spinal cord injury." In: *Spinal Cord* 35.5 (1997). ISI Document Delivery No.: WZ802 Times Cited: 730, pp. 266–274.

[4] W. O. McKinley, A. B. Jackson, D. D. Cardenas, and M. J. DeVivo. "Long-term medical complications after traumatic spinal cord injury: A regional model systems analysis." In: *Archives of Physical Medicine and Rehabilitation* 80.11 (1999). Times Cited: 194, pp. 1402–1410.

[5] M. J. DeVivo. "Causes and costs of spinal cord injury in the United States." In: *Spinal Cord* 35.12 (1997). ISI Document Delivery No.: YN203 Times Cited: 98 Cited Reference Count: 18 DeVivo, MJ Stockton press Basingstoke, pp. 809–813.

[6] A. M. Choo, J. Liu, C. K. Lam, M. Dvorak, W. Tetzlaff, and T. R. Oxland. "Contusion, dislocation, and distraction: primary hemorrhage and membrane permeability in distinct mechanisms of spinal cord injury." In: *Journal of Neurosurgery-Spine* 6.3 (2007). Times Cited: 27, pp. 255–266.

[7] A. M. Choo, J. Liu, M. Dvorak, W. Tetzlaff, and T. R. Oxland. "Secondary pathology following contusion, dislocation, and distraction spinal cord injuries." In: *Experimental Neurology* 212.2 (2008). Times Cited: 17, pp. 490–506.

[8] M. D. Norenberg, J. Smith, and A. Marcillo. "The pathology of human spinal cord injury: Defining the problems." In: *Journal of Neurotrauma* 21.4 (2004). Times Cited: 110, pp. 429–440.

[9] R. M. Hall, R. J. Oakland, R. K. Wilcox, and D. C. Barton. "Spinal cord-fragment interactions following burst fracture: an in vitro model." In: *Journal of Neurosurgery-Spine* 5.3 (2006). Times Cited: 11, pp. 243–250.

[10] R. K. Wilcox, T. O. Boerger, D. J. Allen, D. C. Barton, D. Limb, R. A. Dickson, and R. M. Hall. "A dynamic study of thoracolumbar burst fractures." In: *Journal of Bone and Joint Surgery-american Volume* 85A.11 (Nov. 2003), pp. 2184–2189.

[11] R. K. Wilcox, T. O. Boerger, R. M. Hall, D. C. Barton, D. Limb, and R. A. Dickson. "Measurement of canal occlusion during the thoracolumbar burst fracture process." In: *Journal of Biomechanics* 35.3 (Mar. 2002), pp. 381–384.

[12] I. Yamaura, K. Yone, S. Nakahara, T. Nagamine, H. Baba, K. Uchida, and S. Komiya. "Mechanism of destructive pathologic changes in the spinal cord under chronic mechanical compression." In: *Spine* 27.1 (2002). Times Cited: 31, pp. 21–26.

[13] J. W. McDonald and C. Sadowsky. "Spinal-cord injury." In: *Lancet* 359.9304 (2002). Times Cited: 122, pp. 417–425.

[14] C. Persson, J. L. Summers, and R. M. Hall. "Modelling of Spinal Cord Biomechanics: In Vitro and Computational Approaches." In: *Neural Tissue Biomechanics*. Ed. by L. Bilston. Vol. 3. Studies in Mechanobiology Tissue Engineering and Biomaterials. Springer, 2011, pp. 181–201.

[15] E. C. Clarke and L. E. Bilston. "Contrasting biomechanics and neuropathology of spinal cord injury in neonatal and adult rats following vertebral dislocation." In: *Journal of Neurotrauma* 25.7 (2008). ISI Document Delivery No.: 326BX Times Cited: 3 Cited Reference Count: 41 Clarke, Elizabeth C. Bilston, Lynne E. Mary ann liebert inc New rochelle, pp. 817–832.

[16] R. J. Fiford, L. E. Bilston, P. Waite, and J. Lu. "A vertebral dislocation model of spinal cord injury in rats." In: *Journal of Neurotrauma* 21.4 (2004). Times Cited: 17, pp. 451–458.

[17] R. J. Fiford. "Biomechanics of spinal cord injury in a novel rat model." PhD thesis. School of Aerospace, Mechanical and Mechatronic Engineering, University of Sydney, 2005.

[18] C. Y. Greaves, M. S. Gadala, and T. R. Oxland. "A three-dimensional finite element model of the cervical spine with spinal cord: An investigation of three injury mechanisms." In: *Annals of Biomedical Engineering* 36.3 (2008). Times Cited: 15, pp. 396–405.

[19] C. J. Sparrey, A. M. Choo, J. Liu, W. Tetzlaff, and T. R. Oxland. "The Distribution of Tissue Damage in the Spinal Cord Is Influenced by the Contusion Velocity." In: *Spine* 33.22 (2008). Times Cited: 9, E812–E819.

[20] K. Moore, A. Agur, and A. Dalley. *Essential Clinical Anatomy*. Lippincott Williams & Wilkins, 2010.

[21] R. Drake, W. Vogl, A. Mitchell, and H. Gray. *Gray's Anatomy for Students*. Grays Anatomy for Students pt. 762. Churchill Livingstone/Elsevier, 2010.

[22] D. U. Silverthorn. *Human physiology: an integrated approach / Dee Unglaub Silverthorn ; with contributions by Bruce R. Johnson and William C. Ober, illustration coordinator ; Claire W. Garrison, illustrator ; Andrew C. Silverthorn, clinical consultant*. 6th ed. Boston : Pearson Education, 2013.

[23] J. Sobotta and J. P. McMurrich. *Atlas and textbook of human anatomy*. Philadelphia: Saunders, 1909, pp. 3 v.–.

[24] G. Mitchell and D. Mayor. *The Essentials of Neuroanatomy*. Churchill Livingstone Medical Text. Churchill Livingstone, 1983.

[25] H. Gray. *Anatomy of the human body*. Ed. by W. H. Lewis. Philadelphia: Lea & Febiger, 1918.

[26] C. Persson, J. Summers, and R. M. Hall. "The Importance of Fluid-Structure Interaction in Spinal Trauma Models." In: *Journal of Neurotrauma* 28.1 (2011). Times Cited: 1, pp. 113–125.

[27] J. Kiernan. *Barr's The Human Nervous System: An Anatomical Viewpoint*. Lippincott Williams & Wilkins, 2008.

[28] E. Furnish and C. Schmidt. "Tissue engineering of the peripheral nervous system." In: ed. by C. W. J. Patrick, A. G. Mikos, and L. V. McIntire. New York: Elsevier Science, 1998. Chap. III.10, pp. 514–535.

[29] J. Winter and C. Schmidt. "Biomimetic Strategies and Applications in the Nervous System." In: *Biomimetic Materials And Design*. Ed. by A. L. A Dillow. doi:10.1201/9780203908976.ch12. CRC Press, 2002. Chap. 12.

[30] Human Anatomy Chart. *Spinal Cord Injury Levels*. Aug. 2015. URL: http://humananatomychart.us/tag/spinal-cord-injury-levels/.

[31] C. Schmidt and B. Leach J. "Neural tissue engineering: Strategies for repair and regeneration." In: *Annual Review of Biomedical Engineering* 5 (2003). Times Cited: 323, pp. 293–347.

[32] A. Castano, M. D. Bell, and V. H. Perry. "Unusual aspects of inflammation in the nervous system: Wallerian degeneration." In: *Neurobiology of Aging* 17.5 (1996). <ce:title>Inflammatory Mechanisms and Anti-Inflammatory Therapy in Alzheimer's Disease</ce:title>, pp. 745 –751.

[33] A. M. Avellino, D. Hart, A. T. Dailey, M. Mackinnon, D. Ellegala, and M. Kliot. "Differential Macrophage Responses In The Peripheral And Central-nervous-system During Wallerian Degeneration Of Axons." In: *Experimental Neurology* 136.2 (1995). Times Cited: 98, pp. 183–198.

[34] J. W. Fawcett and R. A. Asher. "The glial scar and central nervous system repair." In: *Brain Research Bulletin* 49.6 (1999). Times Cited: 771, pp. 377–391.

[35] S. Thuret, L. D. F. Moon, and F. H. Gage. "Therapeutic interventions after spinal cord injury." In: *Nature Reviews Neuroscience* 7.8 (2006). Times Cited: 185, pp. 628–643.

[36] M. Bahr and F. Bonhoeffer. "Perspectives On Axonal Regeneration In The Mammalian CNS." In: *Trends in Neurosciences* 17.11 (1994). ISI Document Delivery No.: PN805 Times Cited: 101 Cited Reference Count: 67 Bahr, m bonhoeffer, f Elsevier sci ltd Oxford, pp. 473–479.

[37] C. M. Mann and B. K. Kwon. "An Update on the Pathophysiology of Acute Spinal Cord Injury." In: *Seminars in Spine Surgery* 19.4 (2007), pp. 272–279.

[38] B. Winter and H. Pattani. "Spinal cord injury." In: *Anaesthesia & Intensive Care Medicine* 12.9 (2011), pp. 403–405.

[39] C. H. Tator and I. Koyanagi. "Vascular mechanisms in the pathophysiology of human spinal cord injury." In: *Journal of Neurosurgery* 86.3 (Mar. 1997), pp. 483–492.

[40] C. Matute, M. V. SanchezGomez, L. MartinezMillan, and R. Miledi. "Glutamate receptor-mediated toxicity in optic nerve oligodendrocytes." In: *Proceedings of the National Academy of Sciences of the United States of America* 94.16 (Aug. 1997), pp. 8830–8835.

[41] J. W. McDonald, S. P. Althomsons, K. L. Hyrc, D. W. Choi, and M. P. Goldberg. "Oligodendrocytes from forebrain are highly vulnerable to AMPA/kainate receptor-mediated excitotoxicity." In: *Nature Medicine* 4.3 (Mar. 1998), pp. 291–297.

[42] M. J. Crowe, J. C. Bresnahan, S. L. Shuman, J. N. Masters, and M. S. Beattie. "Apoptosis and delayed degeneration after spinal cord injury in rats and monkeys." In: *Nature Medicine* 3.1 (1997). Times Cited: 537, pp. 73–76.

[43] L. Y. Zhang, K. H. Yang, and A. I. King. "Biomechanics of neurotrauma." In: *Neurological Research* 23.2-3 (Mar. 2001), pp. 144–156.

[44] C. H. Tator and M. G. Fehlings. "Review of clinical trials of neuroprotection in acute spinal cord injury." In: *Neurosurgical focus* 6.1 (1999), e8.

[45] J. M. Jensen and R. Y. Shi. "Effects of 4-aminopyridine on stretched mammalian spinal cord: The role of potassium channels in axonal conduction." In: *Journal of Neurophysiology* 90.4 (2003). Times Cited: 31, pp. 2334–2340.

[46] S. S. Haghighi, S. K. Agrawal, D. Surdell, R. Plambeck, S. Agrawal, G. C. Johnson, and A. Walker. "Effects of methylprednisolone and MK-801 on functional recovery after experimental chronic spinal cord injury." In: *Spinal Cord* 38.12 (2000). Times Cited: 18, pp. 733–740.

[47] J. K. Hyun and H.-W. Kim. "Clinical and experimental advances in regeneration of spinal cord injury." In: *Journal of tissue engineering* 2010 (2010), p. 650857.

[48] M. B. Bunge and D. D. Pearse. "Transplantation strategies to promote repair of the injured spinal cord." In: *Journal of Rehabilitation Research and Development* 40.4 (2003). Times Cited: 70 1 Conference on Translational Research in Spinal Cord Injury Apr 03, 2003 Miami, florida, pp. 55–62.

[49] D. M. Geddes-Klein, K. B. Schiffman, and D. F. Meaney. "Mechanisms and consequences of neuronal stretch injury in vitro differ with the model of trauma." In: *Journal of Neurotrauma* 23.2 (2006). Times Cited: 47, pp. 193–204.

[50] C. H. Tator. "Review of treatment trials in human spinal cord injury: Issues, difficulties, and recommendations." In: *Neurosurgery* 59.5 (2006). Times Cited: 79, pp. 957–982.

[51] L. Bilston. "Finite element analysis of some cervical spinal cord injury modes." In: *Proceedings Of The 1998 International Ircobi Conference On The Biomechanics Of Impact*. 1998.

[52] M. Czyz, K. Scigala, R. Bedzinski, and W. Jarmundowicz. "Finite element modelling of the cervical spinal cord injury - clinical assessment." In: *Acta of Bioengineering and Biomechanics* 14.4 (2012), pp. 23–29.

[53] X.-F. Li and L.-Y. Dai. "Three-Dimensional Finite Element Model of the Cervical Spinal Cord Preliminary Results of Injury Mechanism Analysis." In: *Spine* 34.11 (2009). Times Cited: 7, pp. 1140–1147.

[54] C. M. Russell, A. M. Choo, W. Tetzlaff, T. E. Chung, and T. R. Oxland. "Maximum Principal Strain Correlates with Spinal Cord Tissue Damage in Contusion and Dislocation Injuries in the Rat Cervical Spine." In: *Journal of Neurotrauma* 29.8 (2012). ISI Document Delivery No.: 939NJ Times Cited: 0, pp. 1574–1585.

[55] D. K. Cullen and M. C. LaPlaca. "Neuronal response to high rate shear deformation depends on heterogeneity of the local strain field." In: *Journal of Neurotrauma* 23.9 (Sept. 2006), pp. 1304–1319.

[56] E. East and J. B. Phillips. "Tissue engineered cell culture models for nervous system research." In: *Tissue Engineering Research Trends*. Ed. by G. N. Greco. Nova Science Publishers, Inc, 2008, pp. 141–160.

[57] J. B. Phillips and R. Brown. "Micro-structured Materials and Mechanical Cues in 3D Collagen Gels." In: *3d Cell Culture: Methods and Protocols* 695 (2011), pp. 183–196.

[58] E. East, J. P. Golding, and J. B. Phillips. "Engineering an Integrated Cellular Interface in Three-Dimensional Hydrogel Cultures Permits Monitoring of Reciprocal Astrocyte and Neuronal Responses." In: *Tissue Engineering Part C-Methods* 18.7 (2012). Times Cited: 0, pp. 526–536.

[59] A. R. Allen. “Surgery Of Experimental Lesion Of Spinal Cord Equivalent To Crush Injury Of Fracture Dislocation Of Spinal Column.” In: *Journal of the American Medical Association* LVII.11 (1911), pp. 878–880.

[60] A. M. Gerber and W. S. Corrie. “Effect of Impounder Contact Area On Experimental Spinal-cord Injury.” In: *Journal of Neurosurgery* 51.4 (1979), pp. 539–542.

[61] B. T. Stokes and L. B. Jakeman. “Experimental modelling of human spinal cord injury: a model that crosses the species barrier and mimics the spectrum of human cytopathology.” In: *Spinal Cord* 40.3 (Mar. 2002), pp. 101–109.

[62] P. A. Kearney, S. A. Ridella, D. C. Viano, and T. E. Anderson. “Interaction of Contact Velocity and Cord Compression In Determining the Severity of Spinal Cord Injury.” In: *Journal of Neurotrauma* 5.3 (1988), pp. 187–208.

[63] B. A. Kakulas. “Pathology of spinal injuries.” In: *Central nervous system trauma : journal of the American Paralysis Association* 1.2 (1984), pp. 117–29.

[64] C. H. Tator. “Spine-spinal cord relationships in spinal cord trauma.” In: *Clinical neurosurgery* 30 (1983), pp. 479–94.

[65] J. W. Carter, S. K. Mirza, A. F. Tencer, and R. P. Ching. “Canal geometry changes associated with axial compressive cervical spine fracture.” In: *Spine* 25.1 (Jan. 2000), pp. 46–54.

[66] D. G. Chang, A. F. Tencer, R. P. Ching, B. Treece, D. Senft, and P. A. Anderson. “Geometric Changes In the Cervical Spinal-canal During Impact.” In: *Spine* 19.8 (Apr. 1994), pp. 973–980.

[67] B. E. Fredrickson, W. T. Edwards, W. Rauschning, J. C. Bayley, and H. A. Yuan. “1992 Volvo Award In Experimental Studies - Vertebral Burst Fractures - An Experimental, Morphologic, and Radiographic Study.” In: *Spine* 17.9 (Sept. 1992), pp. 1012–1201.

[68] M. M. Panjabi, M. Kifune, L. Wen, M. Arand, T. R. Oxland, R. M. Lin, W. S. S. Yoon, and A. Vasavada. “Dynamic Canal Encroachment During Thoracolumbar Burst Fractures.” In: *Journal of Spinal Disorders* 8.1 (Feb. 1995), pp. 39–48.

[69] N. T. Tran, N. A. Watson, A. F. Tencer, R. P. Ching, and P. A. Anderson. “Mechanism of the Burst Fracture In the Thoracolumbar Spine - the Effect of Loading Rate.” In: *Spine* 20.18 (Sept. 1995), pp. 1984–1988.

[70] J. Willen, S. Lindahl, L. Irstam, B. Aldman, and A. Nordwall. “The Thoracolumbar Crush Fracture - An Experimental-study On Instant Axial Dynamic Loading - the Resulting Fracture Type and Its Stability.” In: *Spine* 9.6 (1984), pp. 624–631.

[71] C. Persson, S. W. D. McLure, J. Summers, and R. M. Hall. "The effect of bone fragment size and cerebrospinal fluid on spinal cord deformation during trauma: an ex vivo study Laboratory investigation." In: *Journal of Neurosurgery-Spine* 10.4 (2009). Times Cited: 3, pp. 315–323.

[72] D. Limb, D. L. Shaw, and R. A. Dickson. "Neurological Injury In Thoracolumbar Burst Fractures." In: *Journal of Bone and Joint Surgery-british Volume* 77.5 (Sept. 1995), pp. 774–777.

[73] C. Persson. "Biomechanical Modelling Of The Spinal Cord And Bone Fragment Interactions During A Vertebral Burst Fracture." PhD thesis. University of Leeds, School of Mechanical Engineering, 2009.

[74] C. F. Jones, S. G. Kroeker, P. A. Cripton, and R. M. Hall. "The effect of cerebrospinal fluid on the biomechanics of spinal cord - An ex vivo bovine model using bovine and physical surrogate spinal cord." In: *Spine* 33.17 (2008). Times Cited: 9, E580–E588.

[75] C. Persson, J. Summers, and R. M. Hall. "The Effect of Cerebrospinal Fluid Thickness on Traumatic Spinal Cord Deformation." In: *Journal of Applied Biomechanics* 27.4 (2011). Times Cited: 0, pp. 330–335.

[76] C. F. Jones, B. K. Kwon, and P. A. Cripton. "Mechanical indicators of injury severity are decreased with increased thecal sac dimension in a bench-top model of contusion type spinal cord injury." In: *Journal of Biomechanics* 45.6 (2012), pp. 1003–1010.

[77] I. G. Bloomfield, I. H. Johnston, and L. E. Bilston. "Effects of proteins, blood cells and glucose on the viscosity of cerebrospinal fluid." In: *Pediatric Neurosurgery* 28.5 (1998). Times Cited: 38, pp. 246–251.

[78] R. W. Ellis, L. C. Strauss, J. M. Wiley, and T. M. Killmond. "A Simple Method Of Estimating Cerebrospinal-fluid Pressure During Lumbar Puncture." In: *Pediatrics* 89.5 (1992). Times Cited: 10 Part 1, pp. 895–897.

[79] M. G. Richardson and R. N. Wissler. "Density of lumbar cerebrospinal fluid in pregnant and nonpregnant humans." In: *Anesthesiology* 85.2 (1996). Times Cited: 49, pp. 326–330.

[80] M. G. Richardson and R. N. Wissler. "Densities of dextrose-free intrathecal local anesthetics, opioids, and combinations measured at 37 degrees C." In: *Anesthesia and Analgesia* 84.1 (1997). Times Cited: 48 1996 Annual Meeting of the American-Society-of-Regional-Anesthesia Mar 28-31, 1996 San diego, ca Amer Soc Reg Anesthesia, pp. 95–99.

[81] L. E. Bilston and L. E. Thibault. "The mechanical properties of the human cervical spinal cord in vitro." In: *Annals of Biomedical Engineering* 24.1 (1996). Times Cited: 65, pp. 67–74.

[82] J. Holsheimer, J. A. Denboer, J. J. Struijk, and A. R. Rozeboom. "MR Assessment Of The Normal Position Of The Spinal-cord In The Spinal-Canal." In: *American Journal of Neuroradiology* 15.5 (1994). Times Cited: 52, pp. 951–959.

[83] R. J. Oakland, R. M. Hall, R. K. Wilcox, and D. C. Barton. "The biomechanical response of spinal cord tissue to uniaxial loading." In: *Proceedings of the Institution of Mechanical Engineers Part H-Journal of Engineering in Medicine* 220.H4 (2006). Times Cited: 19, pp. 489–492.

[84] J. N. Reddy. *An introduction to the finite element method*. 2nd ed. (Junuthula Narasimha). New York ; London : McGraw-Hill, 1993.

[85] H.-J. Bungartz, M. Mehl, and M. Schafer. *Fluid structure interaction II : modelling, simulation, optimization*. Berlin : Springer Verlag, 2010.

[86] B. Galle, H. Ouyang, R. Shi, and E. Nauman. "Correlations between tissue-level stresses and strains and cellular damage within the guinea pig spinal cord white matter." In: *Journal of Biomechanics* 40.13 (2007), pp. 3029–3033.

[87] K. Ichihara, T. Taguchi, I. Sakuramoto, S. Kawano, and S. Kawai. "Mechanism of the spinal cord injury and the cervical spondylotic myelopathy: new approach based on the mechanical features of the spinal cord white and gray matter." In: *Journal of Neurosurgery* 99.3 (Oct. 2003), pp. 278–285.

[88] H. Ouyang, B. Galle, J. Li, E. Nauman, and R. Shi. "Biomechanics of spinal cord injury: a multimodal investigation using ex vivo guinea pig spinal cord white matter." In: *Journal of neurotrauma* 25.1 (Jan. 2008), pp. 19–29.

[89] G. L. Chang, T. K. Hung, A. Bleyaert, and P. J. Jannetta. "Stress-strain Measurement of the Spinal-cord of Puppies and Their Neurological Evaluation." In: *Journal of Trauma-injury Infection and Critical Care* 21.9 (1981), pp. 807–810.

[90] T. K. Hung and G. L. Chang. "Biomechanical and Neurological Response of the Spinal-cord of A Puppy To Uniaxial Tension." In: *Journal of Biomechanical Engineering* 103.1 (1981), pp. 43–47.

[91] T. K. Hung, G. L. Chang, H. S. Lin, F. R. Walter, and L. Bunegin. "Stress-strain Relationship of the Spinal-cord of Anesthetized Cats." In: *Journal of Biomechanics* 14.4 (1981), pp. 269–276.

[92] J. D. Reid. "Effects of Flexion-extension Movements of the Head and Spine Upon the Spinal Cord and Nerve Roots." In: *Journal of Neurology Neurosurgery and Psychiatry* 23.3 (1960), pp. 214–221.

[93] J. T. Maikos, Z. Qian, D. Metaxas, and D. I. Shreiber. "Finite element analysis of spinal cord injury in the rat." In: *Journal of Neurotrauma* 25.7 (2008). Times Cited: 13, pp. 795–816.

[94] W. Young. "Spinal cord contusion models." In: *Spinal Cord Trauma: Regeneration, Neural Repair and Functional Recovery* 137 (2002), pp. 231–255.

[95] J. T. Maikos, R. A. I. Elias, and D. I. Shreiber. "Mechanical properties of dura mater from the rat brain and spinal cord." In: *Journal of Neurotrauma* 25.1 (2008). Times Cited: 24, pp. 38–51.

[96] C. Persson, S. Evans, R. Marsh, J. L. Summers, and R. M. Hall. "Poisson's Ratio and Strain Rate Dependency of the Constitutive Behavior of Spinal Dura Mater." In: *Annals of Biomedical Engineering* 38.3 (2010). Times Cited: 3, pp. 975–983.

[97] T. K. Hung, H. S. Lin, L. Bunegin, and M. S. Albin. "Mechanical and Neurological Response of Cat Spinal-cord Under Static Loading." In: *Surgical Neurology* 17.3 (1982), pp. 213–217.

[98] D. J. Maiman, J. Coats, and J. B. Myklebust. "Cord/spine motion in experimental spinal cord injury." In: *Journal of spinal disorders* 2.1 (1989), pp. 14–9.

[99] H. Zhang and K.-J. Bathe. "Direct and iterative computing of fluid flows fully coupled with structures." In: *Computational Fluid and Solid Mechanics*. Ed. by K.-J. Bathe. First MIT Conference on CFSM. Elsevier Science Ltd., 2001, pp. 1440–1443.

[100] R. J. Oakland. "A biomechanical study of the spinal cord in the burst fracture process." PhD thesis. School of Mechanical Engineering. Leeds, University of Leeds, 2003.

[101] E. Mazuchowski and L. Thibault. "Biomechanical Properties Of The Human Spinal Cord And Pia Mater." In: *Summer Bioengineering Conference. Sonesta Beach Resort in Key Biscayne, Florida.* 2003.

[102] M. V. Kulkarni, C. B. Mcardle, D. Kopanicky, M. Miner, H. B. Cotler, K. F. Lee, and J. H. Harris. "Acute Spinal-cord Injury - MR Imaging At 1.5-T." In: *Radiology* 164.3 (Sept. 1987), pp. 837–843.

[103] A. Saifuddin, H. Noordeen, B. A. Taylor, and I. Bayley. "The role of imaging in the diagnosis and management of thoracolumbar burst fractures: Current concepts and a review of the literature." In: *Skeletal Radiology* 25.7 (Oct. 1996), pp. 603–613.

[104] X. Z. Liu et al. "Neuronal and glial apoptosis after traumatic spinal cord injury." In: *Journal of Neuroscience* 17.14 (July 1997), pp. 5395–5406.

[105] E. C. Clarke. "Spinal Cord Mechanical Properties." In: *Neural Tissue Biomechanics*. Vol. 3. Springer, 2011, pp. 25–40.

[106] J. T. Maikos and D. I. Shreiber. "Immediate damage to the blood-spinal cord barrier due to mechanical trauma." In: *Journal of Neurotrauma* 24.3 (2007). Times Cited: 25, pp. 492–507.

[107] L. B. Jakeman, Z. Guan, P. Wei, R. Ponnappan, R. Dzwonczyk, P. G. Popovich, and B. T. Stokes. "Traumatic spinal cord injury produced by controlled contusion in mouse." In: *Journal of Neurotrauma* 17.4 (Apr. 2000), Neurotrauma Soc.

[108] A. D. Kloos, L. C. Fisher, M. R. Detloff, D. L. Hassenzahl, and D. M. Basso. "Stepwise motor and all-or-none sensory recovery is associated with nonlinear sparing after incremental spinal cord injury in rats." In: *Experimental Neurology* 191.2 (Feb. 2005), pp. 251–265.

[109] J. J. Monaghan. "An introduction to SPH." In: *Computer Physics Communications* 48.1 (1988), pp. 89–96.

[110] J. A. Galbraith, L. E. Thibault, and D. R. Matteson. "Mechanical and electrical responses of the squid giant axon to simple elongation." In: *Journal of biomechanical engineering* 115.1 (1993). Research Support, U.S. Gov't, Non-P.H.S.,Research Support, U.S. Gov't, P.H.S., pp. 13–22.

[111] R. Shi and A. R. Blight. "Compression injury of mammalian spinal cord in vitro and the dynamics of action potential conduction failure." In: *Journal of Neurophysiology* 76.3 (Sept. 1996), pp. 1572–1580.

[112] P. K. Stys. "White matter injury mechanisms." In: *Current Molecular Medicine* 4.2 (Mar. 2004), pp. 113–130.

[113] P. G. Sullivan, A. G. Rabchevsky, P. C. Waldmeier, and J. E. Springer. "Mitochondrial permeability transition in CNS trauma: Cause or effect of neuronal cell death?" In: *Journal of Neuroscience Research* 79.1-2 (Jan. 2005), pp. 231–239.

[114] K. K. Mendis, R. L. Stalnaker, and S. H. Advani. "A Constitutive Relationship For Large-deformation Finite-element Modeling of Brain-tissue." In: *Journal of Biomechanical Engineering-transactions of the Asme* 117.3 (Aug. 1995), pp. 279–285.

[115] N. Yoganandan, S. Kumaresan, and F. A. Pintar. "Geometric and mechanical properties of human cervical spine ligaments." In: *Journal of Biomechanical Engineering-transactions of the Asme* 122.6 (Dec. 2000), pp. 623–629.

[116] K. E. Lee, A. N. Franklin, M. B. Davis, and B. A. Winkelstein. "Tensile cervical facet capsule ligament mechanics: Failure and subfailure responses in the rat." In: *Journal of Biomechanics* 39.7 (2006), pp. 1256–1264.

[117] K. P. Quinn and B. A. Winkelstein. "Cervical facet capsular ligament yield defines the threshold for injury and persistent joint-mediated neck pain." In: *Journal of Biomechanics* 40.10 (2007), pp. 2299–2306.

[118] R. K. Wilcox, D. J. Allen, R. M. Hall, D. Limb, D. C. Barton, and R. A. Dickson. "A dynamic investigation of the burst fracture process using a combined experimental and finite element approach." In: *European Spine Journal* 13.6 (2004). Times Cited: 33, pp. 481–488.

[119] R. J. Fiford and L. E. Bilston. "The mechanical properties of rat spinal cord in vitro." In: *Journal of Biomechanics* 38.7 (2005). Times Cited: 31, pp. 1509–1515.

[120] C. J. Sparrey, G. T. Manley, and T. M. Keaveny. "Effects of White, Grey, and Pia Mater Properties on Tissue Level Stresses and Strains in the Compressed Spinal Cord." In: *Journal of Neurotrauma* 26.4 (2009). Times Cited: 8, pp. 585–595.

[121] S. Chen and G. D. Doolen. "Lattice Boltzmann method for fluid flows." In: *Annual Review of Fluid Mechanics* 30 (1998). Times Cited: 1911, pp. 329–364.

[122] I. Ionescu, J. E. Guilkey, M. Berzins, R. M. Kirby, and J. A. Weiss. "Simulation of soft tissue failure using the material point method." In: *Journal of Biomechanical Engineering-transactions of the Asme* 128.6 (Dec. 2006), pp. 917–924.

[123] A. J. Wagner. "A Practical Introduction to the Lattice Boltzmann Method." In: *North Dakota State University* 463.March (2008), p. 663.

[124] J. E. Guilkey and J. A. Weiss. "Implicit time integration for the material point method: Quantitative and algorithmic comparisons with the finite element method." In: *International Journal For Numerical Methods In Engineering* 57.9 (July 2003), pp. 1323–1338.

[125] J. E. Guilkey, J. B. Hoying, and J. A. Weiss. "Computational modeling of multicellular constructs with the material point method." In: *Journal of Biomechanics* 39.11 (2006), pp. 2074–2086.

[126] I. Ionescu, J. Guilkey, M. Berzins, R. M. Kirby, and J. Weiss. "Computational Simulation of Penetrating Trauma in Biological Soft Tissues using the Material Point Method." In: *Medicine Meets Virtual Reality 13: the Magical Next Becomes the Medical Now* 111 (2005), pp. 213–218.

[127] F. Harlow. "The Particle-in-Cell Computing Method for Fluid Dynamics in Fundamental Methods in Hydrodynamics." In: *Experimental Arithmetic, High-Speed Computations and Mathematics* Experimental Arithmetic, High-Speed Computations and Mathematics (1964). Edited by B. Alder, S. Fernbach and M. Rotenberg, Academic Press, pp. 319-345., pp. 319–345.

[128] D. Sulsky, Z. Chen, and H. L. Schreyer. "A Particle Method For History-dependent Materials." In: *Computer Methods in Applied Mechanics and Engineering* 118.1-2 (1994). Times Cited: 191, pp. 179–196.

[129] D. Sulsky, S.-J. Zhou, and H. L. Schreyer. "Application of a particle-in-cell method to solid mechanics." In: *Computer Physics Communications* 87.1-2 (1995), pp. 236–252.

[130] Z. Chen and R. Brannon. "An evaluation of the material point method." In: *SAND Report, SAND2002-0482,(February 2002)* (Feb. 2002).

[131] J. Donea, S. Giuliani, and J. P. Halleux. "An arbitrary lagrangian-eulerian finite element method for transient dynamic fluid-structure interactions." In: *Computer Methods in Applied Mechanics and Engineering* 33.1-3 (1982), pp. 689–723.

[132] J. Donea, A. Huerta, J.-P. Ponthot, and A. Rodriguez-Ferran. "Arbitrary Lagrangian-Eulerian Methods." In: *Encyclopedia of Computational Mechanics*. John Wiley & Sons, Ltd, 2004. Chap. 14, pp. 413–437.

[133] G. R. Liu and M. Liu. *Smoothed particle hydrodynamics: a mesh-free particle method / G.R. Liu [and] M.B. Liu*. (Gui-Rong). New Jersey ; London : World Scientific, 2003.

[134] J. E. Guilkey, T. B. Harman, and B. Banerjee. "An Eulerian-Lagrangian approach for simulating explosions of energetic devices." In: *Computers & Structures* 85.11-14 (June 2007), pp. 660–674.

[135] S. Succi. *The lattice Boltzmann equation for fluid dynamics and beyond / Sauro Succi*. Oxford : Oxford University Press, 2001.

[136] . Wolf-Gladrow Dieter A. *Lattice-gas cellular automata and lattice Boltzmann models : an introduction / Dieter A. Wolf-Gladrow.* Includes bibliographical references. Berlin ; London : Springer, 2000.

[137] R. Begum and M. A. Basit. "Lattice Boltzmann Method and its Applications to Fluid Flow Problems." In: *European Journal of Scientific Research* 22.2 (2008), pp. 216–231.

[138] J. Summers. "To Mesh Or Not To Mesh: That Is The Question." In: *First given at iETSI Christmas Away Day,* Institute of Engineering Thermofluids, Surfaces and Interfaces, 2008.

[139] D. B. Chirila. *Introduction to Lattice Boltzmann Methods*. Alfred Wegener Institute. Feb. 2010.

[140] P. L. Bhatnagar, E. P. Gross, and M. Krook. "A Model For Collision Processes In Gases .1. Small Amplitude Processes In Charged and Neutral One-component Systems." In: *Physical Review* 94.3 (1954), pp. 511–525.

[141] X. Y. He and L. S. Luo. "Theory of the lattice Boltzmann method: From the Boltzmann equation to the lattice Boltzmann equation." In: *Physical Review E* 56.6 (Dec. 1997), pp. 6811–6817.

[142] R. A. Gingold and J. J. Monaghan. "Smoothed Particle Hydrodynamics - Theory And Application To Non-spherical Stars." In: *Monthly Notices of the Royal Astronomical Society* 181.2 (1977). Times Cited: 1410, pp. 375–389.

[143] L. B. Lucy. "Numerical Approach To Testing Of Fission Hypothesis." In: *Astronomical Journal* 82.12 (1977). Times Cited: 1554, pp. 1013–1024.

[144] W. G. Hoover. *Smooth particle applied mechanics : the state of the art*. (William Graham). Hackensack, N.J. ; London : World Scientific, 2006.

[145] T. Belytschko, Y. Krongauz, D. Organ, M. Fleming, and P. Krysl. "Meshless methods: An overview and recent developments." In: *Computer Methods In Applied Mechanics and Engineering* 139.1-4 (Dec. 1996), pp. 3–47.

[146] R. G. M. Breuls, B. G. Sengers, C. W. J. Oomens, C. V. C. Bouten, and F. P. T. Baaijens. "Predicting local cell deformations in engineered tissue constructs: A multilevel finite element approach." In: *Journal of Biomechanical Engineering-transactions of the Asme* 124.2 (Apr. 2002), pp. 198–207.

[147] J. D.d. S. Germain, J. McCorquodale, S. G. Parker, and C. R. Johnson. "Uintah: A Massively Parallel Problem Solving Environment." In: *Proceedings of the 9th IEEE International Symposium on High Performance Distributed Computing*. HPDC '00. Washington, DC, USA: IEEE Computer Society, 2000, pp. 33–41.

[148] A. F. Bower. *Applied mechanics of solids*. CRC press, 2011.

[149] R. Ogden. "Large deformation isotropic elasticity-on the correlation of theory and experiment for incompressible rubberlike solids." In: *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences* 326.1567 (1972), pp. 565–584.

[150] ADINA. *ADINA Theory and Modeling Guide Volume I: ADINA Solids and Structures*. ADINA. ADINA R & D, Inc., 71 Elton Avenue, Watertown, MA 02472 USA, Feb. 2008.

[151] J. H. Smith and J. J. Garcia. "Constitutive Modeling of Brain Tissue using Ogden-type Strain Energy Functions." In: *Poromechanics V@ sProceedings of the Fifth Biot Conference on Poromechanics*. ASCE. 2013, pp. 2140–2147.

[152] S. A. Maas, B. J. Ellis, G. A. Ateshian, and J. A. Weiss. "FEBio: Finite Elements for Biomechanics." In: *Journal of Biomechanical Engineering-transactions of the Asme* 134.1 (Jan. 2012), p. 011005.

[153] M. Berzins. "Status of release of the Uintah computational framework." In: *Scientific Computing and Imaging Institute, Tech. Rep. UUSCI-2012-001* (2012).

[154] M. Berzins, J. Schmidt, Q. Meng, and A. Humphrey. "Past, Present and Future Scalability of the Uintah Software." In: *Blue Waters Workshop, Chicago, IL, USA*. 2012.

[155] P. C. Wallstedt and J. E. Guilkey. "An evaluation of explicit time integration schemes for use with the generalized interpolation material point method." In: *Journal of Computational Physics* 227.22 (Nov. 2008), pp. 9628–9642.

[156] J. Guilkey, T. Harman, J. Luitjens, J. Schmidt, J. Thornock, S. Davison de St. Germain J. andShankar, J. Peterson, C. Brownlee, and C. Reid. *Uintah User Guide (version 1.6)*. Scientific Computing and Imaging Institute, University of Utah. Nov. 2012.

[157] S. G. Bardenhagen and E. M. Kober. "The generalized interpolation material point method." In: *Cmes-computer Modeling In Engineering & Sciences* 5.6 (June 2004), pp. 477–495.

[158] R. K. Wilcox. "A Biomechanical Study of Thoracolumbar Burst Fractures." PhD thesis. University of Leeds, 2002.

[159] R. W. Ogden, G. Saccomandi, and I. Sgura. "Fitting hyperelastic models to experimental data." In: *Computational Mechanics* 34.6 (Nov. 2004), pp. 484–502.

[160] S. Bardenhagen, J. Guilkey, K. Roessig, J. Brackbill, W. Witzel, and J. Foster. "An improved contact algorithm for the material point method and application to stress propagation in granular material." English. In: *CMES-COMPUTER MODELING IN ENGINEERING & SCIENCES* 2.4 (2001), pp. 509–522.

[161] B. Kashiwa. *A multifield model and method for fluid-structure interaction dynamics*. Tech. rep. Technical Report LA-UR-01-1136, Los Alamos National Laboratory, Los Alamos., 2001.

[162] F. H. Harlow and A. A. Amsden. "Flow of Interpenetrating Material Phases." In: *Journal of Computational Physics* 18.4 (1975), pp. 440–464.

[163] B. A. Kashiwa, N. T. Padial, R. M. Rauenzahn, and W. B. VanderHeyden. *A cell-centered ICE method for multiphase flow simulations*. Tech. rep. Los Alamos National Lab., NM (United States), 1993.

[164] J. S. Thomsen and T. J. Hartka. "Strange Carnot Cycles; Thermodynamics of a System with a Density Extremum." In: *American Journal of Physics* 30.1 (1962), pp. 26–33.

[165] P. Aimedieu and R. Grebe. "Tensile strength of cranial pia mater: preliminary results." In: *Journal of Neurosurgery* 100.1 (Jan. 2004), pp. 111–114.

[166] C. J. Sparrey and T. M. Keaveny. "Compression behavior of porcine spinal cord white matter." In: *Journal of Biomechanics* 44.6 (2011). Times Cited: 2, pp. 1078–1082.

[167] C. J. Lam, P. Assinck, J. Liu, W. Tetzlaff, and T. R. Oxland. "Impact Depth and the Interaction with Impact Speed Affect the Severity of Contusion Spinal Cord Injury in Rats." In: *Journal of Neurotrauma* 31.24 (Dec. 2014), pp. 1985–1997.

[168] J. H. T. Lee et al. "A Novel Porcine Model of Traumatic Thoracic Spinal Cord Injury." In: *Journal of Neurotrauma* 30.3 (Feb. 2013), pp. 142–159.

[169] E. Mazgajczyk, K. Scigala, M. Czyz, W. Jarmundowicz, and R. Bedzinski. "Mechanical properties of cervical dura mater." In: *Acta of Bioengineering and Biomechanics* 14.1 (2012), pp. 51–58.