Finite Element Modeling of Hyper Elastic Materials

Kenny Erleben
Department of Computer Science
University of Copenhagen
kenny@diku.dk

Marts 2013

Preface

The finite element method is often considered by master students in computer science to be a difficult topic to learn. Thus, we have embarked on writing yet an introduction to the subject through a series of technical reports [Erleben(2011a), Erleben(2011b), Erleben(2011d)]. This work is intended to be an introduction to our notation and terminology.

There are many good textbooks already written which contains an introduction to tensors and continuum mechanics. This text is no substitute for these works. Our text merely aim to give only the foundation needed for computer science master students to read our other technical reports without having to buy expensive textbooks.

There are many books on the subject of continuum mechanics [Chadwick(1999), Spencer(2004), Reddy(2008)] and [Lai et al.(2009)Lai, Rubin, and Krempl] as well as books about the finite element method [Zienkiewicz and Taylor(2000), Bonet and Wood(2000), Cook et al.(2007)Cook, Malkus, Plesha, and Witt]. These books provide in depth detail on tensor algebra and calculus as well as continuum mechanics.

For the students wanting to become specialist on physics based modeling and simulation we strongly encourage them to seek out textbooks like those listed above to supplement the knowledge we introduce in our series of technical reports on the finite element method.

Why have we written yet another introduction to finite element methods? For starters the textbooks mentioned above are written for students in engineering or physics. This leaves the computer scientist somewhat behind. Being computer scientists ourselves we believe that we can write an introduction for those students that will help them move on to more advanced texts from other fields.

Another reason is to give a coherent frame for the many topics we cover. Alone in regards to notation and conventions there are several different styles. In this text and our later texts [Erleben(2011a), Erleben(2011b), Erleben(2011d)] we try to put everything related to simulation of hyper elastic materials using a finite element method into one complete setting. As such this text defines our notation and conventions used later.

Our background originates in the field of computer graphics. This field has applied finite element methods for generating images and movies for many different application areas including games, movies and virtual prototyping among others. In our introduction to the subject we try to look reflectively on past work on computer graphics and explain in detail some of the assumptions, tricks and hacks that have been applied to obtain faster computations.

Contents

Chapter 1

Preliminaries on Tensor Notation and Continuum Mechanics

Having motivated our need for writing yet another text about the finite element method we can embark on introducing the fundamentals needed for reading our later chapters. In Section 1.1 we introduce tensors which is the mathematical tool to be used. Hereafter in Section 1.2 we look at the physics of continuous matter and derive the partial differential equations that describe its motion. Having a firm grasp of the physics we close this preliminary text with introducing the directional derivate in Section 1.3.

1.1 Introduction to Tensor Algebra and Calculus

We will start by introducing some concepts and definitions from tensor algebra and tensor calculus. Our aim is not to be mathematically rigorous nor extensive in our coverage. We simply seek to generalize enough concepts from matrix algebra and vector calculus to allow us to later write up equations needed to explain continuum mechanics. In the following we implicitly assume to be working a 3 dimensional space. However, the tensor algebra and calculus we derive can be stated in for any dimension. However, we only need to work with 3 dimensional spaces in our treatment of continuum mechanics hence we specialize immediately for this case.

The Cartesian basis vectors are denoted by $\vec{e}_1$, $\vec{e}_2$ and $\vec{e}_3$ and by definition we write a vector in tensor notation as

$$\vec{v} = \sum_{i=1}^{3} v_i \vec{e}_i. \tag{1.1}$$

Here v_i is called the component of the vector $\vec{v}$.

As only one index is needed to label the components we refer to this as a first order tensor. Obviously, a scalar would be a zero order tensor as no labeling (indexing) is needed. As we see later a second order tensor needs two indices to label the components and we will observe the similarity between a second order tensor and matrix.

n^{th} **order tensor:** A tensor is of n^{th} order if n indices are needed to label all its components. This gives us the relation to matrix algebra

- 0^{th} order tensor corresponds to a scalar
- 1^{th} order tensor corresponds to a vector
- 2^{th} order tensor corresponds to a matrix

Tensor algebra generalizes into higher order. In this book we will be working with 3^{th} and 4^{th} order tensors as well.

By definition the dot-product of the Cartesian basis vectors are defined as

$$\vec{e}_i \cdot \vec{e}_j = \delta_{ij} \tag{1.2}$$

where δ_{ij} is the Kronecker delta function defined as

$$\delta_{ij} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases} \tag{1.3}$$

From this definition we may now work out what the dot product of two vectors $\vec{v}$ and $\vec{u}$ is.

$$\vec{u} \cdot \vec{v} = \left(\sum_{i=1}^{3} u_i \vec{e}_i \right) \cdot \left(\sum_{j=1}^{3} v_j \vec{e}_j \right) \tag{1.4}$$

$$= \sum_{i=1}^{3} \sum_{j=1}^{3} u_i v_j \vec{e}_i \cdot \vec{e}_j \tag{1.5}$$

$$= \sum_{i=1}^{3} u_i v_i \tag{1.6}$$

The above strategy for deriving our tensor algebra "rule" is general and we will be using the principle of inserting the "defintion" of the corresponding n^{th} or tensor into the rules we wish to prove in order to derive the rules from principle of construction.

We can find the components of a vector by taking the dot product with a basis vector

$$v_i = \vec{v} \cdot e_i = \left(\sum_{j=1}^{3} v_j \vec{e}_j \right) \cdot e_i = \sum_{j=1}^{3} v_j \vec{e}_j \cdot \vec{e}_i = v_i \tag{1.7}$$

Using matrix notation rather than tensor notation we write the components as a column vector

$$\left[\vec{v}\right] = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \tag{1.8}$$

Observe that we use bracketing to indicate that we have changed into a matrix notation. It can be rather confusing when deriving formulas if one does not keep a strict distinction between working in matrix notation or tensor notation. The tensor notation has the benefit that it is independent of the choice of coordinates and generalizes easily and simpler to higher dimensions. Hence it is a "stronger" tool for working out the theory of continuum mechanics and discretizations. However, as soon as in an implementation we choose a coordinate basis to work with everything becomes vectors and matrices as seen from a computer scientists viewpoint.

An important observation is that In tensor notation there is no notion of " column" versus "row" vector. Here a vector is simply just a vector. A vector remains unchanged if one uses a different basis. That is

$$\vec{v} = \sum_{j=1}^{3} v_j \vec{e}_j = \sum_{j=1}^{3} v_j' \vec{E}_j \tag{1.9}$$

where $\vec{E}_i$ denotes the basis vectors of some other set of basis vectors (also obeying the Kronecker delta requirement). In matrix notation the components would be written as

$$\left[\vec{v}\right]_e = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \neq \begin{bmatrix} v_1' \\ v_2' \\ v_3' \end{bmatrix} = \left[\vec{v}\right]_E \tag{1.10}$$

Here the subscript denotes the chosen basis used to express the component values with respect to. If no subscript is given then the default Cartesian basis is assumed. Our default convention for the Cartesian basis vectors is

$$\left[\vec{e}_1\right] = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \left[\vec{e}_2\right] = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \text{and } \left[\vec{e}_3\right] = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{1.11}$$

The dot product of vectors is commutative, that is we can write

$$\vec{u} \cdot \vec{v} = \vec{v} \cdot \vec{u} \tag{1.12}$$

which follows from the definition of the dot product. Scalar multiplication for a vector is defined as

$$a\vec{v} = \vec{v}a = a\left(\sum_{j=1}^{3} v_j \vec{e}_j\right) = \sum_{j=1}^{3} (a v_j) \vec{e}_j \tag{1.13}$$

Let $\vec{u}$, $\vec{v}$ and $\vec{w}$ be vectors then the tensor (or dyadic) product is defined as

$$(\vec{u} \otimes \vec{v})\,\vec{w} = (\vec{v} \cdot \vec{w})\,\vec{u} \tag{1.14}$$

A second order tensor is defined as a linear mapping that maps a vector (a first order tensor) to a vector. That is

$$\vec{v} = \mathbf{A}\vec{u} \tag{1.15}$$

We express a second order tensor as a linear combination of tensor products as

$$\mathbf{A} = \sum_{i=1}^{3}\sum_{j=1}^{3} A_{ij}\,(\vec{e}_i \otimes \vec{e}_j) \tag{1.16}$$

where A_{ij} are termed the components of the tensor. With this expression we may have a second look at the definition of a second order tensor as a linear mapping of vectors

$$\begin{aligned}
\vec{v} &= \mathbf{A}\vec{u} && (1.17)\\
&= \left(\sum_{i=1}^{3}\sum_{j=1}^{3} A_{ij}\vec{e}_i \otimes \vec{e}_j\right)\left(\sum_{k=1}^{3} u_k\vec{e}_k\right) && (1.18)\\
&= \sum_{i=1}^{3}\sum_{j=1}^{3}\sum_{k=1}^{3} A_{ij}u_k\,(\vec{e}_i \otimes \vec{e}_j)\,\vec{e}_k && (1.19)\\
&= \sum_{i=1}^{3}\sum_{j=1}^{3}\sum_{k=1}^{3} A_{ij}u_k\,(\vec{e}_j \cdot \vec{e}_k)\,\vec{e}_i && (1.20)\\
&= \sum_{i=1}^{3}\sum_{j=1}^{3} A_{ij}u_j\vec{e}_i && (1.21)
\end{aligned}$$

We observe the similarity between the defintion of the second order tensor and with the matrix-vector product. With the tensor product expression of the second order tensor we may define what is meant by scalar multiplication of tensors as

$$a\mathbf{A} = \mathbf{A}a = \sum_{i=1}^{3}\sum_{j=1}^{3} aA_{ij}\,(\vec{e}_i \otimes \vec{e}_j) \tag{1.22}$$

where a is some scalar. Using the above expression gives us the familiar formula

$$A_{ij} = \vec{e}_i \cdot \mathbf{A}\vec{e}_j \tag{1.23}$$

This follows from substitution

$$\vec{e}_i \cdot \mathbf{A}\vec{e}_j = \vec{e}_i \cdot \left(\sum_{k=1}^{3}\sum_{l=1}^{3} A_{kl}\vec{e}_k \otimes \vec{e}_l \right) \vec{e}_j \tag{1.24}$$

$$= \vec{e}_i \cdot \left(\sum_{k=1}^{3}\sum_{l=1}^{3} A_{kl}\left(\vec{e}_j \cdot \vec{e}_l\right)\vec{e}_k \right) \tag{1.25}$$

$$= \vec{e}_i \cdot \left(\sum_{k=1}^{3}\sum_{l=1}^{3} A_{kl}\delta_{jl}\vec{e}_k \right) \tag{1.26}$$

$$= \sum_{k=1}^{3}\sum_{l=1}^{3} A_{kl}\delta_{jl}\delta_{ik} \tag{1.27}$$

$$= A_{ij} \tag{1.28}$$

A special tensor is the identity tensor

$$\mathbf{I} = \sum_{i=1}^{3} \vec{e}_i \otimes \vec{e}_i = \sum_{i=1}^{3}\sum_{j=1}^{3} \delta_{ij}\vec{e}_i \otimes \vec{e}_j \tag{1.29}$$

which maps any vector to it self

$$\vec{u} = \mathbf{I}\vec{u} \tag{1.30}$$

as can be verified by substitution

$$\mathbf{I}\vec{u} = \left(\sum_{i=1}^{3} \vec{e}_i \otimes \vec{e}_i \right) \vec{u} = \sum_{i=1}^{3} \left(\vec{e}_i \cdot \vec{u}\right)\vec{e}_i = \sum_{i=1}^{3} u_i\vec{e}_i = \vec{u} \tag{1.31}$$

As with vectors we may write the components in matrix notation

$$\left[\mathbf{A}\right] = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} \tag{1.32}$$

As before we use subscripts to indicate the basis vectors used

$$\left[\mathbf{A}\right]_e = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix}_e \neq \begin{bmatrix} A'_{11} & A'_{12} & A'_{13} \\ A'_{21} & A'_{22} & A'_{23} \\ A'_{31} & A'_{32} & A'_{33} \end{bmatrix}_E = \left[\mathbf{A}\right]_E \tag{1.33}$$

However, from a tensor viewpoint we have

$$\mathbf{A} = \sum_{i=1}^{3}\sum_{j=1}^{3} A_{ij}\vec{e}_i \otimes \vec{e}_j = \sum_{i=1}^{3}\sum_{j=1}^{3} A'_{ij}\vec{E}_i \otimes \vec{E}_j \tag{1.34}$$

The fact that a second order tensor is a linear mapping means that given two second order tensors $\mathbf{A}$ and $\mathbf{B}$ then the linear combination of these

$$\mathbf{C} = a\mathbf{A} + b\mathbf{B} \tag{1.35}$$

is a second order tensor. That is one can show

$$C_{ij} = aA_{ij} + bB_{ij} \tag{1.36}$$

which we leave as an exercise of the reader.

Many of the functions and operations known from matrix algebra have counter parts in tensor algebra. For instance the trace of a tensor is defined as the sum of the diagonal components of the tensor

$$\text{tr}\left(\mathbf{A}\right) = \sum_{i=1}^{3} A_{ii} \tag{1.37}$$

The determinant is defined as the determinant of the matrix counter part

$$\det\left(\mathbf{A}\right) = \det\left(\left[\mathbf{A}\right]\right) \tag{1.38}$$

The transpose is defined operationally through

$$\vec{u} \cdot \mathbf{A}\vec{v} = \vec{v} \cdot \mathbf{A}^T\vec{u} \tag{1.39}$$

where $\mathbf{A}^T$ is the transpose tensor of $\mathbf{A}$. If $\mathbf{A}^T = \mathbf{A}$ then we say $\mathbf{A}$ is a symmetric tensor and if $\mathbf{A} = -\mathbf{A}^T$ then $\mathbf{A}$ is said to be a skew tensor (or skew symmetric). The inverse tensor is defined as the tensor $\mathbf{A}^{-1}$ where

$$\mathbf{A}^{-1}\mathbf{A} = \mathbf{I} \tag{1.40}$$

Summation and multiplication of tensors are defined accordingly

$$\left(\mathbf{A} + \mathbf{B}\right)\vec{u} = \mathbf{A}\vec{u} + \mathbf{B}\vec{u} \tag{1.41}$$

$$\mathbf{A}\mathbf{B}\vec{u} = \mathbf{A}\left(\mathbf{B}\vec{u}\right) \tag{1.42}$$

As an example let $\mathbf{C} = \mathbf{A}\mathbf{B}$ then in terms of components we have

$$\begin{aligned}
\mathbf{C}\vec{u} &= \sum_{i=1}^{3}\sum_{k=1}^{3} A_{ik}\left(\vec{e}_i \otimes \vec{e}_k\right)\left(\sum_{l=1}^{3}\sum_{j=1}^{3} B_{lj}\left(\vec{e}_l \otimes \vec{e}_j\right)\vec{u}\right) && (1.43)\\
&= \sum_{i=1}^{3}\sum_{k=1}^{3} A_{ik}\left(\vec{e}_i \otimes \vec{e}_k\right)\left(\sum_{l=1}^{3}\sum_{j=1}^{3} B_{lj}u_j\vec{e}_l\right) && (1.44)\\
&= \sum_{i=1}^{3}\sum_{k=1}^{3}\sum_{l=1}^{3}\sum_{j=1}^{3} \delta_{kl}A_{ik}B_{lj}u_j\vec{e}_i && (1.45)\\
&= \sum_{i=1}^{3}\sum_{j=1}^{3}\left(\sum_{k=1}^{3} A_{ik}B_{kj}\right)\left(\vec{e}_i \otimes \vec{e}_j\right)\vec{u} && (1.46)
\end{aligned}$$

Hence we have found that for the components of $\mathbf{C}$ we have

$$C_{ij} = \sum_{k=1}^{3} A_{ik} B_{kj} \tag{1.47}$$

As expected from our familiarity with matrix multiplication. An often used operation on tensors is the double contraction (or double product) which is defined as

$$\mathbf{A} : \mathbf{B} = \sum_{i=1}^{3} \sum_{j=1}^{3} A_{ij} B_{ij} \tag{1.48}$$

This can be expressed differently as

$$\mathbf{A} : \mathbf{B} = \text{tr}\left(\mathbf{A}^T \mathbf{B}\right) \tag{1.49}$$

We ask the reader to verify this by now defining $\mathbf{D} = \mathbf{A}^T \mathbf{B}$ and then show that the components of $\mathbf{D}$ is

$$D_{ij} = \sum_{k=1}^{3} A_{ki} B_{kj} \tag{1.50}$$

Lastly using $\text{tr}\left(\mathbf{D}\right) = \sum_{i=1}^{3} D_{ii}$ the reader can verify $\mathbf{A} : \mathbf{B} = \text{tr}\left(\mathbf{A}^T \mathbf{B}\right)$ is true.

Kenny Says: Maybe add some definition of cross product too?

Let a be a zero order tensor then we define ∇a to mean

$$\nabla a \equiv \sum_i \vec{e}_i \partial_i a$$

We call this the gradient operator of a or simply the gradient of a. The ∇ notation may also be defined when applied to a first order tensor $\vec{b}$ to mean

$$\left(\nabla \vec{b}\right)_{ij} = \partial_j b_i$$

Our definition is precisely the partial derivative of the vector function $\vec{b}$ or what is also termed the Jacobian of $\vec{b}$. Observe that when ∇ is applied to a first order tensor the result is a second order tesnor. Another useful differential operation is the divergence operator of a vector which we define as

$$\nabla \cdot \vec{b} \equiv \sum_i \partial_i b_i$$

Notice this maps a first order tensor into a zero order tensor as we expect. The divergence of a second order tensor may also be defined using ∇-notation. Let $\mathbf{A}$ be a second order tensor then we define the divergence operator of $\mathbf{A}$ as

$$\nabla \cdot \mathbf{A} \equiv \sum_j \sum_i \partial_i A_{ij} \vec{e}_j \tag{1.51}$$

where $\vec{e}_j$ is the j^{th} unit vector. In terms of matrix notation this corresponds to

$$\underbrace{\begin{bmatrix}\partial_1\partial_2\partial_3\end{bmatrix}}_{\nabla^T}\begin{bmatrix}A_{11} & A_{12} & A_{13}\\ A_{21} & A_{22} & A_{23}\\ A_{31} & A_{32} & A_{33}\end{bmatrix} = \\ \left[\left(\sum_i \partial_i A_{i1}\right)\quad\left(\sum_i \partial_i A_{i2}\right)\quad\left(\sum_i \partial_i A_{i3}\right)\right] \tag{1.52}$$

Each component is the divergence of the corresponding column vector. Hence we make think conceptually as the divergence operator being applied column wise to the "matrix".

Kenny Says: Above definitions may seem arbitrary, maybe add some common thread/theory to help students overcome the abstraction?

The directional derivative of a tensor $\mathbf{T}$ in the direction of the constant vector $\vec{c}$ is defined as

$$D(\mathbf{T};\vec{c}) \equiv \lim_{\alpha\to 0}\frac{d}{d\alpha}\mathbf{T}\left(\vec{x}+\alpha\,\vec{c}\right)$$

By definition we have that $\vec{x}\equiv\sum_j x_j\vec{e}_j$ and $\vec{c}\equiv\sum_j c_j\vec{e}_j$. Now we let $\vec{y}\equiv\vec{x}+\alpha\,\vec{c}$ then we write the directional derivative as

$$\begin{aligned}
D(\mathbf{T};\vec{c}) &\equiv \lim_{\alpha\to 0}\frac{d}{d\alpha}\mathbf{T}\left(\vec{x}+\alpha\,\vec{c}\right)\\
&= \frac{d}{d\alpha}\mathbf{T}\left(x_1+\alpha\,c_1, x_2+\alpha\,c_2, x_3+\alpha\,c_3\right)\Big|_{\alpha=0}\\
&= \left[\frac{\partial\mathbf{T}}{\partial y_1}\frac{\partial y_1}{\partial\alpha}\quad\frac{\partial\mathbf{T}}{\partial y_2}\frac{\partial y_2}{\partial\alpha}\quad\frac{\partial\mathbf{T}}{\partial y_3}\frac{\partial y_3}{\partial\alpha}\right]_{\alpha=0}\\
&= \left[\frac{\partial\mathbf{T}}{\partial y_1}c_1\quad\frac{\partial\mathbf{T}}{\partial y_2}c_2\quad\frac{\partial\mathbf{T}}{\partial y_3}c_3\right]_{\alpha=0}\\
&= \left[\frac{\partial\mathbf{T}}{\partial x_1}c_1\quad\frac{\partial\mathbf{T}}{\partial x_2}c_2\quad\frac{\partial\mathbf{T}}{\partial x_3}c_3\right]_{\alpha=0}\\
&= \sum_j\frac{\partial\mathbf{T}}{\partial x_j}\left(\vec{e}_j\cdot\vec{c}\right)\\
&= \sum_j\left(\frac{\partial\mathbf{T}}{\partial x_j}\otimes\vec{e}_j\right)\vec{c}
\end{aligned}$$

The last rewrite is a little bit of a "cheat" and requires some explanation. The issue is the specific case for when $\mathbf{T}$ is a zero order tensor as the tensor product definition would not make much sense between a "scalar" and a "vector". However, one may rewrite the zero tensor as $a\mathbf{I}$ $=\equiv \sum_i a\left(\vec{e}_i\otimes\vec{e}_i\right)$ or simply consider a mindless usage of our original defintion, $\left(a\otimes\vec{v}\right)\vec{w} = \left(\vec{v}\cdot\vec{w}\right)a$, gives exactly the meaning we desire for this special case. The last rewrite has its immediate obvious generalization benefits for all higher order tensors. Now using the definitions of a zero order tensor $\mathbf{T}\equiv a$ gives $D(a;\vec{c}) = \left(\sum_i\vec{e}_i\partial_i a\right)\cdot\vec{c}$ and using first order tensor $\mathbf{T}\equiv\vec{b}=\sum_i\vec{e}_i b_i$ gives $D(\vec{b};\vec{c}) = \left(\nabla\vec{b}\right)\vec{c}$. Divergence of a vector may be defined as $\nabla\cdot\vec{b} = \operatorname{tr}\left(\nabla\vec{b}\right)$ from this we see our previous definitions are in agreement with this definition. A recursive approach can be use to derive the divergence of a second order tensor by computing the divergence of $\nabla \cdot (\mathbf{A}\vec{c})$ where $\mathbf{A}$ is second order tensor and $\vec{c}$ is a constant, we observe we can use our first definition of divergence of first order tensor to find a formula for divergence of second order tensor. We discover that $\nabla \cdot (\mathbf{A}\vec{c}) = (\nabla \cdot \mathbf{A}) \cdot \vec{c}$.

The product rule for the divergence operator. Let $\mathbf{A}$ be a second order tensor and $\vec{b}$ a vector and $\nabla \equiv \sum_i \vec{e}_i \partial_i$ Then

$$\nabla \cdot (\mathbf{A}\vec{b}) = (\nabla \cdot \mathbf{A}) \cdot \vec{b} + \mathbf{A} : (\nabla \vec{b})^T \tag{1.53}$$

where

$$\nabla \vec{b} = \begin{bmatrix} \partial_1 \vec{b}_1 & \partial_2 \vec{b}_1 & \partial_3 \vec{b}_1 \\ \partial_1 \vec{b}_2 & \partial_2 \vec{b}_2 & \partial_3 \vec{b}_2 \\ \partial_1 \vec{b}_3 & \partial_2 \vec{b}_3 & \partial_3 \vec{b}_3 \end{bmatrix}. \tag{1.54}$$

The double contraction between two second order tensors $\mathbf{A}, \mathbf{B}$ is defined as

$$\mathbf{A} : \mathbf{B} = \sum_i \sum_j A_{ij} B_{ij} \tag{1.55}$$

Exercise Prove the product rule formula in (1.53).

Answer From definition of divergence of vector

$$\nabla \cdot (\mathbf{A}\vec{b}) = \sum_i \partial_i \sum_j (A_{ij} b_j) \tag{1.56a}$$

Apply product rule

$$\nabla \cdot (\mathbf{A}\vec{b}) = \sum_i \sum_j \left(\left(\partial_i A_{ij} \right) b_j + A_{ij} \left(\partial_i b_j \right) \right) \tag{1.57}$$

Isolate $\vec{b}_j$-terms and use double contraction

$$\nabla \cdot (\mathbf{A}\vec{b}) = \left(\sum_j \sum_i \partial_i A_{ij} \vec{e}_j \right) \cdot \vec{b} + \mathbf{A} : (\nabla \vec{b})^T \tag{1.58}$$

This completes the proof.

The symmetric product rule. If $\mathbf{A}$ is a symmetric tensor then $\mathbf{A}^T = \mathbf{A}$. Assuming a symmetric $\mathbf{A}$ then the product rule from before can be rewritten as

$$\nabla \cdot (\mathbf{A}\vec{b}) = (\nabla \cdot \mathbf{A}) \cdot \vec{b} + \mathbf{A} : (\nabla \vec{b}) \tag{1.59}$$

Exercise Prove the symmetric form of the vector divergence formula (1.59).

Answer Let $\mathbf{A}$ and $\mathbf{B}$ be two second order tensors. From definition of double contraction

$$\mathbf{A} : \mathbf{B}^T = \sum_i \sum_j \mathbf{A}_{ij}\mathbf{B}_{ji} \tag{1.60a}$$

$$= \sum_i \sum_j \mathbf{A}_{ji}\mathbf{B}_{ij} \tag{1.60b}$$

$$= \mathbf{A}^T : \mathbf{B} \tag{1.60c}$$

Now if $\mathbf{A}$ is symmetric we have $\mathbf{A} : \mathbf{B}^T = \mathbf{A} : \mathbf{B}$. Applying this rule to the product rule completes our proof.

Change of variables. Let $\vec{x}$ and $\vec{X}$ be three dimensional vectors and let $\vec{x} = \vec{\Phi}(\vec{X})$ where the domain is given by $\vec{X} \in V$ and the range by $\vec{x} \in v$ then change of variables is given by the relation

$$\int_v f(\vec{x})dv = \int_V f(\vec{\Phi}(\vec{X})) \left| \det \left(\frac{\partial \vec{\Phi}}{\partial \vec{X}} \right) \right| dV \tag{1.61}$$

where the differential volume is given by $dV = dX_1 dX_2 dX_3$ and similar definition apply for dv.

Usually we use the shorthand notation $\mathbf{F} = \frac{\partial \vec{\Phi}}{\partial \vec{X}}$ and $j = |\det(\mathbf{F})|$ and then we have the differential relation

$$dv = jdV \tag{1.62}$$

In a two dimensional space we have correspondingly the differential relation $da = jdA$.

Later we will show that the function $\vec{\Phi}$ is the deformation function transforming some undeformed (material) coordinates $\vec{X}$ into some deformed (spatial) coordinates $\vec{x}$. Hence the above relation becomes important for us when we wish to algebraically manipulate integrals from spatial deformed space into their counterparts in material undeformed space or vice versa.

Exercise Derive formula (5.46)for 2D and 3D case.

Answer In 2D we have that the undeformed differential area is given by $dA = dX_1 dX_2$ where dX_1 and dX_2 are the differentials along the basis vectors. We may define the corresponding differential vectors as

$$d\vec{X}_1 = \vec{e}_1 dX_1 \tag{1.63a}$$

$$d\vec{X}_2 = \vec{e}_2 dX_2 \tag{1.63b}$$

From the definition of the differential we have that an undeformed material vector $d\vec{M}$ maps into its corresponding spatial deformed vector $d\vec{m}$ as

$$d\vec{m} = \mathbf{F} d\vec{M} \tag{1.64}$$

. Hence, we can map the differential vectors from undeformed space into deformed space as follows

$$d\vec{x}_i = \mathbf{F} d\vec{X}_i \quad \text{for } i = 1, 2 \tag{1.65}$$

We may now write the corresponding differential area in the spatial space as

$$da = |d\vec{x}_1 \times d\vec{x}_2| \tag{1.66a}$$
$$= |\mathbf{F}\vec{e}_1 dX_1 \times \mathbf{F}\vec{e}_2 dX_2| \tag{1.66b}$$
$$= |\mathbf{F}\vec{e}_1 \times \mathbf{F}\vec{e}_2| \, dX_1 dX_2 \tag{1.66c}$$
$$= |\text{det}(\mathbf{F})| \, dX_1 dX_2 \tag{1.66d}$$
$$= j dA \tag{1.66e}$$

where we recognized $\mathbf{F}\vec{e}_1 \times \mathbf{F}\vec{e}_2$ to be the cross product of the first and second column of the corresponding matrix of $\mathbf{F}$ and hence in 2D the same as the determinant of that 2-by-2 matrix.

In 3D $d\vec{X}_3 = \vec{e}_3 dX_3$ and $dV = dX_1 dX_2 dX_3$ and the spatial differential volume is given by the triple space product $dv = \left| d\vec{X}_3 \cdot \left(d\vec{X}_1 \times d\vec{X}_2 \right) \right|$. The proof in 3D follows by observing that $\mathbf{F}\vec{e}_3 \cdot (\mathbf{F}\vec{e}_1 \times \mathbf{F}\vec{e}_2) = \text{det}(\mathbf{F})$

Nanson's Relation. Let the volume elements in spatial (lower case) and material (upper case) coordinates be written as

$$dv = d\vec{x} \cdot d\vec{s} \tag{1.67a}$$
$$dV = d\vec{X} \cdot d\vec{S} \tag{1.67b}$$

where $d\vec{x} = \frac{\partial \Phi}{\partial \vec{X}} d\vec{X}$ and $d\vec{s} = ds\vec{n}$ and $d\vec{S} = dS\vec{N}$. From change of variables and using $\mathbf{F} \equiv \frac{\partial \Phi}{\partial \vec{X}}$

$$dv = j dV \tag{1.68a}$$
$$d\vec{x} \cdot d\vec{s} = j d\vec{X} \cdot d\vec{S} \tag{1.68b}$$
$$d\vec{X} \cdot \mathbf{F}^T ds\vec{n} = d\vec{X} \cdot j dS\vec{N} \tag{1.68c}$$

We have Nanson's Relation

$$ds\vec{n} = j\mathbf{F}^{-T} dS\vec{N} \tag{1.69}$$

The Contraction and trace relation,

$$\text{tr}\left(\mathbf{A}\mathbf{B}^T\right) = \text{tr}\left(\mathbf{A}^T\mathbf{B}\right) = \mathbf{A} : \mathbf{B} \tag{1.70}$$

Proof, by definition

$$\text{tr}\left(\mathbf{A}^T\mathbf{B}\right) = \sum_j \left(\mathbf{A}^T\mathbf{B}\right)_{jj} \tag{1.71a}$$
$$= \sum_j \left(\sum_i \mathbf{A}^T_{ji} \mathbf{B}_{ij} \right) \tag{1.71b}$$
$$= \sum_j \sum_i \mathbf{A}_{ij} \mathbf{B}_{ij} \tag{1.71c}$$

The proof follows by comparison with $\mathbf{A} : \mathbf{B} = \sum_j \sum_i \mathbf{A}_{ij}\mathbf{B}_{ij}$ Trace relations. It can be shown for second order tensors $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$, that

$$a\text{tr}\left(\mathbf{A}\right) = \text{tr}\left(a\mathbf{A}\right) \tag{1.72}$$

and

$$\text{tr}\left(\mathbf{AB}\right) = \text{tr}\left(\mathbf{BA}\right) \tag{1.73}$$

also

$$\text{tr}\left(\mathbf{ABC}\right) = \text{tr}\left(\mathbf{CAB}\right) \tag{1.74}$$

The first two "rules" follow straightforward from definitions. The third rule comes from applying the second rule, $\text{tr}\left(\left(\mathbf{AB}\right)\mathbf{C}\right) = \text{tr}\left(\mathbf{C}\left(\mathbf{AB}\right)\right)$. Derivatives with respect to tensors. Let $f(\mathbf{A})$ be a scalar function of the tensor $\mathbf{A}$ then by definition we have

$$\left[\frac{\partial f(\mathbf{A})}{\partial \mathbf{A}}\right]_{ij} = \frac{\partial f(\mathbf{A})}{\partial \mathbf{A}_{ij}} \tag{1.75}$$

Differentiation rules. Let $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{3\times 3}$

$$\frac{\partial\left(\mathbf{A} : \mathbf{B}\right)}{\partial \mathbf{A}} = \mathbf{B} \tag{1.76a}$$

$$\frac{\partial\left(\mathbf{A} : \mathbf{B}\right)}{\partial \mathbf{B}} = \mathbf{A} \tag{1.76b}$$

$$\frac{\partial\left(\mathbf{A} : \mathbf{A}\right)}{\partial \mathbf{A}} = 2\mathbf{A} \tag{1.76c}$$

$$\frac{\partial \text{tr}\left(\mathbf{A}\right)}{\partial \mathbf{A}} = \mathbf{I} \tag{1.76d}$$

$$\frac{\partial \text{tr}\left(\mathbf{A}^2\right)}{\partial \mathbf{A}} = 2\mathbf{A} \tag{1.76e}$$

$$\frac{\partial \text{tr}\left(\mathbf{A}\right)^2}{\partial \mathbf{A}} = 2\text{tr}\left(\mathbf{A}\right)\mathbf{I} \tag{1.76f}$$

Exercise Prove the differentiation rules above in (1.76).

Let $\mathbf{A}$ be a second order tensor then

$$\frac{\partial \det\left(\mathbf{A}\right)}{\partial \mathbf{A}} = \det\left(\mathbf{A}\right)\mathbf{A}^{-T} \tag{1.77}$$

Exercise Prove the rule above in (1.77).

Answer Assuming $\mathbf{A}$ is symmetric and using eigenvalues and eigenvectors we observe

$$\det\left(\mathbf{A}\right)\mathbf{A}^{-T} = \lambda_i\lambda_j\lambda_k \sum_m \frac{1}{\lambda_m}\left(\vec{n}_m \otimes \vec{n}_m\right) = \lambda_i\lambda_j\left(\vec{n}_k \otimes \vec{n}_k\right) + \lambda_i\lambda_k\left(\vec{n}_k \otimes \vec{n}_j\right) + \lambda_j\lambda_k\left(\vec{n}_k \otimes \vec{n}_i\right) \tag{1.78}$$

where λ_k's are eigenvalues and $\vec{n}_k$ are eigenvectors. From the chain rule we find

$$\frac{\partial \det(\mathbf{A})}{\partial \mathbf{A}_{ij}} = \sum_{m=1}^{3} \frac{\partial \det(\mathbf{A})}{\partial \lambda_m} \frac{\partial \lambda_m}{\partial \mathbf{A}_{ij}} = \lambda_i \lambda_j \frac{\partial \lambda_k}{\partial \mathbf{A}_{ij}} + \lambda_i \lambda_k \frac{\partial \lambda_j}{\partial \mathbf{A}_{ij}} + \lambda_j \lambda_k \frac{\partial \lambda_i}{\partial \mathbf{A}_{ij}} \tag{1.79}$$

If we can prove $\frac{\partial \lambda_k}{\partial \mathbf{A}_{ij}} = (\vec{n}_k \cdot \vec{e}_i)(\vec{n}_k \cdot \vec{e}_j) = (\vec{n}_k \otimes \vec{n}_k)_{ij}$ then the theorem holds. By definition of eigenvalues and eigen vectors

$$\mathbf{A}\vec{n}_k = \lambda_k \vec{n}_k$$

and since $\vec{n}_k \dot{\vec{n}}_k = 1$ we find

$$\vec{n}_k \cdot \mathbf{A}\vec{n}_k = \lambda_k \tag{1.80}$$

Now $\mathbf{A}$ is by definition given by

$$\mathbf{A} = \sum_h \sum_g \mathbf{A}_{hg} (\vec{e}_h \otimes \vec{e}_g)$$

Substitution into (1.80) gives

$$\lambda_k = \sum_h \sum_g \mathbf{A}_{hg} \vec{n}_k \cdot (\vec{e}_h \otimes \vec{e}_g)\, \vec{n}_k$$

Now it follows that

$$\frac{\partial \lambda_k}{\partial \mathbf{A}_{ij}} = \vec{n}_k \cdot (\vec{e}_i \otimes \vec{e}_j)\, \vec{n}_k = (\vec{n}_k \otimes \vec{n}_k)_{ij}$$

1.2 A Quick Primer on Continuum Mechanics

In this section we will introduce the equation of motion for an elastic solid. The equation is given by the partial differential equation,

$$\rho \ddot{\vec{x}} = \vec{b} + \nabla \cdot \sigma. \tag{1.81}$$

This is the governing equation when computing elastic deformations numerically. For this task we will introduce the boundary conditions of the Cauchy stress tensor σ which is given by

$$\sigma \vec{n} = \vec{t}, \tag{1.82}$$

where $\vec{t}$ is a known applied surface traction. Finally, a constitutive law is needed to tie things together. The constitutive law provides a relation between stress and strain often expressed as,

$$\mathbf{S} = \frac{\partial \Psi}{\partial \mathbf{E}} \tag{1.83}$$

where $\mathbf{S}$ is second Piola–Kirchhoff stress tensor and $\mathbf{E}$ is the Green strain tensor. Now we will derive all these equations that we need.

1.2.1 The Kinematics – The Strain Tensors

The definition of material and spatial coordinates. We have two set of coordinates

- Let $\vec{X} \in \mathbb{R}^3$ be undeformed coordinates
- and $\vec{x} \in \mathbb{R}^3$ the deformed coordinates

The deformation is given by the mapping $\Phi : \mathbb{R}^3 \mapsto \mathbb{R}^3$ from undeformed coordinates into deformed coordinates,

$$\vec{x} = \vec{\Phi}(\vec{X}) \tag{1.84}$$

Observe that the same global reference coordinate system is used for both set of coordinates. Further, undeformed coordinates are sometimes called material coordinates and deformed coordinates the spatial coordinates. The deformation gradient. By definition of differentials we have

$$d\vec{x} = \underbrace{\frac{\partial \vec{\Phi}}{\partial \vec{X}}}_{\mathbf{F}} d\vec{X} \tag{1.85}$$

The partial derivative $\frac{\partial \vec{\Phi}}{\partial \vec{X}} = \mathbf{F}$ is referred to as the Deformation gradient. In particular we have

$$\mathbf{F}_{ij} = \frac{\partial \vec{\Phi}_i}{\partial \vec{X}_j} = \frac{\partial \vec{x}_i}{\partial \vec{X}_j} \tag{1.86}$$

Using local deformation measures. Think of $d\vec{X}_1$ and $d\vec{X}_2$ as **arbitrary** chosen small needles. By definition we have

$$d\vec{x}_1 = \mathbf{F} d\vec{X}_1, \tag{1.87a}$$

$$d\vec{x}_2 = \mathbf{F} d\vec{X}_2. \tag{1.87b}$$

To describe any local deformation we may investigate the dot products

$$d\vec{x}_1 \cdot d\vec{x}_2 \qquad \text{and} \qquad d\vec{X}_1 \cdot d\vec{X}_2 \tag{1.88}$$

as these hold information about any local angle or length deformations. The definition of the Right Cauchy–Green deformation tensor. First we will try to express $d\vec{y}_1 \cdot d\vec{y}_2$ in terms of material coordinates,

$$d\vec{x}_1 \cdot d\vec{x}_2 = d\vec{x}_1^T d\vec{x}_2 \tag{1.89a}$$

$$= \left(\mathbf{F} d\vec{X}_1\right)^T \left(\mathbf{F} d\vec{X}_2\right) \tag{1.89b}$$

$$= d\vec{X}_1^T \underbrace{\left(\mathbf{F}^T \mathbf{F}\right)}_{\mathbf{C}} d\vec{X}_2 \tag{1.89c}$$

where

$$\mathbf{C} = \mathbf{F}^T\mathbf{F} \tag{1.90}$$

is called the right Cauchy–Green deformation tensor. Tensor invariants. The eigenvalues of **C** are independent of rotation. Thus, the coefficient of the characteristic polynomial are invariants

$$\det(\mathbf{C} - \mathbf{I}\lambda) = -\lambda^3 + \mathrm{I}_C\lambda^2 - \mathrm{II}_C\lambda + \mathrm{III}_C = 0 \tag{1.91}$$

where

$$\mathrm{I}_C = \mathrm{tr}(\mathbf{C}) = \mathbf{C} : \mathbf{I} \tag{1.92a}$$

$$\mathrm{II}_C = \mathbf{C} : \mathbf{C} \tag{1.92b}$$

$$\mathrm{III}_C = \det(\mathbf{C}) \tag{1.92c}$$

An alternative definition of the second invariant are $\mathrm{II}_C^* = \frac{1}{2}\left(\mathrm{tr}(\mathbf{C})^2 - \mathrm{tr}\left(\mathbf{C}^2\right)\right)$. We prefer II_C as it simplify later equations.

Exercise Discuss why the invariants are invariant (ie. independent of rotation)

The eigenvalue decomposition. Since **C** is symmetric positive definite we know that an eigenvalue decomposition exists

$$\mathbf{CN} - \mathbf{N\Lambda} = 0 \tag{1.93}$$

where

$$\mathbf{N} = \begin{bmatrix}\vec{N}_1 & \vec{N}_2 & \vec{N}_3\end{bmatrix} \tag{1.94a}$$

$$\mathbf{\Lambda} = \begin{bmatrix}\lambda_1^2 & 0 & 0\\ 0 & \lambda_2^2 & 0\\ 0 & 0 & \lambda_3^2\end{bmatrix} \tag{1.94b}$$

and $\mathbf{N}^T\mathbf{N} = \mathbf{N}\mathbf{N}^T = \mathbf{I}$ and $0 < \lambda_3^2 \le \lambda_2^2 \le \lambda_1^2 \in \mathbb{R}$. This can be rewritten as

$$\mathbf{C} = \mathbf{N\Lambda N} = \sum_i \lambda_i^2\left(\vec{N}_i\vec{N}_i^T\right) \tag{1.95}$$

Next we will express the invariants in terms of eigenvalues. Observe that knowing the eigenvalues λ_i^2 of **C** we can compute the tensor invariants as

$$\mathrm{I}_C = \lambda_1^2 + \lambda_2^2 + \lambda_3^2 \tag{1.96a}$$

$$\mathrm{II}_C = \lambda_1^4 + \lambda_2^4 + \lambda_3^4 \tag{1.96b}$$

$$\mathrm{III}_C = \lambda_1^2\lambda_2^2\lambda_3^2 \tag{1.96c}$$

These formulas will become handy when we later wish to change coordinates.

Exercise Derive the above formulas from the ones given in terms of **C**. If you have time derive an eigenvalue version of the invariant II_C^*.

Finding the rotation tensor. The square root of C is

$$\mathbf{C} = \sqrt{\mathbf{C}}\sqrt{\mathbf{C}} = \mathbf{U}\mathbf{U} \tag{1.97}$$

with

$$\mathbf{U} = \sum_i \lambda_i \left(\vec{N}_i \vec{N}_i^T\right) \tag{1.98}$$

Let us define the tensor $\mathbf{R}$ as

$$\mathbf{R} = \mathbf{F}\mathbf{U}^{-1} \tag{1.99}$$

We can then prove that $\mathbf{R}$ is a rotation tensor

$$\mathbf{R}^T\mathbf{R} = \mathbf{U}^{-T}\mathbf{F}^T\mathbf{F}\mathbf{U}^{-1} \tag{1.100a}$$
$$= \mathbf{U}^{-T}\mathbf{C}\mathbf{U}^{-1} \tag{1.100b}$$
$$= \mathbf{U}^{-T}\mathbf{U}\mathbf{U}\mathbf{U}^{-1} \tag{1.100c}$$
$$= \mathbf{I} \tag{1.100d}$$

The polar decomposition of the deformation gradient. The new rotation tensor $\mathbf{R}$ implies that $\mathbf{F}$ can be decomposed as

$$\mathbf{F} = \mathbf{R}\mathbf{U} \tag{1.101}$$

The implication is

$$d\vec{x} = \mathbf{F}\vec{dX} = \mathbf{R}\left(\mathbf{U}d\vec{X}\right) \tag{1.102}$$

where $\left(\mathbf{U}d\vec{X}\right)$ is a stretch along the eigenvectors and $\mathbf{R}$ is a rotation of the stretched vector. Therefore

- $\mathbf{U}$ is called the stretch tensor
- $\mathbf{R}$ is called the rotation tensor

Definition of the left Cauchy–Green deformation tensor. Next we will rewrite $d\vec{X}_1 \cdot d\vec{X}_2$ in terms of spatial coordinates,

$$d\vec{X}_1 \cdot d\vec{X}_2 = d\vec{X}_1^T d\vec{X}_2 \tag{1.103a}$$
$$= \left(\mathbf{F}^{-1}d\vec{x}_1\right)^T \left(\mathbf{F}^{-1}d\vec{x}_2\right) \tag{1.103b}$$
$$= d\vec{x}_1^T \left(\underbrace{\mathbf{F}\mathbf{F}^T}_{\mathbf{B}}\right)^{-1} d\vec{x}_2 \tag{1.103c}$$

where

$$\mathbf{B} = \mathbf{F}\mathbf{F}^T \tag{1.104}$$

is called the left Cauchy–Green deformation tensor. The definition of the Lagrange–Green strain tensor. We look at the deformation change. The difference between spatial and material dot products expressed in material coordinates,

$$\begin{aligned}\frac{1}{2}\left(d\vec{x}_1\cdot d\vec{x}_2 - d\vec{X}_1\cdot d\vec{X}_2\right) &= d\vec{x}_1^T d\vec{x}_2 - d\vec{X}_1^T d\vec{X}_2 && (1.105a)\\ &= \frac{1}{2}\left(\mathbf{F}d\vec{X}_1\right)^T\left(\mathbf{F}d\vec{X}_2\right) - d\vec{X}_1^T d\vec{X}_2 && (1.105b)\\ &= d\vec{X}_1^T \underbrace{\frac{1}{2}\left(\mathbf{F}^T\mathbf{F}-\mathbf{I}\right)}_{\mathbf{E}} d\vec{X}_2 && (1.105c)\end{aligned}$$

where

$$\mathbf{E} = \frac{1}{2}\left(\mathbf{F}^T\mathbf{F}-\mathbf{I}\right) = \frac{1}{2}\left(\mathbf{C}-\mathbf{I}\right) \tag{1.106}$$

is called the Green strain tensor. The definition of the Euler–Almansi strain tensor. We describe the deformation change in terms of spatial coordinates,

$$\begin{aligned}\frac{1}{2}\left(d\vec{x}_1\cdot d\vec{x}_2 - d\vec{X}_1\cdot d\vec{X}_2\right) &= d\vec{x}_1^T d\vec{x}_2 - d\vec{X}_1^T d\vec{X}_2 && (1.107a)\\ &= \frac{1}{2}\left(d\vec{x}_1^T d\vec{x}_2 - \left(\mathbf{F}^{-1}d\vec{x}_1\right)^T\left(\mathbf{F}^{-1}d\vec{x}_2\right)\right) && (1.107b)\\ &= d\vec{x}_1^T \underbrace{\frac{1}{2}\left(\mathbf{I}-\left(\mathbf{F}^T\mathbf{F}\right)^{-1}\right)}_{\mathbf{e}} d\vec{x}_2 && (1.107c)\end{aligned}$$

where

$$\mathbf{e} = \frac{1}{2}\left(\mathbf{I}-\left(\mathbf{F}\mathbf{F}^T\right)^{-1}\right) = \frac{1}{2}\left(\mathbf{I}-\mathbf{B}^{-1}\right) \tag{1.108}$$

is called the Euler strain tensor. Conversion from spatial to material tensors. Observe that

$$\begin{aligned}\mathbf{e} &= \mathbf{F}^{-T}\mathbf{E}\mathbf{F}^{-1} && (1.109a)\\ \mathbf{E} &= \mathbf{F}^T\mathbf{e}\mathbf{F} && (1.109b)\end{aligned}$$

These are general tensor conversion formulas. The definition of the displacement field. Let us define the displacement field as follows

$$\vec{u} = \vec{x} - \vec{X} \tag{1.110}$$

Then we have

$$\vec{x} = \vec{u} + \vec{X} \tag{1.111}$$

and

$$\mathbf{F} = \frac{\partial(\vec{u}+\vec{X})}{\partial\vec{X}} = \frac{\partial\vec{u}}{\partial\vec{X}} + \mathbf{I} \tag{1.112}$$

or

$$\frac{\partial\vec{u}}{\partial\vec{X}} = \mathbf{F} - \mathbf{I} \tag{1.113}$$

Using the definition of the displacement field. Let us use $\mathbf{F} = \frac{\partial \vec{u}}{\partial \vec{X}} + \mathbf{I}$ in the definition of the Green strain tensor

$$\mathbf{E} = \frac{1}{2}\left(\left(\frac{\partial \vec{u}}{\partial \vec{X}} + \mathbf{I}\right)^T \left(\frac{\partial \vec{u}}{\partial \vec{X}} + \mathbf{I}\right) - \mathbf{I}\right) \tag{1.114a}$$

$$= \frac{1}{2}\left(\frac{\partial \vec{u}}{\partial \vec{X}}^T \frac{\partial \vec{u}}{\partial \vec{X}} + \frac{\partial \vec{u}}{\partial \vec{X}} + \frac{\partial \vec{u}}{\partial \vec{X}}^T\right) \tag{1.114b}$$

The assumption of small displacement gradients. If $\| \frac{\partial \vec{u}}{\partial \vec{X}} \| \ll 1$ and $\| \frac{\partial \vec{u}}{\partial \vec{X}} \| \ll 1$ then $\frac{\partial \vec{u}}{\partial \vec{X}}^T \frac{\partial \vec{u}}{\partial \vec{X}} \approx \mathbf{0}$ and we have

$$\mathbf{E} \approx \varepsilon_0 = \frac{1}{2}\left(\frac{\partial \vec{u}}{\partial \vec{X}}^T + \frac{\partial \vec{u}}{\partial \vec{X}}\right) \tag{1.115}$$

This strain tensor is called the Cauchy strain tensor. If we used $\mathbf{e}$ instead of $\mathbf{E}$ we would have found

$$\mathbf{e} \approx \varepsilon = \frac{1}{2}\left(\frac{\partial \vec{u}}{\partial \vec{x}}^T + \frac{\partial \vec{u}}{\partial \vec{x}}\right) \tag{1.116}$$

For small displacement gradients $x \approx X$, $\frac{\partial \vec{u}}{\partial \vec{x}} \approx \frac{\partial \vec{u}}{\partial \vec{X}}$, and so $\varepsilon_0 \approx \varepsilon$. In summary we have introduced the following strain tensors

The Green strain tensor

$$\mathbf{E} = \frac{1}{2}\left(\frac{\partial \vec{u}}{\partial \vec{X}}^T \frac{\partial \vec{u}}{\partial \vec{X}} + \frac{\partial \vec{u}}{\partial \vec{X}} + \frac{\partial \vec{u}}{\partial \vec{X}}^T\right) \tag{1.117}$$

The Cauchy strain tensor

$$\varepsilon = \frac{1}{2}\left(\frac{\partial \vec{u}}{\partial \vec{X}}^T + \frac{\partial \vec{u}}{\partial \vec{X}}\right) \tag{1.118}$$

Observe that using $\frac{\partial \vec{u}}{\partial \vec{X}} = \mathbf{F} - \mathbf{I}$ we have

$$\varepsilon_0 = \frac{1}{2}\left(\mathbf{F}^T + \mathbf{F}\right) - \mathbf{I} \tag{1.119}$$

The definition of the velocity field. Since $\vec{x} = \Phi(\vec{X}, t)$ then the spatial velocity of a material point is

$$\vec{v}(\vec{X}, t) \equiv \dot{\vec{x}}(\vec{X}, t) = \frac{\partial \vec{x}(\vec{X}, t)}{\partial t} = \frac{\partial \Phi\left(\vec{X}, t\right)}{\partial t} \tag{1.120}$$

or given in terms of spatial coordinates

$$\vec{v}(\vec{x}, t) = \vec{v}\left(\Phi^{-1}\left(\vec{x}, t\right)\right) \tag{1.121}$$

The velocity gradient with respect to the spatial coordinates is then

$$\mathbf{V} = \frac{\partial \vec{v}(\vec{x}, t)}{\partial \vec{x}} \tag{1.122}$$

Observe

$$\begin{aligned} \dot{\mathbf{F}} &= \frac{d}{dt}\left(\frac{\partial \Phi}{\partial \vec{X}}\right) && (1.123a) \\ &= \frac{\partial}{\partial \vec{X}}\left(\frac{\partial \Phi}{\partial t}\right) && (1.123b) \\ &= \frac{\partial}{\partial \vec{X}}\vec{v} && (1.123c) \\ &= \frac{\partial \vec{v}}{\partial \vec{x}}\frac{\partial \vec{\Phi}}{\partial \vec{X}} && (1.123d) \\ &= \mathbf{VF} && (1.123e) \end{aligned}$$

Resulting in

$$\mathbf{V} = \dot{\mathbf{F}}\mathbf{F}^{-1} \tag{1.124}$$

The definition of the rate of deformation tensor. Taking the derivative

$$\frac{d}{dt}\left(d\vec{x}_1 \cdot d\vec{x}_2\right) = d\vec{X}_1^T \dot{\mathbf{C}} d\vec{X}_2 = 2d\vec{X}_1^T \dot{\mathbf{E}} d\vec{X}_2 \tag{1.125}$$

because $\mathbf{E} = \frac{1}{2}(\mathbf{C} - \mathbf{I})$. The material strain rate tensor is

$$\dot{\mathbf{E}} = \frac{1}{2}\dot{\mathbf{C}} = \frac{1}{2}\left(\dot{\mathbf{F}}^T\mathbf{F} + \mathbf{F}^T\dot{\mathbf{F}}\right) \tag{1.126}$$

Using $d\vec{X} = \mathbf{F}^{-1}d\vec{x}$ we have

$$\frac{1}{2}\frac{d}{dt}\left(d\vec{x}_1 \cdot d\vec{x}_2\right) = d\vec{x}_1^T \underbrace{\mathbf{F}^{-T}\dot{\mathbf{E}}\mathbf{F}^{-1}}_{\mathbf{D}} d\vec{x}_2 \tag{1.127}$$

where $\mathbf{D}$ is the rate of deformation tensor. Coordinate conversion of the rate of deformation tensors. Observe that similar as for the Euler–Almansi and Lagrange–Green strain tensors we have

$$\begin{aligned} \mathbf{D} &= \mathbf{F}^{-T}\dot{\mathbf{E}}\mathbf{F}^{-1} && (1.128a) \\ \dot{\mathbf{E}} &= \mathbf{F}^T\mathbf{D}\mathbf{F} && (1.128b) \end{aligned}$$

From this we find that

$$\begin{aligned} \mathbf{D} &= \mathbf{F}^{-T}\dot{\mathbf{E}}\mathbf{F}^{-1} && (1.129a) \\ &= \mathbf{F}^{-T}\frac{1}{2}\left(\dot{\mathbf{F}}^T\mathbf{F} + \mathbf{F}^T\dot{\mathbf{F}}\right)\mathbf{F}^{-1} && (1.129b) \\ &= \frac{1}{2}\left(\mathbf{V} + \mathbf{V}^T\right) && (1.129c) \end{aligned}$$

because $\mathbf{V} = \dot{\mathbf{F}}\mathbf{F}^{-1}$. Thus, $\mathbf{D}$ is the symmetric part of the velocity gradient.

1.2.2 The Dynamics – The Stress Tensors and Power

The spatial surface element $d\vec{s}$ can be written as

$$d\vec{s} = \begin{bmatrix} d\vec{s}_1 \\ d\vec{s}_2 \\ d\vec{s}_3 \end{bmatrix} \tag{1.130}$$

where $d\vec{s}_i$ is the projection of the area of surface $d\vec{s}$ onto the i^{th} coordinate axis. Observe that the surface normal is given by

$$\vec{n} = \frac{d\vec{s}}{\parallel d\vec{s} \parallel} \tag{1.131}$$

We can now introduce Cauchy's stress hypothesis. By definition we have

- Stress is force per unit area

If we apply the force $d\vec{f}$ to the surface element $d\vec{s}$

$$d\vec{f}_1 = \sigma_{11} d\vec{s}_1 + \sigma_{12} d\vec{s}_3 + \sigma_{13} d\vec{s}_3 \tag{1.132a}$$

$$d\vec{f}_2 = \sigma_{21} d\vec{s}_1 + \sigma_{22} d\vec{s}_3 + \sigma_{23} d\vec{s}_3 \tag{1.132b}$$

$$d\vec{f}_3 = \sigma_{31} d\vec{s}_1 + \sigma_{32} d\vec{s}_3 + \sigma_{33} d\vec{s}_3 \tag{1.132c}$$

where σ_{ij} depends on position and time, collecting them

$$\sigma = \left[\sigma_{ij}\right] \tag{1.133}$$

we have

$$d\vec{f} = \sigma d\vec{s} \tag{1.134}$$

where σ is the Cauchy stress tensor. The first Piola–Kirchhoff stress tensor. By definition the Cauchy stress tensor

$$d\vec{f} = \sigma d\vec{s} \tag{1.135}$$

The first Piola–Kirchhoff stress tensor is defined as

$$d\vec{f} = \mathbf{P} d\vec{S} \tag{1.136}$$

So we must have

$$\sigma d\vec{s} = d\vec{f} = \mathbf{P} d\vec{S} \tag{1.137}$$

Using Nanson's Relation

$$\sigma j \mathbf{F}^{-T} d\vec{S} = d\vec{f} = \mathbf{P} d\vec{S} \tag{1.138}$$

From this we have the relation between the Cauchy stress tensor and the first Piola–Kirchhoff stress tensor

$$\mathbf{P} = j\sigma\mathbf{F}^{-T} \tag{1.139}$$

The second Piola–Kirchhoff stress tensor. By definition the Cauchy stress tensor

$$d\vec{f} = \sigma d\vec{s} \tag{1.140}$$

The second Piola–Kirchhoff stress tensor is defined as

$$d\vec{F} = \mathbf{S}d\vec{S}. \tag{1.141}$$

Since $d\vec{f} = \mathbf{F}d\vec{F}$ we must have

$$\sigma d\vec{s} = d\vec{f} = \mathbf{F}d\vec{F} = \mathbf{F}\mathbf{S}d\vec{S} \tag{1.142a}$$

$$\sigma d\vec{s} = \mathbf{F}\mathbf{S}d\vec{S} \tag{1.142b}$$

$$j\mathbf{F}^{-1}\sigma\mathbf{F}^{-T}d\vec{S} = \mathbf{S}d\vec{S} \tag{1.142c}$$

So

$$\mathbf{S} = j\mathbf{F}^{-1}\sigma\mathbf{F}^{-T} \tag{1.143}$$

The traction force. We have

$$d\vec{f} = \sigma d\vec{s} \tag{1.144}$$

If we divide by surface area we will have traction $\vec{t} = d\vec{f}/ \parallel d\vec{s} \parallel$

$$\vec{t} = \sigma\vec{n} \tag{1.145}$$

where

$$\begin{bmatrix} \vec{t}_1 \\ \vec{t}_2 \\ \vec{t}_3 \end{bmatrix} = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} \\ \sigma_{21} & \sigma_{22} & \sigma_{23} \\ \sigma_{31} & \sigma_{32} & \sigma_{33} \end{bmatrix} \begin{bmatrix} \vec{n}_1 \\ \vec{n}_2 \\ \vec{n}_3 \end{bmatrix} \tag{1.146}$$

Volume integration of forces. Given the body volume force density $\vec{b}$ and arbitrary spatial volume element v with spatial surface s then the total force on the volume is found by integration

$$\vec{\mathcal{F}} = \int_v \vec{b}dv + \oint_s \vec{t}ds \tag{1.147a}$$

$$= \int_v \vec{b}dv + \oint_s (\sigma\vec{n})\,ds \tag{1.147b}$$

Recall Gauss–Divergence Theorem, for tensor field $\mathbf{A}$ over any closed volume v with surface s

$$\int_v \nabla \cdot \mathbf{A}dv = \oint_s \mathbf{A}\vec{n}ds \tag{1.148}$$

Applying this to

$$\vec{\mathcal{F}} = \int_v \vec{b}dv + \oint_s \sigma\vec{n}ds \tag{1.149}$$

yields

$$\vec{\mathcal{F}} = \int_v \underbrace{\left(\vec{b} + \nabla \cdot \sigma\right)}_{\vec{f}^*} dv \tag{1.150}$$

The total force and the effective force. In mechanical equilibrium the effective force $\vec{f}^*$ must be zero since v was chosen arbitrary. Thus, writing it out

$$\vec{b} + \nabla \cdot \sigma = 0 \tag{1.151a}$$

This is Cauchy's equation of equilibrium. In non equilibrium we must take inertia forces into account, $\vec{\mathcal{F}} = \int_v \rho \ddot{\vec{x}} dv$,

$$\rho \ddot{\vec{x}} = \vec{b} + \nabla \cdot \sigma \tag{1.152a}$$

This is the motion of equation in spatial coordinates for any continuum. We will now recall the classical mechanics definitions of work and power. Let $\vec{x}(t)$ be a particle and $\vec{f}$ some force acting on the particle. Work is force times distance, for a constant force over a displacement $\vec{u}$

$$W = \vec{f} \cdot \vec{u} \tag{1.153}$$

Or more general

$$W = \int \vec{f} \cdot d\vec{x} \tag{1.154}$$

Power is the rate at which work is performed

$$P \equiv \frac{dW}{dt} \tag{1.155}$$

The instantaneous power is $P = \vec{f} \cdot \vec{v}$ and so $W = \int_0^t \vec{f} \cdot \vec{v} dt$. The instantaneous power of the effective force is by definition,

$$p = \vec{f}^* \cdot \vec{v} \tag{1.156}$$

The total instantaneous power applied by the effective force

$$P = \int_v p dv = \int_v \vec{f}^* \cdot \vec{v} dv \tag{1.157}$$

Rewriting into

$$P = \int_v \left(\vec{b} + (\nabla \cdot \sigma) \right) \cdot \vec{v} dv \tag{1.158a}$$

$$= \int_v (\nabla \cdot \sigma) \cdot \vec{v} dv + \int_v \vec{b} \cdot \vec{v} dv \tag{1.158b}$$

Using

$$\nabla \cdot (\sigma \vec{v}) = (\nabla \cdot \sigma) \cdot \vec{v} + \sigma : (\nabla \vec{v}^T) \tag{1.159}$$

Then

$$P = \int_v (\nabla \cdot \sigma) \cdot \vec{v} dv + \int_v \vec{b} \cdot \vec{v} dv \tag{1.160}$$

Becomes

$$P = \int_v \nabla \cdot (\sigma \vec{v}) dv - \int_v \sigma : \mathbf{V}^T dv + \int_v \vec{b} \cdot \vec{v} dv \tag{1.161}$$

Recall $\nabla \vec{v}^T = \mathbf{V}^T$. Using the Gauss–Divergence theorem

$$P = \oint_s \vec{n} \cdot (\sigma \vec{v}) ds - \int_v \sigma : \mathbf{V}^T dv + \int_v \vec{b} \cdot \vec{v} dv \tag{1.162}$$

Using $\vec{t} = \sigma \vec{n}$ and $\sigma^T = \sigma$

$$P = \oint_s \vec{t} \cdot \vec{v} ds - \int_v \sigma : \mathbf{V} dv + \int_v \vec{b} \cdot \vec{v} dv \tag{1.163}$$

Using $\mathbf{D} = \frac{1}{2}\left(\mathbf{V} + \mathbf{V}^T\right)$ and symmetry of σ

$$P = \oint_s \vec{t} \cdot \vec{v} ds - \int_v \sigma : \mathbf{D} dv + \int_v \vec{b} \cdot \vec{v} dv \tag{1.164}$$

Exercise Prove that if $\sigma = \sigma^T$ then $\sigma : \mathbf{V}^T = \sigma : \mathbf{V}$ and that $\sigma : \mathbf{V} = \sigma : \mathbf{D}$.

Definition of the Eulerian instantaneous power. The internal stress power term in spatial coordinates is

$$P_{\text{e}} = \int_v \sigma : \mathbf{V} dv = \int_v \sigma : \mathbf{D} dv \tag{1.165}$$

Thus, in terms of power we say that σ is power conjugate to $\mathbf{D}$. Our next task is to rewrite the internal power in terms of material coordinates. The result is to find the power conjugate quantities of

- First Piola–Kirchhoff stress tensor
- Second Piola–Kirchhoff stress tensor

The power conjugate of $\mathbf{P}$.

$$\begin{aligned}
P_{\text{e}} &= \int_v \sigma : \mathbf{V} dv && \text{(by definition)} \\
&= \int_V j\sigma : \mathbf{V} dV && (\text{by } dv = jdV) \\
&= \int_V j\sigma : \left(\dot{\mathbf{F}}\mathbf{F}^{-1}\right) dV && (\text{by } \mathbf{V} = \dot{\mathbf{F}}\mathbf{F}^{-1}) \\
&= \int_V \operatorname{tr}\left(j\sigma\left(\dot{\mathbf{F}}\mathbf{F}^{-1}\right)\right) dV && (\text{by } \mathbf{A} : \mathbf{B} = \operatorname{tr}\left(\mathbf{A}^T\mathbf{B}\right)) \\
&= \int_V \operatorname{tr}\left(\left(j\mathbf{F}^{-1}\sigma\right)\dot{\mathbf{F}}\right) dV && (\operatorname{tr}(\mathbf{AB}) = \operatorname{tr}(\mathbf{BA})) \\
&= \int_V \mathbf{P} : \dot{\mathbf{F}} dV && (\text{by } \operatorname{tr}\left(\mathbf{A}^T\mathbf{B}\right) = \mathbf{A} : \mathbf{B})
\end{aligned}$$

The power conjugate of $\mathbf{S}$.

$$\begin{aligned}
P_{\text{e}} &= \int_v \sigma : \mathbf{D} dv && \text{(by definition)} \\
&= \int_V j\sigma : \mathbf{D} dV && \text{(by } dv = jdV\text{)} \\
&= \int_V j\sigma : \left(\mathbf{F}^{-T}\dot{\mathbf{E}}\mathbf{F}^{-1}\right) dV && \text{(by } \mathbf{D} = \mathbf{F}^{-T}\dot{\mathbf{E}}\mathbf{F}^{-1}\text{)} \\
&= \int_V \text{tr}\left(j\sigma\left(\mathbf{F}^{-T}\dot{\mathbf{E}}\mathbf{F}^{-1}\right)\right) dV && \text{(by } \mathbf{A} : \mathbf{B} = \text{tr}\left(\mathbf{A}^T\mathbf{B}\right)\text{)} \\
&= \int_V \text{tr}\left(\left(j\mathbf{F}^{-1}\sigma\mathbf{F}^{-T}\right)\dot{\mathbf{E}}\right) dV && \text{(by tr}\left(\mathbf{AB}\right) = \text{tr}\left(\mathbf{BA}\right)\text{)} \\
&= \int_V \mathbf{S} : \dot{\mathbf{E}} dV && \text{(by tr}\left(\mathbf{A}^T\mathbf{B}\right) = \mathbf{A} : \mathbf{B}\text{)}
\end{aligned}$$

We may now study the energy balance. For the equilibrium system mechanical energy balance means that $P = 0$. That is the time derivative of mechanical energy is zero – constant in time. For the case of non-equilibrium system we must include inertia forces,

$$P = \int_v \left(\rho\ddot{\vec{x}} - \vec{b} - \nabla \cdot \sigma\right) \cdot \vec{v} dv \tag{1.168}$$

Repeating the previous derivations leans to the total power becomes

$$P = \int_v \rho\ddot{\vec{x}} \cdot v dv + \int_v \sigma : \mathbf{D} dv - \oint_s \vec{t} \cdot \vec{v} dv - \int_v \vec{b} \cdot \vec{v} dv \tag{1.169}$$

and it must be zero for conservation of mechanical energy.

1.2.3 Constitutive Equations

Expressing the strain energy in terms of $\mathbf{P}$. The total strain energy Ψ,

$$\Psi(\mathbf{F}) = \int_0^t \underbrace{\mathbf{P} : \dot{\mathbf{F}}}_{=\dot{\Psi}} dt \tag{1.170}$$

By the chain rule we have

$$\dot{\Psi} = \frac{d}{dt}\Psi(\mathbf{F}) = \frac{\partial \Psi}{\partial \mathbf{F}} : \dot{\mathbf{F}} \tag{1.171}$$

So

$$\mathbf{P} : \dot{\mathbf{F}} = \dot{\Psi} = \frac{\partial \Psi}{\partial \mathbf{F}} : \dot{\mathbf{F}} \tag{1.172}$$

and we must have

$$\mathbf{P} = \frac{\partial \Psi}{\partial \mathbf{F}} \tag{1.173}$$

Next we will examine the strain energy in terms of $\mathbf{S}$. The total strain energy Ψ,

$$\Psi(\mathbf{F}) = \int_0^t \underbrace{\mathbf{S} : \dot{\mathbf{E}}}_{=\dot{\Psi}} dt \tag{1.174}$$

By the chain rule we have

$$\dot{\Psi} = \frac{d}{dt}\Psi(\mathbf{E}) = \frac{\partial \Psi}{\partial \mathbf{E}} : \dot{\mathbf{E}} \tag{1.175}$$

So

$$\mathbf{S} : \dot{\mathbf{E}} = \dot{\Psi} = \frac{\partial \Psi}{\partial \mathbf{E}} : \dot{\mathbf{E}} \tag{1.176}$$

and we must have

$$\mathbf{S} = \frac{\partial \Psi}{\partial \mathbf{E}} \tag{1.177}$$

When considering isotropic materials then the strain energy is a function of invariants of $\mathbf{C}$

$$\Psi(\mathbf{C}) = \Psi(\mathrm{I}_C, \mathrm{II}_C, \mathrm{III}_C) \tag{1.178}$$

The Second Piola–Kirchhoff stress tensor computed as

$$\mathbf{S} = 2\frac{\partial \Psi}{\partial \mathrm{I}_C}\mathbf{I} + 4\frac{\partial \Psi}{\partial \mathrm{II}_C}\mathbf{C} + 2j^2\frac{\partial \Psi}{\partial \mathrm{III}_C}\mathbf{C}^{-1} \tag{1.179}$$

Exercise Derive the relation in (1.179).

Answer By definition $2(\mathbf{E} + \mathbf{I}) = \mathbf{C}$

$$\mathbf{S} = \frac{\partial \Psi(\mathbf{C})}{\partial \mathbf{C}}\frac{\partial \mathbf{C}}{\partial \mathbf{E}} = 2\frac{\partial \Psi}{\partial \mathbf{C}} \tag{1.180}$$

For isotropic materials stretch is independent of rotation $\Psi(\mathbf{C}) = \Psi(\mathrm{I}_C, \mathrm{II}_C, \mathrm{III}_C)$

$$\mathbf{S} = 2\left(\frac{\partial \Psi}{\partial \mathrm{I}_C}\frac{\partial \mathrm{I}_C}{\partial \mathbf{C}} + \frac{\partial \Psi}{\partial \mathrm{II}_C}\frac{\partial \mathrm{II}_C}{\partial \mathbf{C}} + \frac{\partial \Psi}{\partial \mathrm{III}_C}\frac{\partial \mathrm{III}_C}{\partial \mathbf{C}}\right) \tag{1.181}$$

where

$$\frac{\partial \mathrm{I}_C}{\partial \mathbf{C}} = \mathbf{I} \tag{1.182a}$$

$$\frac{\partial \mathrm{II}_C}{\partial \mathbf{C}} = \mathrm{tr}\,(\mathbf{C})\,\mathbf{I} - \mathbf{C} \tag{1.182b}$$

$$\frac{\partial \mathrm{III}_C}{\partial \mathbf{C}} = \det(\mathbf{C})\,\mathbf{C}^{-T} = j^2\mathbf{C}^{-1} \tag{1.182c}$$

The saint Venant–Kirchhoff model. For the simplest hyper elastic material model the strain-energy is given by

$$\Psi(\mathbf{E}) = \frac{\lambda}{2}\left(\mathrm{tr}\,(\mathbf{E})\right)^2 + \mu\mathrm{tr}\left(\mathbf{E}^2\right) \tag{1.183}$$

where λ and μ are the Lamé constants. From $\mathbf{S} = \frac{\partial \Psi}{\partial \mathbf{E}}$ we have the stress-strain relation

$$\mathbf{S} = \lambda\mathrm{tr}\,(\mathbf{E})\,\mathbf{I} + 2\mu\mathbf{E} \tag{1.184}$$

Exercise Derive (1.184) by differentiation of Ψ in (1.183).

1.3 Direction Derivatives

We wish to find a closed form formula for the directional derivative of deformation gradient. Let $\vec{u} \neq \vec{0}$ be some arbitrary direction

$$D\mathbf{F}(\vec{x})[\vec{u}] = \lim_{s\to 0} \frac{\mathbf{F}(\vec{x}+s\vec{u}) - \mathbf{F}(\vec{x})}{s} \tag{1.185a}$$

Recall that $\vec{x} = \Phi(\vec{X})$ and $\mathbf{F} = \frac{\partial \vec{x}}{\partial \vec{X}} = \frac{\partial \Phi(\vec{X})}{\partial \vec{X}}$ so

$$\mathbf{F}(\vec{x}+s\vec{u}) = \frac{\partial \Phi(\Phi^{-1}(\vec{x}+s\vec{u}))}{\partial \vec{X}} = \frac{\partial(\vec{x}+s\vec{u})}{\partial \vec{X}} \tag{1.186}$$

Then

$$D\mathbf{F}(\vec{x})[\vec{u}] = \lim_{s\to 0} \frac{\frac{\partial(\vec{x}+s\vec{u})}{\partial \vec{X}} - \frac{\partial \Phi(\vec{x})}{\partial \vec{X}}}{s} \tag{1.187a}$$

$$= \lim_{s\to 0} \frac{\frac{\partial \vec{x}}{\partial \vec{X}} + s\frac{\partial \vec{u}}{\partial \vec{X}} - \frac{\partial \vec{x}}{\partial \vec{X}}}{s} \tag{1.187b}$$

$$= \frac{\partial \vec{u}}{\partial \vec{X}} \tag{1.187c}$$

Using the chain rule we may rewrite

$$D\mathbf{F}(\vec{x})[\vec{u}] = \frac{\partial \vec{u}}{\partial \vec{X}} = \frac{\partial \vec{u}}{\partial \vec{x}}\frac{\partial \vec{x}}{\partial \vec{X}} = (\nabla \vec{u})\mathbf{F} \tag{1.188}$$

Observe that if $\vec{u}$ is considered to be a function of $\vec{X}$ and not $\vec{x}$ then we write

$$\vec{u}_0 = \vec{u}(\vec{X}) \tag{1.189}$$

and then we have

$$D\mathbf{F}(\vec{x})[\vec{u}_0] = \frac{\partial \vec{u}_0}{\partial \vec{X}} = \nabla_0 \vec{u}_0 \tag{1.190}$$

The directional derivative of the Green strain tensor. By definition we have

$$\mathbf{E} = \frac{1}{2}\left(\mathbf{F}^T\mathbf{F} - I\right) \tag{1.191}$$

so

$$D\mathbf{E}(\vec{x})[\vec{u}] = \frac{1}{2}\left(D\mathbf{F}(\vec{x})[\vec{u}]^T\mathbf{F} + \mathbf{F}^T D\mathbf{F}(\vec{x})[\vec{u}]\right) \tag{1.192a}$$

$$= \frac{1}{2}\mathbf{F}^T\left(\frac{\partial \vec{u}}{\partial \vec{X}}^T + \frac{\partial \vec{u}}{\partial \vec{X}}\right)\mathbf{F} \tag{1.192b}$$

$$= \mathbf{F}^T \varepsilon \mathbf{F} \tag{1.192c}$$

The directional derivative of second Piola—Kirchhoff stress tensor. We know

$$D\mathbf{S}(\vec{x})[\vec{u}] = D\left(\mathbf{S}(\mathbf{E}(\vec{x}))\right)[\vec{u}] \tag{1.193a}$$

From the chain rule we have

$$D\mathbf{S}_{IJ}(\vec{x})[\vec{u}] = \sum_{K=1}^{3}\sum_{L=1}^{3} \frac{\partial \mathbf{S}_{IJ}(\vec{x})}{\partial \mathbf{E}_{KL}} D\mathbf{E}_{KL}(\vec{x})[\vec{u}] \tag{1.194a}$$

That is

$$D\mathbf{S}(\vec{x})[\vec{u}] = \mathcal{C}(\vec{x}) : D\mathbf{E}(\vec{x})[\vec{u}] \tag{1.195}$$

where the fourth order tensor $\mathcal{C}$ is given as

$$\mathcal{C}(\vec{x}) = \frac{\partial \mathbf{S}(\vec{x})}{\partial \mathbf{E}} \tag{1.196}$$

We know that the partial derivative of the strain energy gives us the stress tensor

$$\mathbf{S} = \frac{\partial \psi}{\partial \mathbf{E}} = 2\frac{\partial \psi}{\partial \mathbf{C}} \tag{1.197}$$

So that means

$$\mathcal{C} = \frac{\partial \mathbf{S}}{\partial \mathbf{E}} = 2\frac{\partial^2 \psi}{\partial \mathbf{C}\partial \mathbf{E}} = 4\frac{\partial^2 \psi}{\partial \mathbf{C}\partial \mathbf{C}} \tag{1.198}$$

Or in index notation

$$\mathcal{C}_{IJKL} = \frac{4\partial^2 \psi}{\partial \mathbf{C}_{IJ}\partial \mathbf{C}_{KL}} = \mathcal{C}_{KLIJ} \tag{1.199}$$

1.4 Summary

We have introduced tensor algebra and calculus and shown how to apply these as a mathematical framework for continuum mechanics. In particular we have derived several strain and stress tensors as well as the equation of motion. Finally, we have explored some formulas for directional derivatives that will be helpful in later texts.

Chapter 2

Time Integration

2.0.1 Time Discretization

The finite element method resulted in the second order differential equation

$$\mathbf{M}\ddot{\vec{x}} + \mathbf{C}\dot{\vec{x}} + \mathbf{K}\vec{u} = \vec{f}. \tag{2.1}$$

We have dropped the tilde notation to make things more readable. We will now discretize time using finite difference approximations. First we observe that time derivative of the velocity is the acceleration, $\dot{\vec{v}} = \ddot{\vec{x}}$, and so using first order Euler,

$$\dot{\vec{v}} \approx \frac{\vec{v}^{t+\Delta t} - \vec{v}^{t}}{\Delta t}, \tag{2.2}$$

and

$$\vec{x}^{t+\Delta t} \approx \vec{x}^{t} + \Delta t \vec{v}^{t+\Delta t}. \tag{2.3}$$

Substituting the finite difference approximations we have

$$\mathbf{M}\frac{\vec{v}^{t+\Delta t} - \vec{v}^{t}}{\Delta t} + \mathbf{K}\left(\vec{x}^{t+\Delta t} - \vec{X}\right) + \mathbf{C}\vec{v}^{t+1} = \vec{f}. \tag{2.4}$$

Further manipulation

$$\begin{aligned}\mathbf{M}\vec{v}^{t+\Delta t} - \mathbf{M}\vec{v}^{t} + \Delta t\mathbf{K}\left(\vec{x}^{t} + \Delta t\vec{v}^{t+\Delta t} - \vec{X}\right)& \\ + \Delta t\mathbf{C}\vec{v}^{t+1} = \Delta t\vec{f}.&\end{aligned} \tag{2.5}$$

Finally

$$\mathbf{A}\vec{v}^{t+\Delta t} = \vec{b}, \tag{2.6}$$

where

$$\mathbf{A} = \mathbf{M} + \Delta t\mathbf{C} + \Delta t^2\mathbf{K}, \tag{2.7a}$$

$$\vec{b} = \mathbf{M}\vec{v}^{t} + \Delta t\left(\vec{f} - \mathbf{K}\vec{x}^{t} + \vec{f}_0\right). \tag{2.7b}$$

Here we have introduced $\vec{f}_0 = \mathbf{K}\vec{X}$. The $\mathbf{A}$ matrix is sparse block symmetric and positive definite matrix. Thus, the linear system can be solved effectively using a numerical method such as the conjugate gradient method. Having found $\vec{v}^{t+\Delta t}$ we may do the position update

$$\vec{x}^{t+\Delta t} = \vec{x}^t + \Delta t \vec{v}^{t+\Delta t}. \tag{2.8}$$

2.0.2 Time Discretization

We may rewrite the second order differential equation into a system of coupled first order differential equations

$$\begin{aligned} \dot{\vec{x}} &= \vec{v}, & (2.9a)\\ \mathbf{M}\dot{\vec{v}} + \mathbf{C}\vec{v} + \vec{k}(\vec{x} - \vec{X}) &= \vec{f}, & (2.9b) \end{aligned}$$

where we have dropped the tilde notation for sake of readability. Using finite differencing we can discretize the time derivatives. Here we will consider first order approximations only. At one extreme we may take all terms to be explicit,

$$\begin{aligned} \vec{x}^{t+\Delta t} &= \vec{x}^t + \Delta t \vec{v}^t, & (2.10a)\\ \vec{v}^{t+\Delta t} &= \vec{v}^t + \Delta t \mathbf{M}^{-1}\left(\vec{f} - \mathbf{C}\vec{v}^t - \vec{k}(\vec{x}^t - \vec{X})\right). & (2.10b) \end{aligned}$$

Here superscript denotes the time. In computer graphics it is often the case that the elastic forces should be treated explicitly whereas the position update should be treated implicitly. In this case a semi implicit time integration method results as

$$\begin{aligned} \vec{v}^{t+\Delta t} &= \vec{v}^t + \Delta t \mathbf{M}^{-1}\left(\vec{f} - \mathbf{C}\vec{v}^t - \vec{k}(\vec{x}^t - \vec{X})\right), & (2.11a)\\ \vec{x}^{t+\Delta t} &= \vec{x}^t + \Delta t \vec{v}^{t+\Delta t}. & (2.11b) \end{aligned}$$

For stiff materials the explicit update rules may require very small time steps to be stable. To counter this we may make a full implicit scheme.

$$\begin{aligned} &\mathbf{M}\left(\vec{v}^{t+\Delta t} - \vec{v}^t\right) - \\ &\quad \Delta t\left(\vec{f} + \mathbf{C}\vec{v}^{t+\Delta t} + \vec{k}(\vec{x}^{t+\Delta t} - \vec{X})\right) = \vec{0} \end{aligned} \tag{2.12}$$

and

$$\vec{x}^{t+\Delta t} - \vec{x}^t - \Delta t \vec{v}^{t+1} = \vec{0}. \tag{2.13}$$

The details of the implicit method are treated in depth in [Erleben(2011b)]. What remains to be dealt with is an efficient way to compute the $\vec{k}^e$ term. We have already derived the equation for the $\vec{k}^e_a$ term

$$\vec{k}^e_a(\vec{u}^e) = \int_{V^e} \mathbf{P}\nabla_0 N_a dV. \tag{2.14}$$

By definition of the deformation gradient and using our shape functions we find

$$\mathbf{F}(\vec{X}) = \sum_{a\in e} \vec{x}_a \otimes \nabla_0 N_a(\vec{X}). \tag{2.15}$$

Knowing $\mathbf{F}$ we may compute $\mathbf{C} = \mathbf{F}^T\mathbf{F}$ and from this we find $\mathbf{S} = 2\frac{\partial\Psi}{\partial\mathbf{C}}$ and finally we may compute $\mathbf{P} = \mathbf{FS}$. In the case of using linear shape functions for a tetrahedral mesh element we know that $\nabla_0 N_a(\vec{X})$ is constant over the whole element and so $\mathbf{F}^e$ is a constant per element, this implies $\mathbf{C}^e$ is a constant and that $\mathbf{S}^e$ and finally $\mathbf{P}^e$ are constants too. In this case our integral gives the closed form solution,

$$\vec{k}^e_a(\vec{u}^e) = V^e\mathbf{F}^e\mathbf{S}^e\nabla_0 N_a. \tag{2.16}$$

If non-linear shape functions were used then it might not be possible to derive a nice closed form solution and a numerical approach like quadrature must be used instead.

2.1 Newmark Time Integration

The equations of motion for non-linear elasticity can be written as

$$\mathbf{M}\ddot{\vec{u}} + \mathbf{C}\dot{\vec{u}} + \vec{k}(\vec{u}) = \vec{f}. \tag{2.17}$$

where we used $\vec{x} = \vec{X} + \vec{u}$ to replace $\vec{x}$ with $\vec{u}$. For simplicity we will assumed that $\vec{f}$ has little time dependency so it can be viewed as a constant force in the following. In Newmark's approach [Newmark(1959), Wriggers(2002), Quek and Liu(2003)] we start by the Taylor series expansions of $\vec{u}^{t+\Delta t}$ and $\dot{\vec{u}}^{t+\Delta t}$ around t,

$$\vec{u}^{t+\Delta t} = \vec{u}^t + \Delta t\,\dot{\vec{u}}^t + \frac{1}{2}\Delta t^2\,\ddot{\vec{u}}^t + \frac{1}{6}\Delta t^2\,\dddot{\vec{u}}^t + \mathcal{O}(\Delta t^4) \tag{2.18a}$$

$$\dot{\vec{u}}^{t+\Delta t} = \dot{\vec{u}}^t + \Delta t\,\ddot{\vec{u}}^t + \frac{1}{2}\Delta t^2\,\dddot{\vec{u}}^t + \mathcal{O}(\Delta t^3) \tag{2.18b}$$

which can be truncate and expressed as

$$\vec{u}^{t+\Delta t} \approx \vec{u}^t + \Delta t\,\dot{\vec{u}}^t + \frac{1}{2}\Delta t^2\,\ddot{\vec{u}}^t + \beta\Delta t^3\,\dddot{\vec{u}} \tag{2.19a}$$

$$\dot{\vec{u}}^{t+\Delta t} \approx \dot{\vec{u}}^t + \Delta t\,\ddot{\vec{u}}^t + \gamma\Delta t^2\,\dddot{\vec{u}} \tag{2.19b}$$

If the acceleration is assumed to be linear within the time step then

$$\dddot{\vec{u}} = \frac{\ddot{\vec{u}}^{t+\Delta t} - \ddot{\vec{u}}^t}{\Delta t} \tag{2.20}$$

Substitution into the Taylor approximations yields the update rules

$$\vec{u}^{t+\Delta t} = \vec{u}^t + \Delta t\,\dot{\vec{u}}^t + \Delta t^2\left(\left(\frac{1}{2} - \beta\right)\ddot{\vec{u}}^t + \beta\,\ddot{\vec{u}}^{t+\Delta t}\right) \tag{2.21a}$$

$$\dot{\vec{u}}^{t+\Delta t} = \dot{\vec{u}}^t + \Delta t\left((1-\gamma)\,\ddot{\vec{u}}^t + \gamma\,\ddot{\vec{u}}^{t+\Delta t}\right) \tag{2.21b}$$

Newmark showed that a reasonable value of γ is $\frac{1}{2}$. A typical setting of β is $\frac{1}{4}$ which yields the constant average acceleration method. Next we substitute the update rules into the equations of motion.

$$\mathbf{M}\ddot{\vec{u}}^{t+\Delta t} + \mathbf{C}\dot{\vec{u}}^{t+\Delta t} + \vec{k}(\vec{u}^{t+\Delta t}) = \vec{f}. \tag{2.22}$$

and get

$$\begin{aligned}\mathbf{M}\ddot{\vec{u}}^{t+\Delta t} &+ \mathbf{C}\left(\dot{\vec{u}}^{t} + \Delta t\left((1-\gamma)\,\ddot{\vec{u}}^{t} + \gamma\,\ddot{\vec{u}}^{t+\Delta t}\right)\right)\\ &+ \vec{k}\left(\vec{u}^{t} + \Delta t\,\dot{\vec{u}}^{t} + \Delta t^2\left(\left(\frac{1}{2} - 1\beta\right)\ddot{\vec{u}}^{t} + \beta\,\ddot{\vec{u}}^{t+\Delta t}\right)\right)\\ &= \vec{f}.\end{aligned} \tag{2.23}$$

This equation forms the basis for the time discretization. We will next consider how to compute the values at $t + \Delta$ using this equation as our building block. First we consider the linear case afterwards we consider the non-linear case.

2.1.1 The Linear Case

For the particular case of linear elasticity we have $\vec{k}(\vec{u}) = \mathbf{K}\vec{u}$ using this and rearranging the equation yields

$$\mathbf{A}_{\text{new}}\ddot{\vec{u}}^{t+\Delta t} = \vec{b}_{\text{new}} \tag{2.24}$$

where

$$\mathbf{A}_{\text{new}} = \mathbf{M} + \gamma\Delta t\mathbf{C} + \beta\Delta t^2\mathbf{K} \tag{2.25}$$

and

$$\begin{aligned}\vec{b}_{\text{new}} = \vec{f} &- \mathbf{K}\left[\vec{u}^{t} + \Delta t\,\dot{\vec{u}}^{t} + \left(\tfrac{1}{2} - \beta\right)\Delta t^2\,\ddot{\vec{u}}^{t}\right]\\ &- \mathbf{C}\left[\dot{\vec{u}}^{t} + (1-\gamma)\,\Delta t\,\ddot{\vec{u}}^{t}\right]\end{aligned} \tag{2.26}$$

When Newmark integration is applied then one performs the steps

Step 1 one is given $\vec{u}^t$ and $\dot{\vec{u}}^t$ as initial conditions

Step 2 From the equations of motion one can compute $\ddot{\vec{u}}^t$ by solving

$$\mathbf{M}\ddot{\vec{u}}^{t} = \vec{f} - \mathbf{C}\dot{\vec{u}}^{t} - \mathbf{K}\vec{u}^{t} \tag{2.27}$$

Step 3 Setup $\mathbf{A}_{\text{new}}$ and $\vec{b}_{\text{new}}$ and solve for $\ddot{\vec{u}}^{t+\Delta t}$

Step 4 Apply the update rules to compute $\vec{u}^{t+\Delta t}$ and $\dot{\vec{u}}^{t+\Delta t}$.

Step 5 If time integration not finished then goto step 1 otherwise halt.

The time integration is unconditional stable if $\gamma \geq \frac{1}{2}$ and $\beta \geq \frac{1}{16}\left(2\gamma+1\right)^2$. The unconditional nature allows for larger time steps when the external forces are of slow time variation.

Let us take another approach. We may rewrite the first update rule as follows

$$\ddot{\vec{u}}^{t+\Delta t} = b_1\,\left(\vec{u}^{t+\Delta t} - \vec{u}^t\right) + b_2\,\dot{\vec{u}}^t + b_3\,\ddot{\vec{u}}^t \tag{2.28}$$

where

$$b_1 = \frac{1}{\beta \Delta t^2}, \tag{2.29a}$$

$$b_2 = -\frac{1}{\beta \Delta t}, \tag{2.29b}$$

$$b_3 = -\frac{\left(\frac{1}{2} - \beta\right)}{\beta}. \tag{2.29c}$$

Now we may rewrite the second update rule by substituting the result of the rewrite of the first update rule

$$\dot{\vec{u}}^{t+\Delta t} = b_4\,\left(\vec{u}^{t+\Delta t} - \vec{u}^t\right) + b_5\,\dot{\vec{u}}^t + b_6\,\ddot{\vec{u}}^t \tag{2.30}$$

where

$$b_4 = \Delta t\,\gamma\,b_1, = \frac{\gamma}{\Delta t \beta} \tag{2.31a}$$

$$b_5 = \Delta t\,\gamma\,b_2 + 1, = \left(1 - \frac{\gamma}{\beta}\right) \tag{2.31b}$$

$$b_6 = \Delta t\,\left(1 + \gamma\,b_3 - \gamma\right) = \Delta t\left(1 - \frac{\gamma}{2\beta}\right). \tag{2.31c}$$

With these modified update rules we make the substitution into the equations of motion

$$\begin{aligned}
\mathbf{M}&\left(b_1\,\left(\vec{u}^{t+\Delta t} - \vec{u}^t\right) + b_2\,\dot{\vec{u}}^t + b_3\,\ddot{\vec{u}}^t\right) \\
&+ \mathbf{C}\left(b_4\,\left(\vec{u}^{t+\Delta t} - \vec{u}^t\right) + b_5\,\dot{\vec{u}}^t + b_6\,\ddot{\vec{u}}^t\right) \\
&+ \mathbf{K}\vec{u}^{t+\Delta t} = \vec{f}.
\end{aligned} \tag{2.32}$$

We may now collect the unknown terms $\vec{u}^{t+\Delta t}$ on the left hand side to yield

$$\mathbf{A}_{\text{new2}}\vec{u}^{t+\Delta t} = \vec{b}_{\text{new2}} \tag{2.33}$$

where

$$\mathbf{A}_{\text{new2}} = \left(b_1\,\mathbf{M} + b_4\,\mathbf{C} + \mathbf{K}\right) \tag{2.34}$$

and

$$\begin{aligned}
\vec{b}_{\text{new2}} = \vec{f} + \mathbf{M}&\left(b_1\,\vec{u}^t - b_2\,\dot{\vec{u}}^t - b_3\,\ddot{\vec{u}}^t\right) \\
&+ \mathbf{C}\left(b_4\,\vec{u}^t - b_5\,\dot{\vec{u}}^t - b_6\,\ddot{\vec{u}}^t\right)
\end{aligned} \tag{2.35}$$

With the modifications in place we can now phrase the modified Newmark integration method as follows

Step 1 one is given $\vec{u}^t$ and $\dot{\vec{u}}^t$ as initial conditions

Step 2 From the equations of motion one can compute $\ddot{\vec{u}}^t$ by solving

$$\mathbf{M}\ddot{\vec{u}}^t = \vec{f} - \mathbf{C}\dot{\vec{u}}^t - \mathbf{K}\vec{u}^t \tag{2.36}$$

Step 3 Setup $\mathbf{A}_{\text{new2}}$ and $\vec{b}_{\text{new2}}$ and solve for $\vec{u}^{t+\Delta t}$

Step 4 Apply the update rules (2.28) and (2.30) to compute $\dot{\vec{u}}^{t+\Delta t}$ and $\ddot{\vec{u}}^{t+\Delta t}$.

Step 5 If time integration not finished then goto step 3 otherwise halt.

2.1.2 The Non-linear Case

Let us substitute the modified Newmark update rules into the non-linear version of the equations of motion

$$\begin{aligned}\vec{\Phi}_{\text{new}}\left(\vec{u}^{t+1}\right) \equiv \mathbf{M}&\left(b_1\ \left(\vec{u}^{t+\Delta t} - \vec{u}^t\right) + b_2\,\dot{\vec{u}}^t + b_3\,\ddot{\vec{u}}^t\right)\\ &+ \mathbf{C}\left(b_4\ \left(\vec{u}^{t+\Delta t} - \vec{u}^t\right) + b_5\,\dot{\vec{u}}^t + b_6\,\ddot{\vec{u}}^t\right)\\ &+ \vec{k}(\vec{u}^{t+\Delta t}) - \vec{f} = \vec{0}\end{aligned} \tag{2.37}$$

This is a non-linear equation and a root search problem which can be solved using a non-linear Newton method. That is In the k^{th} iteration we are seeking $\Delta\vec{u}^k$ such that we can make the Newton update

$$\vec{u}^{k+1} = \vec{u}^k + \Delta\vec{u}^k. \tag{2.38}$$

We want $\Delta\vec{u}^k$ to be such that

$$\vec{\Phi}_{\text{new}}\left(\vec{u}^{k+1}\right) = \vec{\Phi}_{\text{new}}\left(\vec{u}^k + \Delta\vec{u}^k\right) = \vec{0}. \tag{2.39}$$

Now making a first order Taylor series approximation around $\vec{u}^k$ yields

$$\vec{\Phi}_{\text{new}}\left(\vec{u}^k + \Delta\vec{u}^k\right) \approx \vec{\Phi}_{\text{new}}\left(\vec{u}^k\right) + \frac{\partial\vec{\Phi}_{\text{new}}\left(\vec{u}^k\right)}{\partial\vec{u}^{t+1}}\Delta\vec{u}^k \tag{2.40}$$

The partial derivative is given by

$$\begin{aligned}\mathbf{J}^k_{\text{new}} \equiv \frac{\partial\vec{\Phi}_{\text{new}}\left(\vec{u}^k\right)}{\partial\vec{u}^{t+1}} &= b_1\mathbf{M} + b_4\mathbf{C} + \underbrace{\frac{\partial\vec{k}\left(\vec{u}^k\right)}{\partial\vec{u}^{t+1}}}_{\mathbf{K}^k}\\ &= b_1\mathbf{M} + b_4\mathbf{C} + \mathbf{K}^k.\end{aligned} \tag{2.41}$$

Here we have used superscript to make dependence on iteration $\vec{u}^k$ more clear. Finally we can state the Newton system

$$\mathbf{J}^k_{\text{new}} \Delta \vec{u}^k = -\vec{\Phi}^k_{\text{new}}. \tag{2.42}$$

We can now put all pieces together and summarize the implicit Newmark time integration method as follows.

```
01 : Algorithm implicit-newmark-time-step($\mathbf{M}, \mathbf{C}, \vec{u}^t, \dot{\vec{u}}^t, \ldots$)
02 :   $\vec{k} \leftarrow 1$
03 :   $(\vec{u}^k, \dot{\vec{u}}^k) \leftarrow (\vec{u}^t, \dot{\vec{u}}^t)$
04 :   $\ddot{\vec{u}}^t \leftarrow \mathbf{M}^{-1}\left(\vec{f} - \vec{k}(\vec{u}^t) - \mathbf{C}\dot{\vec{u}}^t\right)$
05 :   while not converged
06 :     $\mathbf{K}^k \leftarrow$ assemble from $\frac{\partial \vec{k}^e}{\partial \vec{u}^e}$
07 :     $\mathbf{J}^k_{\text{new}} \leftarrow b_1\mathbf{M} + b_4\mathbf{C} + \mathbf{K}^k$
08 :     $\vec{r} \leftarrow -\vec{\Phi}^k_{\text{new}}$
09 :     $\Delta\vec{u}^k \leftarrow \left(\mathbf{J}^k_{\text{new}}\right)^{-1} \vec{r}$
10 :     $\tau^k \leftarrow$ do-line-search
11 :     $\vec{u}^{k+1} \leftarrow \vec{u}^k + \tau^k \Delta\vec{u}^k$
12 :     $k \leftarrow k + 1$
13 :   End while
14 :   $\vec{u}^{t+\Delta t} = \vec{u}^k$
15 :   $\dot{\vec{u}}^{t+\Delta t} = b_4\left(\vec{u}^{t+\Delta t} - \vec{u}^t\right) + b_5\,\dot{\vec{u}}^t + b_6\,\ddot{\vec{u}}^t$
16 :   $\ddot{\vec{u}}^{t+\Delta t} = b_1\left(\vec{u}^{t+\Delta t} - \vec{u}^t\right) + b_2\,\dot{\vec{u}}^t + b_3\,\ddot{\vec{u}}^t$
17 :   $(\vec{x}^{t+1}, \vec{v}^{t+1}) \leftarrow (\vec{X} + \vec{u}^k, \dot{\vec{u}}^{t+1})$
18 :   End algorithm
```

Observe we added a line-search to globalize the Newton method. The stoping criteria can be as simple as a maximum iteration count on k and a tolerance threshold on the residual $\vec{r}$. In [Erleben(2011b)] and [Erleben(2011d)] more details can be found on stopping criteria, line-search method and initial parameter values.

2.1.3 Full Implicit Backward Euler Time Integration

We may rewrite the second order differential equation into a system of coupled first order differential equations

$$\dot{\vec{x}} = \vec{v}, \tag{2.43a}$$

$$\mathbf{M}\dot{\vec{v}} + \mathbf{C}\vec{v} + \vec{k}(\vec{x} - \vec{X}) = \vec{f}. \tag{2.43b}$$

Using finite differencing we can discretize the time derivatives. In computer graphics it is often the case that the elastic forces should be treated explicitly whereas the position update should be treated implicitly as shown in [Erleben(2011a)].

For stiff materials the explicit update rule may require very small time steps to be stable. To counter this we make a full implicit scheme

$$\vec{x}^{t+\Delta t} - \vec{x}^{t} - \Delta t \vec{v}^{t+1} = \vec{0}, \tag{2.44}$$

and

$$\begin{aligned} &\mathbf{M}\left(\vec{v}^{t+\Delta t} - \vec{v}^{t}\right) + \\ &\quad \Delta t \left(\mathbf{C}\vec{v}^{t+\Delta t} + \vec{k}(\vec{x}^{t+\Delta t} - \vec{X}) - \vec{f}\right) = \vec{0} \end{aligned} \tag{2.45}$$

where superscript denotes the time. The implicit update rules above are equivalent to the root search problem

$$\vec{\Phi}(\vec{x}, \vec{v}) = \begin{bmatrix} \vec{\Phi}_x(\vec{x}, \vec{v}) \\ \vec{\Phi}_v(\vec{x}, \vec{v}) \end{bmatrix} = \vec{0}, \tag{2.46}$$

where

$$\vec{\Phi}_x(\vec{x}, \vec{v}) = \vec{x} - \vec{x}^{t} - \Delta t \vec{v}, \tag{2.47a}$$

$$\vec{\Phi}_v(\vec{x}, \vec{v}) = \left(\mathbf{M} + \Delta t \mathbf{C}\right)\vec{v} - \mathbf{M}\vec{v}^{t} - \Delta t \vec{f} + \Delta t \vec{k}(\vec{x} - \vec{X}). \tag{2.47b}$$

A solution $\vec{x}^*$ and $\vec{v}^*$ for the root search problem corresponds to the updates we are looking for. That is $\vec{x}^* = \vec{x}^{t+\Delta t}$ and $\vec{v}^* = \vec{v}^{t+\Delta t}$. A non-linear root search problem can be solved using Newtons method [Nocedal and Wright(1999)]. That is in the k^{th} iteration of the Newton method one makes the first order approximation around the k^{th} values resulting in the k^{th} Newton system

$$\begin{aligned} \vec{\Phi}(\vec{x}^{t+\Delta t}, \vec{v}^{t+\Delta t}) \approx\ & \vec{\Phi}(\vec{x}^{k}, \vec{v}^{k}) \\ & + \frac{\partial \vec{\Phi}(\vec{x}^{k}, \vec{v}^{k})}{\partial \vec{x}} \Delta \vec{x}^{k} \\ & + \frac{\partial \vec{\Phi}(\vec{x}^{k}, \vec{v}^{k})}{\partial \vec{v}} \Delta \vec{v}^{k} = 0 \end{aligned} \tag{2.48}$$

that is solved for the k^{th} Newton direction $(\Delta\vec{x}^{k}, \Delta\vec{v}^{k})$ which is then used in the k^{th} Newton update

$$\vec{x}^{k+1} = \vec{x}^{k} + \tau^{k} \Delta \vec{x}^{k}, \tag{2.49a}$$

$$\vec{v}^{k+1} = \vec{v}^{k} + \tau^{k} \Delta \vec{v}^{k}. \tag{2.49b}$$

We introduced the k^{th} step length τ^{k} that tells how far along the Newton direction one should step. A line search method can be used to determine τ^{k} this results in a damped Newton method [Nocedal and Wright(1999)]. Initially for $k = 1$ we take $\vec{x}^{k} = \vec{x}^{t}$ and $\vec{v}^{k} = \vec{v}^{t}$. The first order approximation is unlikely to yield a solution in one step, so often several iterations is taken until some sufficient stopping criteria is meet.

We can derive closed-form solutions for the Jacobians to be

$$\frac{\partial \vec{\Phi}}{\partial \vec{x}} = \begin{bmatrix} \mathbf{I} \\ \Delta t \frac{\partial \vec{k}}{\partial \vec{u}} \end{bmatrix}, \tag{2.50a}$$

$$\frac{\partial \vec{\Phi}}{\partial \vec{v}} = \begin{bmatrix} -\Delta t \mathbf{I} \\ \mathbf{M} + \Delta t \mathbf{C} \end{bmatrix}. \tag{2.50b}$$

Using our formulas for the Jacobians we can write the Newton system compactly as

$$\begin{bmatrix} \mathbf{I} & -\Delta t \mathbf{I} \\ \Delta t \frac{\partial \vec{k}}{\partial \vec{u}} & (\mathbf{M} + \Delta t \mathbf{C}) \end{bmatrix} \begin{bmatrix} \Delta \vec{x}^k \\ \Delta \vec{v}^k \end{bmatrix} = \begin{bmatrix} -\vec{\Phi}_x(\vec{x}^k, \vec{v}^k) \\ -\vec{\Phi}_v(\vec{x}^k, \vec{v}^k) \end{bmatrix}. \tag{2.51}$$

Observe that the diagonal blocks are trivially invertible so a Shur complement method [Saad(2003)] seems appropriate to solve for the Newton direction. From the first row of the Newton system we obtain

$$\Delta \vec{x}^k = \Delta t \Delta \vec{v}^k - \vec{\Phi}_x(\vec{x}^k, \vec{v}^k). \tag{2.52}$$

Kenny Says: Sifakis Siggraph 2012 notes takes a different approach on this. Clearly the Taylor series approximation for the positional parts yields $\vec{\Phi}_x^{k+1} \approx \vec{\Phi}_x^k + \frac{\partial \Phi_x^k}{\partial \vec{x}} \Delta \vec{x}^k + \frac{\partial \Phi_x^k}{\partial \vec{v}} \Delta \vec{v}^k = \vec{0}$. This combined with some rearranging yields

$$\vec{x}^k - \vec{x}^t - \Delta t \vec{v}^k + \Delta \vec{x}^k - \Delta t \Delta \vec{v}^k = \vec{0} \tag{2.53a}$$

$$\left(\vec{x}^k + \Delta \vec{x}^k\right) - \vec{x}^t - \Delta t \left(\vec{v}^k + \Delta \vec{v}^k\right) = \vec{0} \tag{2.53b}$$

$$\vec{\Phi}_x^{k+1} = \vec{0} \tag{2.53c}$$

Hence the linearization of $\vec{\Phi}_x^{k+1} = \vec{0}$ around $(\vec{x}^k, \vec{v}^k)$ gives $\vec{\Phi}_x^{k+1} \approx \vec{\Phi}_x^{k+1} = \vec{0}$. This must hold for all $k > 1$ (What about $k = 1$?). Hence we have $\vec{\Phi}_x^k = \vec{0}$, $\vec{\Phi}_x^{k-1} = \vec{0}$ and so. Now we subtract

$$\vec{\Phi}_x^{k+1} - \vec{\Phi}_x^k = \vec{0} \tag{2.54}$$

which after some simple algebra results in the relation

$$\Delta \vec{x}^k = \Delta t \Delta \vec{v}^k \tag{2.55}$$

This is different from (2.52). Why is this so? For $k = 1$ we have $\vec{\Phi}_x^1 = \vec{x}^t - \vec{x}^t - \Delta t \, \vec{v}^t = -\Delta t \, \vec{v}^t$. Clearly this will be non-zero if $\vec{v}^t$ is non-zero? Discussion: Of course one is free to pick any starting iterate value $\vec{v}^1$. Hence, one can just think of the numerical scheme as picking a suitable starting iterate $\vec{v}^1$ such that $\vec{\Phi}_x^1$ is zero.

Substitution into the second row gives the final system for the k^{th} velocity update

$$\underbrace{\left(\mathbf{M} + \Delta t \mathbf{C} + \Delta t^2 \frac{\partial \vec{k}}{\partial \vec{u}}\right)}_{\mathbf{A}} \Delta \vec{v}^k = \underbrace{\Delta t \frac{\partial \vec{k}}{\partial \vec{u}} \vec{\Phi}_x(\vec{x}^k, \vec{v}^k) - \vec{\Phi}_v(\vec{x}^k, \vec{v}^k)}_{\vec{b}}. \tag{2.56}$$

Kenny Says: Again comparing with Sifakis 2012 Siggraph notes. If we substituted $\Delta\vec{x}^k = \Delta t \Delta\vec{v}^k$ instead and solved for $\Delta\vec{x}^k$ then we would obtain the linear system

$$\underbrace{\left(\frac{1}{\Delta t^2}\mathbf{M} + \frac{1}{\Delta t}\mathbf{C} + \frac{\partial\vec{k}}{\partial\vec{u}}\right)}_{\mathbf{A}} \Delta\vec{x}^k = \underbrace{-\frac{1}{\Delta t}\vec{\Phi}_v(\vec{x}^k, \vec{v}^k)}_{\vec{b}}. \tag{2.57}$$

This is almost the same linear system as we have. Except it is divided by the $\frac{1}{\Delta t^2}$ and the term $\vec{\Phi}_x^k$ has been dropped from the right hand side vector.

If we consider the construction of the matrices in this equation then we observe that we already know how to build most terms except for the $\frac{\partial\vec{k}}{\partial\vec{u}}$ term. This term is called the tangent stiffness matrix term.

We now sketch a pseudo code version of the implicit time step method of the non-linear finite element method.

```
01 : Algorithm implicit-time-step($\mathbf{M}, \mathbf{C}, \vec{x}^t, \vec{v}^t, \ldots$)
02 :   $\vec{k} \leftarrow 1$
03 :   $(\vec{x}^k, \vec{v}^k) \leftarrow (\vec{x}^t, \vec{v}^t)$
04 :   while not converged
05 :     $\mathbf{K} \leftarrow$ assemble from $\frac{\partial\vec{k}^e}{\partial\vec{u}^e}$
06 :     $\mathbf{A} \leftarrow \mathbf{M} + \Delta t\mathbf{C} + \Delta t^2\mathbf{K}$
07 :     $\vec{b} \leftarrow \Delta t \frac{\partial\vec{k}}{\partial\vec{u}} \vec{\Phi}_x(\vec{x}^k, \vec{v}^k) - \vec{\Phi}_v(\vec{x}^k, \vec{v}^k)$
08 :     $\Delta\vec{v}^k \leftarrow \mathbf{A}^{-1}\vec{b}$
09 :     $\Delta x^k \leftarrow \Delta t \Delta\vec{v}^k - \vec{\Phi}_x(\vec{x}^k, \vec{v}^k)$
10 :     $\tau^k \leftarrow$ do-line-search
11 :     $\vec{x}^{k+1} \leftarrow \vec{x}^k + \tau^k \Delta\vec{x}^k$
12 :     $\vec{v}^{k+1} \leftarrow \vec{v}^k + \tau^k \Delta\vec{v}^k$
13 :     $k \leftarrow k + 1$
14 :   End while
15 :   $(\vec{x}^{t+1}, \vec{v}^{t+1}) \leftarrow (\vec{x}^k, \vec{v}^k)$
16 :   End algorithm
```

Observe that the global mass and damping matrices $\mathbf{M}$ and $\mathbf{C}$ can be assembled once prior to simulation whereas the global tangent stiffness matrix must be re-evaluated and re-assembled in each iteration of the Newton method (line 05). The right hand side vector $\vec{b}$ depends directly on the values $\vec{x}^k$ and $\vec{v}^k$ and must therefore be re-evaluated in each iteration (line 07). Rather than solving the velocity update (line 08) using a direct method one may apply an iterative method. If the global tangent stiffness matrix is symmetric then $\mathbf{A}$ will be a symmetric positive definite matrix and a conjugate gradient method may be used. The $\mathbf{M}$ matrix could be a good candidate for a preconditioner. This would correspond to a time constant $\mathbf{A}$ matrix with $\Delta t = 0$.

If an iterative solver is used then one often only need to evaluate the value of $\mathbf{A}$ multiplied by a vector. One does not need to assemble the $\mathbf{A}$ matrix in order to evaluate the matrix-vector product. A matrix-free solution can be used instead. This is elaborated on in Appendix A.2.

2.2 Summary

Chapter 3

Hyper Elasticity for Small and Large Deformations based on a Lagrangian Formulation

The corotational linear finite element method (FEM) is a popular choice for interactive simulations in the field of computer graphics. The method provides a nice tradeoff between computational cost and animation quality. This paper will give a rigorous derivation of this method with focus on the physical laws, and the assumptions and discretization that lie behind. The contribution lies mostly in the presentation and can hopefully be beneficial to those who wish to understand the degree of physical correctness of the method. Following this we extend the governing model to large displacements and derive a non-linear finite element method for this case. Finally, we make the connection to the finite volume method (FVM) for hyper elastic materials with large displacements.

3.1 Introduction

Deformable models cover a wide range of simulations from solids, shells, rods, wires to cloth and hair [Baraff and Witkin(1998), Bridson et al.(2002)Bridson, Fedkiw, and Anderson, Grinspun(2006), Selle et al.(2008)Selle, Lentine, and Fedkiw, Bergou et al.(2008)Bergou, Wardetzky, Robinson, Aud
The methods range from physical based to more ad hoc or geometry inspired methods [Müller et al.(2005)Müller, Heidelberger, Teschner, and Gross, Choi and Ko(2005), Botsch et al.(2006)Botsch, Pauly, Gross, and Kobbelt]. In this paper we wish to focus on fast methods that can handle real solid materials undergoing large deformations. Thus, we focus only on work relevant to physical based simulation of solids. Other interactive works are not well suited for accurate material simulation such as the versatile method and position-based physics [Teschner et al.(2004)Teschner, Heidelberger, Muller, a
Müller et al.(2007)Müller, Heidelberger, Hennix, and Ratcliff] as one can not easily model real materials with those works.

Initial work on deformable models in computer graphics were based on finite difference methods and differential geometry measures such as first and second fundamental forms [Terzopoulos et al.(1987)Terzopoulos, Platt, Barr, and Fleischer]. Green strain and Piola–Kirchhoff stress tensors are often applied for offline animation methods [Debunne et al.(2001)Debunne, Desbrun, Cani, and Barr, O'Brien and Hodgins(1999), Teran et al.(2003)Teran, Blemker, Hing, and Fedkiw, Teran et al.(2005b)Teran, Sifakis, Irving, and Fedki Barbič and James(2005), Irving et al.(2007)Irving, Schroeder, and Fedkiw, Bargteil et al.(2007)Bargteil, Whereas the corotational linear finite element method based on Cauchy strain and stress tensors is often favored for interactive animation [Müller et al.(2002)Müller, Dorsey, McMillan, Müller and Gross(2004), Galoppo et al.(2007)Galoppo, Otaduy, Tekin, Gross, and Lin, Galoppo et al.(2009)Galoppo, Otaduy, Moss, Sewall, Curtis, and Lin, Wicke et al.(2010)Wicke, Ritchie, I Zhu et al.(2010)Zhu, Sifakis, Teran, and Brandt]. It is mostly finite element methods that have been used and are in our view considered the dominating methods. Hence our focus is on this group of methods. [Sarah F. F. Gibson(1997)] contains an early survey outlining the linear elastic finite element method for small displacements. A more recent survey can be found in[Nealen et al.(2006)Nealen, Müller, Keiser, Bo which covers most of the basic theory, but only few details on the non-linear/large deformation case is given.

There are many books on the subject of continuum mechanics [Chadwick(1999), Spencer(2004), Reddy(2008)] and [Lai et al.(2009)Lai, Rubin, and Krempl] as well as books about the finite element method [Zienkiewicz and Taylor(2000), Bonet and Wood(2000), Cook et al.(2007)Cook, Malkus, Plesha, and Witt]. These books provide in depth detail on the non-linear behavior for large deformations. The learning curve for non-specialists is high and it is not always obvious how pieces and arguments fit together. This has motivated the writing of a series of technical reports addressing the usage of finite element method in computer graphics including this paper and [Erleben(2011b)].

We develop the corotational finite element method in Section 3.2 following this we extend to the non-linear case in Section 3.3 and finally show the finite volume method alternative in Section 3.4. The reader may consult Appendix A.1 for details and refreshing of mathematical details.

3.2 The Corotational Linear Elastic Finite Element Method

We have an elastic solid defined in a fixed constant reference frame called the material coordinates $\vec{X}$. The material coordinates are then deformed into spatial coordinates $\vec{x}$ by the spatial temporal deformation $\vec{x} = \Phi(\vec{X}, t)$. By convention we think of material coordinates as the spatial coordinates at time $t = 0$. That is $\vec{X} = \Phi(\vec{X}, 0)$. The material volume of the solid is denoted by V and the spatial volume by v. The boundary of the solid is denoted by ∂V and ∂v in material and spatial coordinates, respectively. Our problem is now to develop a numerical method for computing the spatial coordinates $\vec{x}$ at some given future time t knowing only the initial state at time $t = 0$.

We will solve our problem by applying a finite element method to a physical model of an elastic solid. In order to do so we must first rewrite our model problem into volume integral formulation which we convert into a weak form. Once the weak form is obtained we substitute our discretization, represented by our shape functions, into the weak form integrals. The end result of the finite element method is a time dependent matrix equation, an ordinary differential equation. Hereafter we apply finite difference method to discretize the time dependent variables. This final step leads to the final numerical method for our problem.

It may be worthwhile to refresh most of the basic math in Appendix A.1.1 prior to reading the remainder of the paper.

3.2.1 The Physical Model

From the Cauchy equation we get the continuum mechanics equivalent of Newtons second law at some spatial point $\vec{x}$

$$\rho\ddot{\vec{x}} = \nabla \cdot \sigma + \vec{b}, \tag{3.1}$$

where $\vec{b}$ represents any body force density we wish to use, ρ is spatial mass density, and σ is the Cauchy stress tensor as defined by Cauchy' stress hypothesis

$$\vec{t} = \sigma\vec{n}, \tag{3.2}$$

where $\vec{n}$ is an unit normal vector of a plane and $\vec{t}$ is the corresponding traction on the plane. Conservation of angular momentum implies that the Cauchy stress tensor is a symmetric tensor.

$$\sigma^T = \sigma. \tag{3.3}$$

We write a model of our problem as

$$\rho\ddot{\vec{x}} = \nabla \cdot \sigma + \vec{b}, \qquad \forall \vec{x} \in v, \tag{3.4a}$$

$$\vec{t} = \sigma\vec{n}, \qquad \forall \vec{x} \in \partial v_t. \tag{3.4b}$$

Here $\partial v_t \subseteq \partial v$ is the subset of the boundary were a known non-zero surface traction is applied. This could in principle be some contact stress from some interaction.

3.2.2 Weak Form Reformulation

We cast our model into a volume integral by integrating over the material volume V while multiplying our momentum equation with an admissible test function $\vec{w}$. Observe in the following that quantities are taken to be functions of material coordinates. That is $\rho(\vec{X},t)$, $\sigma(\vec{X},t)$, $\vec{b}(\vec{X},t)$, $\vec{t}(\vec{X},t)$ and so on. We refer to Appendix A.1.2 for details. On a side note if we think of $\vec{w}$ as a spatial virtual displacement then the rewrite is equivalent to stating the principle of virtual work. Our volume integral reads

$$\int_V \left(\rho\ddot{\vec{x}} - \nabla\cdot\sigma - \vec{b}\right)\cdot\vec{w}dV = 0 \tag{3.5}$$

which we split into terms

$$0 = \int_V \rho\ddot{\vec{x}}\cdot\vec{w}dV - \int_V (\nabla\cdot\sigma)\cdot\vec{w}dV - \int_V \vec{b}\cdot\vec{w}dV. \tag{3.6}$$

Next step involves the tensor equivalent of the product rule for the term $\nabla\cdot(\sigma\vec{w}) = (\nabla\cdot\sigma)\cdot\vec{w} + \sigma : \nabla\vec{w}^T$ ([1]),

$$\begin{aligned} 0 = \int_V \rho\ddot{\vec{x}}\cdot\vec{w}dV + \int_V \sigma : \nabla\vec{w}dV \\ - \int_V \nabla\cdot(\sigma\vec{w})dV - \int_V \vec{b}\cdot\vec{w}dV, \end{aligned} \tag{3.7}$$

here we used the fact that σ is symmetric. Next we apply the Gauss divergence theorem to rewrite the volume integral of $\nabla\cdot(\sigma\vec{w})$ and use (3.2),

$$\int_V \nabla\cdot(\sigma\vec{w})dV = \int_{\partial V} (\sigma\vec{w})\cdot\vec{n}dS, \tag{3.8a}$$

$$= \int_{\partial V} \vec{w}\cdot(\sigma\vec{n})dS, \tag{3.8b}$$

$$= \int_{\partial V} \vec{t}\cdot\vec{w}dS, \tag{3.8c}$$

$$= \int_{\partial V_t} \vec{t}\cdot\vec{w}dS. \tag{3.8d}$$

In the final step we have applied our boundary condition $\sigma\vec{n} = \vec{t}$ on ∂V_t so $\vec{t} = 0$ everywhere else. For convenience and readability we introduce the symbols

$$P_\rho = \int_V \rho\ddot{\vec{x}}\cdot\vec{w}dV, \tag{3.9a}$$

$$P_e = \int_V \sigma : \nabla\vec{w}dV, \tag{3.9b}$$

$$P_t = -\int_{\partial V_t} \vec{t}\cdot\vec{w}dS, \tag{3.9c}$$

$$P_b = -\int_V \vec{b}\cdot\vec{w}dV. \tag{3.9d}$$

We now have

$$0 = P_\rho + P_e + P_t + P_b. \tag{3.10}$$

[1]The double contraction between two second order tensors $\mathbf{A}$ and $\mathbf{B}$ is defined as $\mathbf{A} : \mathbf{B} = \sum_j\sum_i \mathbf{A}_{ij}\mathbf{B}_{ij}$ as explained in Appendix A.1.1

For now we will consider free floating objects so $P_t = 0$. Further, we will use a viscous damping as our body forces. This means,

$$P_b = -\int_V -c\dot{\vec{x}} \cdot \vec{w} dV = \int_V c\dot{\vec{x}} \cdot \vec{w} dV \tag{3.11}$$

where $c > 0$ is a viscous damping coefficient. The virtual elastic energy term is given as

$$P_e = \int_V \sigma : \nabla \vec{w} dV. \tag{3.12}$$

Since the Cauchy stress tensor is symmetric we have

$$P_e = \int_V \sigma : \frac{1}{2} \left(\nabla \vec{w} + \nabla \vec{w}^T \right) dV. \tag{3.13}$$

Recall the infinitesimal strain tensor is by definition given as (see Appendix A.1.3)

$$\varepsilon = \frac{1}{2} \left(\nabla \vec{u} + \nabla \vec{u}^T \right), \tag{3.14}$$

so we define

$$\varepsilon_w = \frac{1}{2} \left(\nabla \vec{w} + \nabla \vec{w}^T \right). \tag{3.15}$$

That means using the virtual infinitesimal stress tensor we have

$$P_e = \int_V \sigma : \varepsilon_w dV. \tag{3.16}$$

3.2.3 The Constitutive Equation

For linear elasticity in the infinitesimal strain limit the stress strain relation comes form $\sigma = \frac{\partial \Psi}{\partial \varepsilon}$ where Ψ is the strain energy function per unit initial volume and ε is the infinitesimal strain tensor called the Cauchy stress tensor. It is defined as

$$\varepsilon = \frac{1}{2}(\nabla \vec{u} + \nabla \vec{u}^T), \tag{3.17}$$

where $\vec{u} = \vec{x} - \vec{X}$ is known as the displacement field. In case of isotropic linear elastic (compressible Neo–Hookean) solids the strain energy function is given as

$$\Psi = \frac{\lambda}{2} \text{tr} \left(\varepsilon \right)^2 + \mu \varepsilon : \varepsilon, \tag{3.18}$$

where λ and μ are the Lamé constants. Thus, the stress strain relation has the form

$$\sigma = \lambda \text{tr} \left(\varepsilon \right) \mathbf{I} + 2\mu \varepsilon. \tag{3.19}$$

Writing out each components of the Cauchy stress tensor yields

$$\sigma_{xx} = \lambda\left(\varepsilon_{xx} + \varepsilon_{yy} + \varepsilon_{zz}\right) + 2\mu\varepsilon_{xx}, \tag{3.20a}$$

$$\sigma_{yy} = \lambda\left(\varepsilon_{xx} + \varepsilon_{yy} + \varepsilon_{zz}\right) + 2\mu\varepsilon_{yy}, \tag{3.20b}$$

$$\sigma_{zz} = \lambda\left(\varepsilon_{xx} + \varepsilon_{yy} + \varepsilon_{zz}\right) + 2\mu\varepsilon_{zz}, \tag{3.20c}$$

$$\sigma_{xy} = 2\mu\varepsilon_{xy}, \tag{3.20d}$$

$$\sigma_{xz} = 2\mu\varepsilon_{xz}, \tag{3.20e}$$

$$\sigma_{yz} = 2\mu\varepsilon_{yz}. \tag{3.20f}$$

This can be written in matrix form by defining the vectors

$$\vec{\sigma} = \begin{bmatrix}\sigma_{xx} & \sigma_{yy} & \sigma_{zz} & \sigma_{xy} & \sigma_{xz} & \sigma_{yz}\end{bmatrix}^T \tag{3.21}$$

and

$$\vec{\varepsilon} = \begin{bmatrix}\varepsilon_{xx} & \varepsilon_{yy} & \varepsilon_{zz} & 2\varepsilon_{xy} & 2\varepsilon_{xz} & 2\varepsilon_{yz}\end{bmatrix}^T \tag{3.22}$$

then

$$\begin{bmatrix}\sigma_{xx}\\ \sigma_{yy}\\ \sigma_{zz}\\ \sigma_{xy}\\ \sigma_{xz}\\ \sigma_{yz}\end{bmatrix} = \begin{bmatrix}\lambda+2\mu & \lambda & \lambda & 0 & 0 & 0\\ \lambda & \lambda+2\mu & \lambda & 0 & 0 & 0\\ \lambda & \lambda & \lambda+2\mu & 0 & 0 & 0\\ 0 & 0 & 0 & \mu & 0 & 0\\ 0 & 0 & 0 & 0 & \mu & 0\\ 0 & 0 & 0 & 0 & 0 & \mu\end{bmatrix}\begin{bmatrix}\varepsilon_{xx}\\ \varepsilon_{yy}\\ \varepsilon_{zz}\\ 2\varepsilon_{xy}\\ 2\varepsilon_{xz}\\ 2\varepsilon_{yz}\end{bmatrix}. \tag{3.23}$$

In terms of Youngs moduls E and Poisson ratio ν the Lamé constants are

$$\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}, \tag{3.24a}$$

$$\mu = \frac{E}{2(1+\nu)}. \tag{3.24b}$$

Using the above relation we get the final form of the constitutive equation that we will use,

$$\vec{\sigma} = \mathbf{D}\vec{\varepsilon} \tag{3.25}$$

where

$$\mathbf{D} = \frac{E}{(1+\nu)(1-2\nu)}\begin{bmatrix}d_0 & d_1 & d_1 & 0 & 0 & 0\\ d_1 & d_0 & d_1 & 0 & 0 & 0\\ d_1 & d_1 & d_0 & 0 & 0 & 0\\ 0 & 0 & 0 & d_2 & 0 & 0\\ 0 & 0 & 0 & 0 & d_2 & 0\\ 0 & 0 & 0 & 0 & 0 & d_2\end{bmatrix} \tag{3.26}$$

and

$$d_0 = (1-\nu), \tag{3.27a}$$

$$d_1 = \nu, \tag{3.27b}$$

$$d_2 = \frac{(1-2\nu)}{2}. \tag{3.27c}$$

3.2.4 The Finite Element Discretization

Putting it together and changing from tensor notation to matrix (Voigt) notation our weak form integrals now reads

$$P_e = \int_V \vec{\sigma} \cdot \vec{\varepsilon}_w dV = \int_V \vec{\varepsilon}_w^T \mathbf{D} \vec{\varepsilon} dV. \tag{3.28}$$

Thus, our mathematical model can be stated as

$$0 = P_\rho + P_e + P_b. \tag{3.29}$$

We have applied continuum mechanics to develop formulas for the terms above. Our final versions are,

$$P_\rho = \int_V \rho \ddot{\vec{x}} \cdot \vec{w} dV, \tag{3.30a}$$

$$P_e = \int_V \vec{\varepsilon}_w^T \mathbf{D} \vec{\varepsilon} dV, \tag{3.30b}$$

$$P_b = \int_V c \dot{\vec{x}} \cdot \vec{w} dV. \tag{3.30c}$$

Now we have reached the point where we wish to discretize our model using our shape functions.

3.2.5 Shape Functions

We let $\vec{x}$ and $\vec{u}$ be defined at the nodes of a tetrahedral mesh and use the tetrahedra as our finite elements. Notice a volume integral over the solid can be written as a sum of element wise volume integrals

$$\int_V (\ldots) dV = \sum_e \int_{V^e} (\ldots) dV. \tag{3.31}$$

We use this with out loss of generality to simplify our equations by only considering one tetrahedral element in our derivations. We used e to label the tetrahedral elements. The process represented by the above equation is known as the assembly process.

Now if we have a point $\vec{X}$ inside a tetrahedron with nodal labels i, j, k, and m then we may interpolate values at this point using linear shape functions. That is

$$\vec{X} = N_i \vec{X}_i + N_j \vec{X}_j + N_k \vec{X}_k + N_m \vec{X}_m, \tag{3.32}$$

$$\vec{u} = N_i \vec{u}_i + N_j \vec{u}_j + N_k \vec{u}_k + N_m \vec{u}_m, \tag{3.33}$$

where the N_a's for $a \in e \equiv \{i, j, k, m\}$ are the barycentric/volume coordinates of the point $\vec{X}$ with respect to the e^{th} tetrahedron. These interpolation weights are equivalent with the local hat functions. They are easily computed from geometry as

$$N_i = \frac{V_{jkm}}{V^e} \tag{3.34a}$$

$$= \frac{(\vec{X} - \vec{X}_j) \cdot ((\vec{X}_k - \vec{X}_j) \times (\vec{X}_m - \vec{X}_k))}{(\vec{X}_m - \vec{X}_i) \cdot ((\vec{X}_j - \vec{X}_i) \times (\vec{X}_k - \vec{X}_i))} \tag{3.34b}$$

where $\vec{V}^e$ is the tetrahedron material volume and V_{jkm} is the volume of the sub tetrahedron spanned by $\vec{X}$ and the face with nodes j, k, and m. In particular the material gradients of the weights are easily computed from the above formulas.

$$\nabla_0 N_i = \frac{(\vec{X}_k - \vec{X}_j) \times (\vec{X}_m - \vec{X}_k)}{6V^e}, \tag{3.35}$$

Where $\nabla_0 \equiv \frac{\partial}{\partial \vec{X}} = \begin{bmatrix} \partial_X & \partial_Y & \partial_Z \end{bmatrix}^T$ is the material gradient operator whereas $\nabla \equiv \frac{\partial}{\partial \vec{x}} = \begin{bmatrix} \partial_x & \partial_y & \partial_z \end{bmatrix}^T$ is the spatial gradient operator. Reexamining our model equation we observe that the spatial gradients are used but integrals are taken over material coordinates. Our assumption of infinitesimal displacements means $\vec{x} \approx \vec{X}$, $\parallel \nabla \vec{u} \parallel \ll 1$ and so it follows $\nabla_0 \vec{u} \approx \nabla \vec{u}$. This means that we can think of all spatial gradients as material gradients.

The idea is to replace field variables in our volume integrals with corresponding interpolation formulas. These interpolation formulas are interpolating corresponding discrete nodal values from the computational mesh. For the displacement field we have

$$\vec{u} \approx \sum_{a \in \{i,j,k,m\}} N_a \vec{u}_a = \mathbf{N}^e \begin{bmatrix} \tilde{\vec{u}}_i \\ \tilde{\vec{u}}_j \\ \tilde{\vec{u}}_k \\ \tilde{\vec{u}}_m \end{bmatrix} = \mathbf{N}^e \tilde{\vec{u}}^e. \tag{3.36}$$

Similar for $\vec{w} \approx \mathbf{N}^e \tilde{\vec{w}}^e$ the spatial positions $\vec{x} \approx \mathbf{N}^e \tilde{\vec{x}}^e$. Where

$$\mathbf{N}^e = \begin{bmatrix} N_i \mathbf{I} & N_j \mathbf{I} & N_k \mathbf{I} & N_l \mathbf{I} \end{bmatrix} \quad \text{and} \quad \mathbf{I} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.37}$$

Recall that the strain tensor was defined as $\varepsilon = \frac{1}{2}(\nabla_0 \vec{u} + \nabla_0 \vec{u}^T)$ using our vector notation and substituting our interpolation formulas

$$\vec{\varepsilon} \approx \underbrace{\begin{bmatrix} \partial_X & 0 & 0 \\ 0 & \partial_Y & 0 \\ 0 & 0 & \partial_Z \\ \partial_Y & \partial_X & 0 \\ \partial_Z & 0 & \partial_X \\ 0 & \partial_Z & \partial_Y \end{bmatrix}}_{\mathbf{S}} \mathbf{N}^e \tilde{\vec{u}}^e = \underbrace{\mathbf{S} \mathbf{N}^e}_{\mathbf{B}^e} \tilde{\vec{u}}^e = \mathbf{B}^e \tilde{\vec{u}}^e, \tag{3.38}$$

similar for the test strain $\vec{\varepsilon}_w^e \approx \mathbf{B}^e \tilde{\vec{w}}^e$. Writing out we have

$$\mathbf{B}^e = \begin{bmatrix} \mathbf{B}_i^e & \mathbf{B}_j^e & \mathbf{B}_k^e & \mathbf{B}_m^e \end{bmatrix} \tag{3.39}$$

where

$$\mathbf{B}_i^e = \begin{bmatrix} \partial_X N_i & 0 & 0 \\ 0 & \partial_Y N_i & 0 \\ 0 & 0 & \partial_Z N_i \\ \partial_Y N_i & \partial_X N_i & 0 \\ \partial_Z N_i & 0 & \partial_X N_i \\ 0 & \partial_Z N_i & \partial_Y N_i \end{bmatrix}. \tag{3.40a}$$

Similar for $\mathbf{B}_j^e$, $\mathbf{B}_k^e$, and $\mathbf{B}_m^e$. Applying all approximations we have

$$P_\rho = (\tilde{\vec{w}}^e)^T \left(\int_{V^e} \rho (\mathbf{N}^e)^T \mathbf{N}^e dV \right) \ddot{\tilde{\vec{x}}}^e, \tag{3.41a}$$

$$P_e = (\tilde{\vec{w}}^e)^T \left(\int_{V^e} (\mathbf{B}^e)^T \mathbf{D} \mathbf{B}^e dV \right) \tilde{\vec{u}}^e, \tag{3.41b}$$

$$P_b = (\tilde{\vec{w}}^e)^T \left(\int_{V^e} c (\mathbf{N}^e)^T \mathbf{N}^e dV \right) \dot{\tilde{\vec{x}}}^e, \tag{3.41c}$$

The equation $P_\rho + P_e + P_b = 0$ must hold for arbitrary values of $\tilde{\vec{w}}^e$ so we end up with

$$\mathbf{M}^e \ddot{\tilde{\vec{x}}}^e + \mathbf{C}^e \dot{\tilde{\vec{x}}}^e + \mathbf{K}^e \tilde{\vec{u}}^e = 0, \tag{3.42}$$

where

$$\mathbf{M}^e = \int_{V^e} \rho (\mathbf{N}^e)^T \mathbf{N}^e dV, \tag{3.43a}$$

$$\mathbf{C}^e = \int_{V^e} c (\mathbf{N}^e)^T \mathbf{N}^e dV, \tag{3.43b}$$

$$\mathbf{K}^e = \int_{V^e} (\mathbf{B}^e)^T \mathbf{D} \mathbf{B}^e dV. \tag{3.43c}$$

For computational efficiency one may apply mass lumping and similar for the damping matrix. Summing up over all tetrahedra, the assembly process, we have the global system

$$\mathbf{M} \ddot{\tilde{\vec{x}}} + \mathbf{C} \dot{\tilde{\vec{x}}} + \mathbf{K} \tilde{\vec{u}} = 0. \tag{3.44}$$

If we had included gravity in the body forces or similar then we may add an extra $\vec{f}$ term to the right hand side to account for this constant body force term.

Observe that the infinitesimal assumption allowed us to interchange spatial gradients with material gradients. The consequence of this is basically that the $\mathbf{K}$ matrix is a constant matrix and can be precomputed. Further, we observe what would happen if we did not employ the infinitesimal assumption then we had to use spatial gradients and $\mathbf{K}$ matrix would depend on the current spatial coordinates.

3.2.6 The Corotational Formulation

The infinitesimal assumptions implied that the element stiffness matrix $\mathbf{K}^e$ was a constant matrix. If we dropped the assumptions then the matrix would be a non-linear function of the current spatial coordinates. The elastic force term would be given by $\mathbf{K}^e(\vec{x}^e)\vec{u}^e$. The drawback is that the computational cost increases drastically. Instead we seek a compromise that can approximate the term $\mathbf{K}^e(\vec{x}^e)\vec{u}^e$ but stil have some of the computational advantages from the constant matrix version.

The idea is that large deformations are due to the current spatial rotation. To counter the large displacement effect of the spatial rotations the idea is to rotate the spatial coordinates back into an unrotated frame before computing the elastic forces. Finally the elastic forces are rotated back into the spatial coordinates. Thus, we write

$$\mathbf{K}^e(\vec{x}^e)\vec{u}^e \approx \mathbf{O}^e\mathbf{K}^e\left((\mathbf{O}^e)^T\vec{x}^e - \vec{X}^e\right), \tag{3.45}$$

where $\vec{O}^e$ is the block rotation matrix

$$\mathbf{O}^e = \begin{bmatrix} \mathbf{R}^e & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}^e & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{R}^e & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{R}^e \end{bmatrix} \tag{3.46}$$

and $\mathbf{R}^e$ is the rotation matrix that "best" describes how the material coordinates of the e^{th} tetrahedron have been transformed into their current spatial coordinates. In a least square error sense this could be understood as the problem

$$\mathbf{R}^e = \arg\min_{\mathbf{R}} \sum_{a=\{i,j,k,m\}} \| \vec{x}_a - \mathbf{R}\vec{X}_a \|^2 \tag{3.47}$$

subject to the constraint

$$\mathbf{R}^T\mathbf{R} = \mathbf{I}. \tag{3.48}$$

Rather than solving this constrained minimization problem one may exploit the polar decomposition of the deformation gradient. In the following we will omit the element superscript for readability.

The deformation gradient is defined as $\mathbf{F} = \frac{\partial \vec{x}}{\partial \vec{X}}$ and transforms material vectors to spatial vectors. Thus, we define the edge vectors

$$\vec{e}_{ab} = \vec{x}_a - \vec{x}_b, \tag{3.49a}$$

$$\vec{E}_{ab} = \vec{X}_a - \vec{X}_b, \tag{3.49b}$$

for all $a \neq b$ and $a, b \in \{i, j, k, m\}$. By definition of the deformation gradient we have

$$\vec{e}_{ab} = \mathbf{F}\vec{E}_{ab}. \tag{3.50}$$

Now we pick three linear independent edge vectors and form the linear system

$$\underbrace{\begin{bmatrix}\vec{e}_{ji} & \vec{e}_{ki} & \vec{e}_{mi}\end{bmatrix}}_{\mathbf{E}} = \mathbf{F} \underbrace{\begin{bmatrix}\vec{E}_{ji} & \vec{E}_{ki} & \vec{E}_{mi}\end{bmatrix}}_{\mathbf{E}_0}. \tag{3.51}$$

The deformation gradient is then easily computed as

$$\mathbf{F} = \mathbf{E}\mathbf{E}_0^{-1}. \tag{3.52}$$

This can be done very efficiently as $\mathbf{E}_0^{-1}$ is defined in material coordinates and can be computed and stored as a preprocessing step. There exists a polar decomposition of the deformation gradient such that

$$\mathbf{F} = \mathbf{R}\mathbf{U}, \tag{3.53}$$

where $\mathbf{R}$ is an orthogonal matrix and $\mathbf{U}$ is a symmetric stretch matrix. From the right Cauchy–Green strain tensor we have

$$\mathbf{C} = \mathbf{F}^T\mathbf{F} = \mathbf{U}^2. \tag{3.54}$$

Performing eigenvalue decomposition of $\mathbf{C}$ we find the diagonal matrix $\mathbf{\Lambda}$ of eigenvalues and $\mathbf{\Omega}$ the orthogonal matrix of eigenvectors such that

$$\mathbf{C} = \mathbf{\Omega}\mathbf{\Lambda}\mathbf{\Omega}^T \tag{3.55}$$

so

$$\mathbf{U} = \mathbf{\Omega}\sqrt{\mathbf{\Lambda}}\mathbf{\Omega}^T. \tag{3.56}$$

Finally we can compute $\mathbf{R}$ as

$$\mathbf{R} = \mathbf{F}\mathbf{U}^{-1} = \mathbf{F}\left(\sqrt{\mathbf{\Lambda}^{-1}}\right)\mathbf{\Omega}^T. \tag{3.57}$$

Before performing the velocity update by solving $\mathbf{A}\vec{v}^{t+\Delta t} = \vec{b}$ one must compute $\mathbf{R}^e$ for each tetrahedron according to the above equation and then update each local stiffness element leading to

$$\mathbf{K}^{e'} = \mathbf{O}^e\mathbf{K}^e(\mathbf{O}^e)^T, \tag{3.58a}$$

$$\vec{f}_0^{e'} = \mathbf{O}^e\mathbf{K}^e\vec{X}^e. \tag{3.58b}$$

The primed quantities are now used in place of the unprimed quantities in the assembly before computing the velocity update. Observe that a new assembly process must be done as the first thing in each iteration of the simulation loop.

3.3 The Non-linear Finite Element Method for Large Displacements

This section assumes that one is familiar with the weak form derivation (see Section 3.2.2) as it is taken as ground truth and a lot of symbols are reused here without being re-introduced.

3.3.1 The Equations of Motion

The deformation gradient is defined as

$$\mathbf{F} = \frac{\partial \vec{x}}{\partial \vec{X}} = \nabla_0 \vec{x} = \mathbf{I} + \nabla_0 \vec{u}, \tag{3.59}$$

Observe that $\mathbf{F}^{-1} = \frac{\partial \vec{X}}{\partial \vec{x}}$. The determinant of the deformation gradient is written as

$$j = \det(\mathbf{F}). \tag{3.60}$$

It is used when changing from material to spatial differential volume elements

$$dv = jdV. \tag{3.61}$$

The Cauchy equation describes the motion of the whole solid and is written in terms of the first Piola–Kirchhoff stress tensor as

$$\rho_0 \ddot{\vec{x}} = \vec{b}_0 + \nabla_0 \cdot \mathbf{P}, \qquad \forall \vec{X} \in V, \tag{3.62}$$

where ρ_0 is the material density and $\vec{b}_0$ is the material body force densities. Often one apply the boundary condition,

$$\mathbf{PN} = \vec{t}, \qquad \forall \vec{X} \in \partial V_t, \tag{3.63}$$

on the appropriate parts of ones solid surface. Here $\vec{N}$ is the outward unit material normal on the surface boundary ∂V_t and $\vec{t}$ is some prescribed surface traction. The boundary condition originates from the Cauchy stress hypothesis. The relation between the first Piola–Kirchhoff stress tensor and the Cauchy stress tensor is given as

$$\mathbf{P} = j\sigma\mathbf{F}^{-T} \tag{3.64}$$

and the second Piola–Kirchhoff tensor is related as

$$\mathbf{S} = j\mathbf{F}^{-1}\sigma\mathbf{F}^{-T}. \tag{3.65}$$

Combining those relations give us

$$\mathbf{P} = \mathbf{FS}. \tag{3.66}$$

Isotropic hyper elasticity implies that we can write the second Piola–Kirchhoff stress tensor as

$$\mathbf{S} = \frac{\partial \Psi}{\partial \mathbf{E}} = 2\frac{\partial \Psi}{\partial \mathbf{C}}, \tag{3.67}$$

where Ψ is the strain energy function and $\mathbf{C} = \mathbf{F}^T\mathbf{F}$ is the right Cauchy–Green strain tensor and $\mathbf{E} = \frac{1}{2}(\mathbf{C} - \mathbf{I})$ is the Green strain tensor.

3.3.2 The Finite Element Method

Our mathematical model can be stated as

$$0 = P_\rho + P_e + P_b. \tag{3.68}$$

We have applied continuum mechanics to develop formulas for the terms above. Our final versions using spatial integrals are,

$$P_\rho = \int_v \rho \ddot{\vec{x}} \cdot \vec{w} dv, \tag{3.69a}$$

$$P_e = \int_v \sigma : \nabla \vec{w} dv, \tag{3.69b}$$

$$P_b = \int_v c\dot{\vec{x}} \cdot \vec{w} dv. \tag{3.69c}$$

We will now develop the model further by rewriting it to use a different stress tensor than the Cauchy stress tensor. The Eulerian weak form of the elastic stress term is

$$P_e = \int_v \sigma : \nabla \vec{w} dv = \int_V \mathbf{P}\mathbf{F}^T : \nabla \vec{w} dV. \tag{3.70}$$

By the chain rule we know

$$\nabla \vec{w} = \frac{\partial \vec{w}}{\partial \vec{x}} = \frac{\partial \vec{w}}{\partial \vec{X}} \frac{\partial \vec{X}}{\partial \vec{x}} = \nabla_0 \vec{w} \mathbf{F}^{-1}, \tag{3.71}$$

so that means the Lagrangian elastic stress term can be written as

$$P_e = \int_V \mathbf{P}\mathbf{F}^T : \nabla_0 \vec{w} \mathbf{F}^{-1} dV = \int_V \mathbf{P} : \nabla_0 \vec{w} dV. \tag{3.72}$$

The discretization means we have the approximation,

$$\vec{w} \approx \sum_{a \in e} N_a(\vec{X}) \tilde{\vec{w}}_a, \tag{3.73}$$

so ([2])

$$\nabla_0 \vec{w} = \sum_{a \in e} \tilde{\vec{w}}_a \otimes \nabla_0 N_a \tag{3.74}$$

and we get

$$P_e = \int_V \mathbf{P} : \left(\sum_{a \in e} \tilde{\vec{w}}_a \otimes \nabla_0 N_a \right) dV, \tag{3.75a}$$

$$= \sum_{a \in e} \left(\tilde{\vec{w}}_a \cdot \int_V \mathbf{P} \nabla_0 N_a dV \right). \tag{3.75b}$$

[2]The tensor product is defined as follows $\left(\vec{a} \otimes \vec{b}\right) \vec{c} = \left(\vec{b} \cdot \vec{c}\right) \vec{a}$ for three given vectors $\vec{a}$, $\vec{b}$ and $\vec{c}$.

Putting it all together we know that

$$P_\rho + P_e + P_b = 0, \tag{3.76}$$

must hold for all $\tilde{\vec{w}}_a$, and further with substitution of the shape functions we can now write for the e^{th} element

$$\mathbf{M}^e \ddot{\tilde{\vec{x}}}^e + \mathbf{C}^e \dot{\tilde{\vec{x}}}^e + \vec{k}^e(\tilde{\vec{u}}^e) = \vec{f}^e \tag{3.77}$$

where

$$\mathbf{M}^e = \int_{V^e} \rho_0 (\mathbf{N}^e)^T \mathbf{N}^e dV, \tag{3.78a}$$

$$\mathbf{C}^e = \int_{V^e} c_0 (\mathbf{N}^e)^T \mathbf{N}^e dV, \tag{3.78b}$$

$$\vec{k}^e_a(\tilde{\vec{u}}^e) = \int_{V^e} \mathbf{P} \nabla_0 N_a dV, \tag{3.78c}$$

$$\vec{k}^e = \begin{bmatrix} \vec{k}^e_i \\ \vec{k}^e_j \\ \vec{k}^e_k \\ \vec{k}^e_m \end{bmatrix}. \tag{3.78d}$$

In the last equation above we have with out loss of generality assumed that the e^{th} element has four nodes labeled $e \equiv \{i, j, k, m\}$. This corresponds to a tetrahedral mesh. Further we have assumed that the $\vec{f}^e$-vector is coming from some constant body forces like gravity. Observe that the inertia and damping terms are the same as in the corotational linear elasticity finite element method we derived previously only the stiffness term is different.

After performing the assembly we have the differential equation to time integrate

$$\mathbf{M}\ddot{\tilde{\vec{x}}} + \mathbf{C}\dot{\tilde{\vec{x}}} + \vec{k}(\tilde{\vec{u}}) = \vec{f}. \tag{3.79}$$

3.4 The Finite Volume Method

In this section we will derive the method from [Teran et al.(2003)Teran, Blemker, Hing, and Fedkiw] using our notation and conventions. As before the Cauchy equation written in terms of the first Piola–Kirchhoff stress tensor as

$$\rho_0 \ddot{\vec{x}} = \vec{b}_0 + \nabla_0 \cdot \mathbf{P}, \qquad \forall \vec{X} \in V, \tag{3.80}$$

with the boundary condition

$$\mathbf{P}\mathbf{N} = \vec{t}, \qquad \forall \vec{X} \in \partial V_t. \tag{3.81}$$

The relation to the Cauchy stress tensor is given as

$$\mathbf{P} = j \sigma \mathbf{F}^{-T} \tag{3.82}$$

and the second Piola–Kirchhoff tensor is related as

$$\mathbf{S} = j\mathbf{F}^{-1}\sigma\mathbf{F}^{-T}. \tag{3.83}$$

Combining those relations give us

$$\mathbf{P} = \mathbf{F}\mathbf{S}. \tag{3.84}$$

We will use a tetrahedral mesh for our discretization. We will use a median dual cell vertex control volume to develop our finite volume method. Let us integrate the equation of motion over the volume of the control volume for the i^{th} vertex,

$$\int_{V^i} \rho_0 \ddot{\vec{x}} dV = \int_{V^i} \vec{b}_0 dV + \int_{V^i} \nabla_0 \cdot \mathbf{P} dV. \tag{3.85}$$

We will assume that body forces are constant over the control volumes (like gravity) and by applying the mid-point rule and Leibniz integral rule,

$$V^i \rho_0 \ddot{\vec{x}}_i = V^i \vec{b}_{0,i} + \int_{V^i} \nabla_0 \cdot \mathbf{P} dV. \tag{3.86}$$

For the last term we will apply the Gauss divergence theorem

$$\int_{V^i} \nabla_0 \cdot \mathbf{P} dV = \int_{\partial V^i} \mathbf{P}\vec{N} dS. \tag{3.87}$$

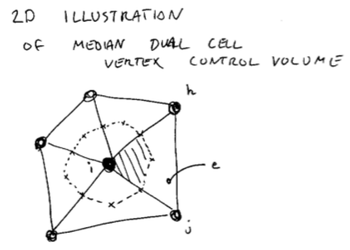
Next we will examine the surface integral in more detail by breaking it down into sub-parts. Let the index set of tetrahedrons that share the i^{th} vertex be written as $\mathcal{N}(i)$. Then

$$\int_{\partial V^i} \mathbf{P}\vec{N} dS = \sum_{e \in \mathcal{N}(i)} \int_{\partial V_e^i} \mathbf{P}\vec{N} dS, \tag{3.88}$$

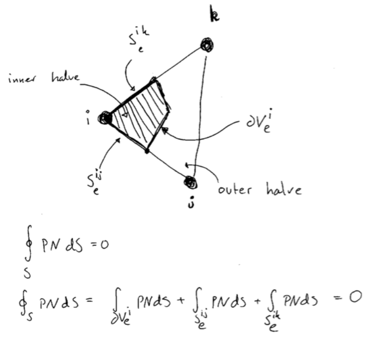
where ∂V_e^i is the sub-part of the surface of the i^{th} control volume that is contained in the e^{th} tetrahedron as illustrated in Figure 3.1(b).

We will now develop an efficient method for computing the sub-part of the surface integral coming from ∂V_e^i. The idea is to create a closed surface over the e^{th} tetrahedron by patching ∂V_e^i with the faces of the e^{th} tetrahedron. Now without loss of generality we write the vertex indices of the e^{th} tetrahedron as $\{i, j, k, m\}$. Observe that the surface ∂V_e^i intersects the e^{th} tetrahedron in two parts. As a convenience we introduce the surface notation S_e^{ijm} to mean the contained surface sub-part of the triangle face defined by the mesh vertex positions $\vec{x}_i$, $\vec{x}_j$ and $\vec{x}_m$ as illustrated in Figure 3.1(c). The closed surface of the inside part of the tetrahedron containing the i^{th} vertex can now be written as the union of the surfaces ∂V_e^i, S_e^{ikj}, S_e^{imk} and S_e^{ijm}. If we assume that the stress tensor $\mathbf{P}$ is constant over the e^{th} tetrahedron then we have

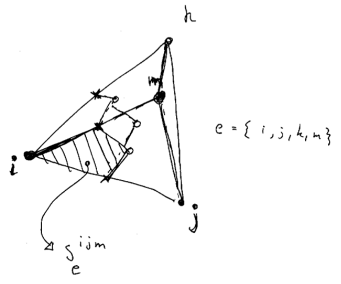
$$\begin{aligned}\int_{\partial V_e^i} \mathbf{P}\vec{N} dS + \int_{S_e^{ikj}} \mathbf{P}\vec{N} dS \\ + \int_{S_e^{imk}} \mathbf{P}\vec{N} dS + \int_{S_e^{ijm}} \mathbf{P}\vec{N} dS = \vec{0}.\end{aligned} \tag{3.89}$$



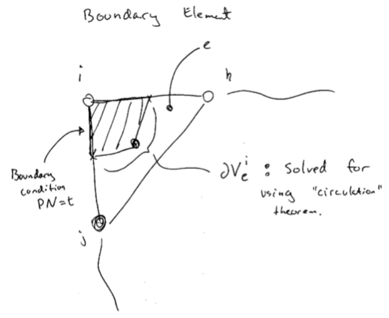
(a)


k
S_e^{ik}
inner halve
i
∂V_e^i
S_e^{ij}
j
outer halve
$\oint_S PN\,dS = 0$
$\oint_S PN\,dS = \int_{\partial V_e^i} PN\,dS + \int_{S_e^{ij}} PN\,dS + \int_{S_e^{ik}} PN\,dS = 0$

(b)



(c)



(d)

Figure 3.1: Illustrations showing how the e^{th} sub-part of the surface integral of the i^{th} volume is computed. (a) illustrates a 2D median dual vertex cell control volume.(b) shows how one element from the vertex cell control volume are divided into two parts for the 2D case. (c) illustrates the corresponding 3D case of (b). Finally (d) illustrates how the boundary conditions are used for control volumes on the boundary.

Because any closed surface integral over a constant value is always zero. When dealing with boundary control volumes such as the one illustrated in 3.1(d) one must apply the boundary condition $\mathbf{PN} = \vec{t}$ on the boundary part of the surface integral. This corresponds to adding traction forces along the boundary surface.

Now realizing that a sub-surface is always planar means that the unit outward normal of S_e^{ijk} is always constant. Let the area of the triangle given by the vertex i, k, and j be written as $A^{ikj} = \frac{1}{2} \parallel (\vec{x}_k - \vec{x}_i) \times (\vec{x}_j - \vec{x}_i) \parallel$ then the outward unit normal of S_e^{ikj} is

$$\vec{N}_e^{ikj} = \frac{(\vec{x}_k - \vec{x}_i) \times (\vec{x}_j - \vec{x}_i)}{2A^{ikj}}. \tag{3.90}$$

Using all this we have

$$\begin{aligned}\int_{\partial V_e^i} \mathbf{P}\vec{N} dS = \\ -\frac{1}{3}\mathbf{P}\left(\vec{N}_e^{ikj}A^{ikj} + \vec{N}_e^{imk}A^{imk} + \vec{N}_e^{ijm}A^{ijm}\right).\end{aligned} \tag{3.91}$$

Putting it all together we obtain the second order differential equation

$$m_i \ddot{\vec{x}}_i = \vec{f}_i - \vec{k}, \tag{3.92}$$

where $m_i = V^i \rho_0$ is the total mass of the i^{th} control volume (this corresponds to a lumped mass matrix) and the elastic forces are

$$\vec{k} = \frac{1}{3} \sum_{e \in \mathcal{N}(i)} \mathbf{P}^e \left(\vec{N}_e^{ikj}A^{ikj} + \vec{N}_e^{imk}A^{imk} + \vec{N}_e^{ijm}A^{ijm}\right) \tag{3.93}$$

where $\mathbf{P}^e$ is the constant stress of the e^{th} tetrahedron and $\vec{f}_i = V^i \vec{b}_{0,i}$ is the control volume body forces. The factor $\frac{1}{3}$ comes from the fact that we used a median dual cell vertex control volume.

We saw previously for the corotational formulation that for linear shape functions on a tetrahedral mesh one may compute the deformation gradient as

$$\mathbf{F}^e = \mathbf{E}\mathbf{E}_0^{-1}. \tag{3.94}$$

This is constant over the e^{th} tetrahedron. Therefore the right Cauchy–Green strain tensor $\mathbf{C}^e = (\mathbf{F}^e)^T(\mathbf{F}^e)$ will be a constant over the tetrahedral element and further the Green strain tensor is constant and given by $\mathbf{E}^e = \frac{1}{2}(\mathbf{C}^e - \mathbf{I})$. The constant element stress tensor is then obtained from some constitutive law as

$$\mathbf{S}^e = \frac{\partial \Psi}{\partial \mathbf{E}^e}. \tag{3.95}$$

For the particular case of Saint Venant–Kirchhoff materials we have

$$\mathbf{S}^e = \lambda \text{tr}(\mathbf{E}^e)\mathbf{I} + 2\mu\mathbf{E}^e. \tag{3.96}$$

Observe that the choice of linear shape functions is consistent with our assumption of constant stress tensors.

When we rewrite the second order differential equation into a coupled first order system then we may time discretize the coupled system using a semi implicit Euler scheme,

$$\vec{v}_i^{t+\Delta t} = \vec{v}_i^t + \frac{\Delta t}{m_i}\left(\vec{f} - \vec{k}^t\right), \tag{3.97a}$$

$$\vec{x}_i^{t+\Delta t} = \vec{x}_i^t + \Delta t \vec{v}_i^{t+\Delta t}. \tag{3.97b}$$

Observe the elastic forces are treated explicit and therefor $\mathbf{F}^e$ and subsequently $\mathbf{P}^e = \mathbf{F}^e\mathbf{S}^e$ are computed based on $\vec{x}^t$ values. This has the added benefit of not needing to evaluate the Jacobian of the elastic forces which would be needed if an implicit velocity update was wanted.

3.5 Summary

We derived the mathematical foundation for the corotational linear finite element method. Our result is given by (3.42) with discrete matrices shown in (3.43). Finite difference approximations lead us to the velocity and position update rules given by (2.7) and (2.8). Finally adding the corotational formulation resulted in a rotation given by (3.57) which are used to update the stiffness terms using (3.58).

Hereafter we considered large displacements and derived a non-linear finite element method in (3.77) with discrete matrices given by (3.78). We showed two different time discretizations that lead to an explicit and an implicit scheme as shown in (2.10) and (2.11). The implicit scheme was only mentioned and full details are in [Erleben(2011b)].

Finally, we derived a finite volume method resulting in an elastic force term given by (3.93) and a semi-implicit time integration update rule given by (3.97).

Chapter 4

The Tangent Stiffness Matrix for the Implicit Non-linear Finite Element Method

Hyper elastic materials under large deformations is easily modeled using non-linear finite element analysis. Time discretization of the resulting linear finite element model results in initial value problems given by coupled first order differential systems. We have derived an showed this in previous work [Erleben(2011a)].

When an implicit method is used the tangent stiffness matrix must be evaluated. In previous work we omitted detailed formulas for evaluating the tangent stiffness matrix. This is the topic of this technical report.

4.1 Introduction

For convenience we have restated a short version of our derivation of the non-linear finite element method in Section 4.2 where we derive a formula for computing the elastic forces on a node a from a finite element e as

$$\vec{k}_a^e(\vec{u}) = \int_{V^e} \mathbf{P}\nabla_0 N_a dV. \tag{4.1}$$

where V^e is the volume of the e^{th} element. $\mathbf{P}$ is the first Piola–Kirchhoff stress tensor and N_a is the material shape function associated with the a^{th} node and $\vec{u}$ is the displacement field.

We assume without loss of generality that we have a tetrahedral mesh. Then our elements are tetrahedra and consist of four nodes. Thus, we write $e \equiv \{i, j, k, m\}$ and $a \in e$. For any material point $\vec{X}$ from the e^{th} element we make the approximations

$$\vec{u}(\vec{X}) \approx \sum_{c\in e} N_c(\vec{X})\vec{u}_c \tag{4.2}$$

and

$$\nabla_0\vec{u}(\vec{X}) \approx \sum_{c\in e} \vec{u}_c \otimes \nabla_0 N_c(\vec{X}) \tag{4.3}$$

where $\nabla = \frac{\partial}{\partial \vec{x}}$ is the spatial gradient and $\nabla_0 = \frac{\partial}{\partial \vec{X}}$ the material gradient.

The problem we address in this paper is simply that of finding a formula for evaluating the term

$$\frac{\partial \vec{k}_a^e}{\partial \vec{u}_b} = \frac{\partial}{\partial \vec{u}_b}\int_{V^e} \mathbf{P}\nabla_0 N_a dV, \tag{4.4}$$

for any $b \in e$. This is a 3-by-3 matrix telling us how node b as seen from e influences the node a as seen from e. Thus, it is a sub-block of the local tangent stiffness matrix $\mathbf{K}^e$ defined as,

$$\mathbf{K}^e = \frac{\partial \vec{k}^e}{\partial \vec{u}^e} = \begin{bmatrix} \frac{\partial \vec{k}_i^e}{\partial \vec{u}_i} & \frac{\partial \vec{k}_i^e}{\partial \vec{u}_j} & \frac{\partial \vec{k}_i^e}{\partial \vec{u}_k} & \frac{\partial \vec{k}_i^e}{\partial \vec{u}_m} \\ \frac{\partial \vec{k}_j^e}{\partial \vec{u}_i} & \frac{\partial \vec{k}_j^e}{\partial \vec{u}_j} & \frac{\partial \vec{k}_j^e}{\partial \vec{u}_k} & \frac{\partial \vec{k}_j^e}{\partial \vec{u}_m} \\ \frac{\partial \vec{k}_k^e}{\partial \vec{u}_i} & \frac{\partial \vec{k}_k^e}{\partial \vec{u}_j} & \frac{\partial \vec{k}_k^e}{\partial \vec{u}_k} & \frac{\partial \vec{k}_k^e}{\partial \vec{u}_m} \\ \frac{\partial \vec{k}_m^e}{\partial \vec{u}_i} & \frac{\partial \vec{k}_m^e}{\partial \vec{u}_j} & \frac{\partial \vec{k}_m^e}{\partial \vec{u}_k} & \frac{\partial \vec{k}_m^e}{\partial \vec{u}_m} \end{bmatrix}. \tag{4.5}$$

To assemble the whole tangent stiffness matrix one must iterative over all elements and add them up appropriately. For completeness Appendix A.2 outlines an assembly process.

4.2 The Non-linear Finite Element Method

We will briefly present a derivation of the non-linear finite element method for hyper elastic materials under large deformations.

4.2.1 The Governing and Constitutive Equations

We will start by stating the elastic model as a partial differential equation. We will use common continuum mechanics definitions and terms as found in many textbooks[Chadwick(1999), Bonet and Wood(2000), Spencer(2004), Reddy(2008)] and [Lai et al.(2009)Lai, Rubin, and Krempl]. The deformation gradient is defined as

$$\mathbf{F} = \frac{\partial \vec{x}}{\partial \vec{X}} = \nabla_0 \vec{x} = \mathbf{I} + \nabla_0 \vec{u}, \tag{4.6}$$

where $\vec{x}$ is current spatial position, $\vec{X}$ the corresponding material position and $\vec{u} = \vec{x} - \vec{X}$ is the displacement field. The gradient operator $\nabla_0 \equiv \frac{\partial}{\partial \vec{X}}$ is the material gradient. Observe that $\mathbf{F}^{-1} = \frac{\partial \vec{X}}{\partial \vec{x}}$. The determinant of the deformation gradient is written as

$$j = \det(\mathbf{F}). \tag{4.7}$$

It is used when changing from material to spatial differential volume elements

$$dv = jdV. \tag{4.8}$$

The Cauchy equation written in terms of the first Piola–Kirchhoff stress tensor is

$$\rho_0 \ddot{\vec{x}} = \vec{b}_0 + \nabla_0 \cdot \mathbf{P}, \qquad \forall \vec{X} \in V \tag{4.9}$$

where ρ_0 is the material mass density and $\vec{b}_0$ is material body density forces. The boundary condition is

$$\mathbf{PN} = \vec{t}, \qquad \forall \vec{X} \in \partial V_t. \tag{4.10}$$

where $\vec{N}$ is the outward unit material normal on the material surface boundary ∂V_t and $\vec{t}$ is some pre-scribed surface traction. The relation to the Cauchy stress tensor σ is given as

$$\mathbf{P} = j\sigma\mathbf{F}^{-T} \tag{4.11}$$

and the second Piola–Kirchhoff tensor $\mathbf{S}$ is related as

$$\mathbf{S} = j\mathbf{F}^{-1}\sigma\mathbf{F}^{-T}. \tag{4.12}$$

Combining those relations give us

$$\mathbf{P} = \mathbf{FS}. \tag{4.13}$$

Isotropic hyper elasticity implies that we can write the second Piola–Kirchhoff stress tensor as

$$\mathbf{S} = \frac{\partial \Psi}{\partial \mathbf{E}} = 2\frac{\partial \Psi}{\partial \mathbf{C}}, \tag{4.14}$$

where Ψ is the strain energy function and $\mathbf{C} = \mathbf{F}^T\mathbf{F}$ is the right Cauchy–Green strain tensor and $\mathbf{E} = \frac{1}{2}(\mathbf{C} - \mathbf{I})$ is the Green strain tensor.

4.2.2 The Finite Element Method

The model is derived by multiplying the Cauchy equation by an admissible test function $\vec{w}$, taking the spatial volume integral and rewriting the integrals into weak form. The resulting mathematical model can be stated as

$$0 = P_\rho + P_e + P_v, \tag{4.15}$$

where the weak form spatial integrals are given as,

$$P_\rho = \int_v \rho \ddot{\vec{x}} \cdot \vec{w} dv, \tag{4.16a}$$

$$P_e = \int_v \sigma : \nabla \vec{w} dv, \tag{4.16b}$$

$$P_v = \int_v c\dot{\vec{x}} \cdot \vec{w} dv. \tag{4.16c}$$

Here P_ρ is the inertia term, P_e the elastic stress term and P_v a viscous damping term. Without loss of generality we have omitted terms for other body forces and surface traction. We will now develop the model further by rewriting it to use a different stress tensor than the Cauchy stress tensor. The spatial weak form of the elastic stress term is rewritten into the material frame

$$P_e = \int_v \sigma : \nabla \vec{w} dv = \int_V \mathbf{P}\mathbf{F}^T : \nabla \vec{w} dV. \tag{4.17}$$

By the chain rule we know

$$\nabla \vec{w} = \frac{\partial \vec{w}}{\partial \vec{x}} = \frac{\partial \vec{w}}{\partial \vec{X}} \frac{\partial \vec{X}}{\partial \vec{x}} = \nabla_0 \vec{w} \mathbf{F}^{-1}, \tag{4.18}$$

so that means the material elastic stress term can be written

$$P_e = \int_V \mathbf{P}\mathbf{F}^T : \nabla_0 \vec{w} \mathbf{F}^{-1} dV = \int_V \mathbf{P} : \nabla_0 \vec{w} dV. \tag{4.19}$$

The discretization means we have the approximation

$$\vec{w} \approx \sum_c N_c(\vec{X}) \vec{w}_c, \tag{4.20}$$

so

$$\nabla_0 \vec{w} = \sum_c \vec{w}_c \otimes \nabla_0 N_c \tag{4.21}$$

where c is the nodal (or quadrature point) indices of the whole computational mesh and $N_c(\vec{X})$ is the material shape function associated with the c node. The same approximations can be used for any other fields like the displacement field $\vec{u}$. Substitution of our approximations give,

$$P_e = \int_V \mathbf{P} : \left(\sum_c \vec{w}_c \otimes \nabla_0 N_c \right) dV, \tag{4.22a}$$

$$= \sum_c \left(\vec{w}_c \cdot \int_V \mathbf{P} \nabla_0 N_c dV \right). \tag{4.22b}$$

Putting it all together we know that

$$P_\rho + P_e + P_v = 0, \tag{4.23}$$

must hold for all $\vec{w}_c$, and further with substitution of approximations for $\vec{x}$ and $\vec{u}$ we can write for the e^{th} element

$$\mathbf{M}^e \ddot{\vec{x}}^e + \mathbf{C}^e \dot{\vec{x}}^e + \vec{k}^e(\vec{u}^e) = \vec{f}^e. \tag{4.24}$$

If we with out loss of generality assume that the e^{th} element has four nodes labeled $e \equiv \{i, j, k, m\}$ which corresponds to a tetrahedral mesh then the terms in the equation of the e^{th} element is given as

$$\vec{u}^e = \begin{bmatrix} \vec{u}^e_i \\ \vec{u}^e_j \\ \vec{u}^e_k \\ \vec{u}^e_m \end{bmatrix}, \tag{4.25a}$$

$$\vec{x}^e = \begin{bmatrix} \vec{x}^e_i \\ \vec{x}^e_j \\ \vec{x}^e_k \\ \vec{x}^e_m \end{bmatrix}, \tag{4.25b}$$

and

$$\mathbf{N}^e = \begin{bmatrix} N_i\mathbf{I} & N_j\mathbf{I} & N_k\mathbf{I} & N_l\mathbf{I} \end{bmatrix}, \tag{4.26a}$$

$$\mathbf{M}^e = \int_{V^e} \rho_0 (\mathbf{N}^e)^T \mathbf{N}^e dV, \tag{4.26b}$$

$$\mathbf{C}^e = \int_{V^e} c_0 (\mathbf{N}^e)^T \mathbf{N}^e dV, \tag{4.26c}$$

$$\vec{k}^e_c(\vec{u}^e) = \int_{V^e} \mathbf{P}\nabla_0 N_c dV, \tag{4.26d}$$

$$\vec{k}^e = \begin{bmatrix} \vec{k}^e_i \\ \vec{k}^e_j \\ \vec{k}^e_k \\ \vec{k}^e_m \end{bmatrix}. \tag{4.26e}$$

We have assumed that the $\vec{f}^e$-vector is coming from some constant body forces like gravity. Observe that the inertia and damping terms are the same as in the corotational linear elasticity finite element method derived in [Erleben(2011a)]. After performing the assembly we have the differential equation to time integrate

$$\mathbf{M}\ddot{\vec{x}} + \mathbf{C}\dot{\vec{x}} + \vec{k}(\vec{u}) = \vec{f}. \tag{4.27}$$

4.3 The Straightforward Calculus Approach

For sake of readability we introduce the notation $\vec{g} = \nabla_0 N$ and note that the material shape function gradients are independent of the displacement field $\vec{u}$. We will consider the scalar element $\mathbb{I}, \mathbb{K}$ of the local tangent stiffness matrix and use Leibniz rule to finally obtain

$$\frac{\partial \vec{k}^e_{a,\mathbb{I}}}{\partial \vec{u}_{b,\mathbb{K}}} = \frac{\partial}{\partial \vec{u}_{b,\mathbb{K}}} \int_{V^e} \sum_{\mathbb{J}=1}^{3} \left(\mathbf{P}_{\mathbb{I},\mathbb{J}} \vec{g}_{a,\mathbb{J}}\right) dV, \tag{4.28a}$$

$$= \sum_{\mathbb{J}=1}^{3} \int_{V^e} \frac{\partial}{\partial \vec{u}_{b,\mathbb{K}}} \left(\mathbf{P}_{\mathbb{I},\mathbb{J}} \vec{g}_{a,\mathbb{J}}\right) dV, \tag{4.28b}$$

$$= \sum_{\mathbb{J}=1}^{3} \int_{V^e} \left(\frac{\partial \mathbf{P}_{\mathbb{I},\mathbb{J}}}{\partial \vec{u}_{b,\mathbb{K}}}\right) \vec{g}_{a,\mathbb{J}} dV. \tag{4.28c}$$

From this we observe that we are seeking a formula for $\frac{\partial \mathbf{P}_{\mathbb{I},\mathbb{J}}}{\partial \vec{u}_{b,\mathbb{K}}}$.

The first Piola–Kirchhoff stress tensor is by definition a function of the deformation gradient which again is a function of the displacement field so we write $\mathbf{P}(\mathbf{F}(\mathbf{u}))$. From definitions we have,

$$\mathbf{F} = \mathbf{I} + \nabla_0 \vec{u}, \tag{4.29a}$$

$$\mathbf{P} = \frac{\partial \Psi(\mathbf{F})}{\partial \mathbf{F}}, \tag{4.29b}$$

where $\Psi(\mathbf{F})$ is the strain energy function which is either given in closed form or measured from experiments. By the chain rule we find,

$$\frac{\partial \mathbf{P}_{\mathbb{I},\mathbb{J}}}{\partial \vec{u}_{b,\mathbb{K}}} = \frac{\partial \frac{\partial \Psi}{\partial \mathbf{F}_{\mathbb{I},\mathbb{J}}}}{\partial \vec{u}_{b,\mathbb{K}}} = \sum_{\mathbb{L}=1}^{3} \sum_{\mathbb{M}=1}^{3} \frac{\partial^2 \Psi}{\partial \mathbf{F}_{\mathbb{L},\mathbb{M}} \partial \mathbf{F}_{\mathbb{I},\mathbb{J}}} \frac{\partial \mathbf{F}_{\mathbb{L},\mathbb{M}}}{\partial \vec{u}_{b,\mathbb{K}}}. \tag{4.30}$$

One usually write the fourth order elasticity tensor $\mathcal{A}$ for the first Piola–Kirchhoff stress tensor as

$$\mathcal{A}_{\mathbb{I},\mathbb{J},\mathbb{L},\mathbb{M}} \equiv \frac{\partial^2 \Psi}{\partial \mathbf{F}_{\mathbb{L},\mathbb{M}} \partial \mathbf{F}_{\mathbb{I},\mathbb{J}}} = \mathcal{A}_{\mathbb{L},\mathbb{M},\mathbb{I},\mathbb{J}}. \tag{4.31}$$

Given a constitutive model for Ψ one can derive closed form solutions for computing $\mathcal{A}$. Thus, we now have,

$$\frac{\partial \mathbf{P}_{\mathbb{I},\mathbb{J}}}{\partial \vec{u}_{b,\mathbb{K}}} = \sum_{\mathbb{L}=1}^{3} \sum_{\mathbb{M}=1}^{3} \mathcal{A}_{\mathbb{I},\mathbb{J},\mathbb{L},\mathbb{M}} \frac{\partial \mathbf{F}_{\mathbb{L},\mathbb{M}}}{\partial \vec{u}_{b,\mathbb{K}}}. \tag{4.32}$$

If we substitue $\nabla_0 \vec{u}(\vec{X}) \approx \sum_{c \in e} \vec{u}_c \otimes \nabla_0 N_c(\vec{X})$ into $\mathbf{F} = \mathbf{I} + \nabla_0 \vec{u}$ then we obtain,

$$\mathbf{F}_{\mathbb{L},\mathbb{M}} \approx \delta_{\mathbb{L},\mathbb{M}} + \sum_{c \in e} \vec{u}_{c,\mathbb{L}} \vec{g}_{c,\mathbb{M}}, \tag{4.33}$$

where $\delta_{\mathbb{L},\mathbb{M}}$ is the Kronecker delta function defined as

$$\delta_{\mathbb{L},\mathbb{M}} = \begin{cases} 1 & \text{if } \mathbb{L} = \mathbb{M} \\ 0 & \text{otherwise} \end{cases}. \tag{4.34}$$

From this we have

$$\frac{\partial \mathbf{F}_{\mathbb{L},\mathbb{M}}}{\partial \vec{u}_{b,\mathbb{K}}} = \delta_{\mathbb{L},\mathbb{K}} \vec{g}_{b,\mathbb{M}}. \tag{4.35}$$

Putting it all together we have the final closed form solution

$$\frac{\partial \mathbf{P}_{\mathbb{I},\mathbb{J}}}{\partial \vec{u}_{b,\mathbb{K}}} = \sum_{\mathbb{L}=1}^{3} \sum_{\mathbb{M}=1}^{3} \mathcal{A}_{\mathbb{I},\mathbb{J},\mathbb{L},\mathbb{M}} \left(\delta_{\mathbb{L},\mathbb{K}} \nabla_0 N_{b,\mathbb{M}} \right) \tag{4.36}$$

and

$$\begin{aligned} \frac{\partial \vec{k}^e_{a,\mathbb{I}}}{\partial \vec{u}_{b,\mathbb{K}}} = \sum_{\mathbb{J}=1}^{3} \int_{V^e} \Bigg(\sum_{\mathbb{L}=1}^{3} \sum_{\mathbb{M}=1}^{3} \mathcal{A}_{\mathbb{I},\mathbb{J},\mathbb{L},\mathbb{M}} \\ \left(\delta_{\mathbb{L},\mathbb{K}} \nabla_0 N_{b,\mathbb{M}} \right) \Bigg) \nabla_0 N_{a,\mathbb{J}} dV \end{aligned} \tag{4.37}$$

which is conveniently reduced to

$$\frac{\partial \vec{k}^e_{a,\mathbb{I}}}{\partial \vec{u}_{b,\mathbb{K}}} = \sum_{\mathbb{J}=1}^{3} \sum_{\mathbb{M}=1}^{3} \int_{V^e} \mathcal{A}_{\mathbb{I},\mathbb{J},\mathbb{K},\mathbb{M}} \nabla_0 N_{b,\mathbb{M}} \nabla_0 N_{a,\mathbb{J}} dV. \tag{4.38}$$

From the symmetry of $\mathcal{A}$ it follows that

$$\mathcal{A}_{\mathbb{I},\mathbb{J},\mathbb{K},\mathbb{M}} \nabla_0 N_{b,\mathbb{M}} \nabla_0 N_{a,\mathbb{J}} = \mathcal{A}_{\mathbb{K},\mathbb{M},\mathbb{I},\mathbb{J}} \nabla_0 N_{b,\mathbb{M}} \nabla_0 N_{a,\mathbb{J}} \tag{4.39}$$

and since we are free to swap the indices $\mathbb{M}$ and $\mathbb{J}$ we find that

$$\frac{\partial \vec{k}^e_{a,\mathbb{I}}}{\partial \vec{u}_{b,\mathbb{K}}} = \frac{\partial \vec{k}^e_{a,\mathbb{K}}}{\partial \vec{u}_{b,\mathbb{I}}}. \tag{4.40}$$

That is the local tangent stiffness matrices are symmetric and hence the global tangent stiffness matrix is symmetric too.

4.4 The Bonet and Wood Approach

In [Bonet and Wood(2000)] a different approach is used for deriving the tangent stiffness matrix. Here the authors use the directional derivative to make the first order Taylor approximation. As an example let $\vec{H}$ be some arbitrary vector function then

$$\vec{H}(\vec{x} + \vec{u}) \approx \vec{H}(\vec{x}) + \frac{\partial \vec{H}(\vec{x})}{\partial \vec{u}} \vec{u} \equiv \vec{H}(\vec{x}) + D\vec{H}(\vec{x})[\vec{u}] \tag{4.41}$$

where $D\vec{H}(\vec{x})[\vec{u}]$ denotes the directional derivative of $\vec{H}$ at the point $\vec{x}$ in the direction $\vec{u}$. Another difference is that the formulas for the tangent stiffness matrix is derived before substitution of the finite element approximations. In our approach substitutions were made before the derivatives were computed. However, both approaches lead to the same formulas.

We wish to compute the directional derivative of $\mathbf{P} : \nabla_0 \vec{w}$. Using the product rule we have,

$$D\left(\mathbf{P} : \nabla_0 \vec{w}\right)[\vec{u}] = D\mathbf{P}[\vec{u}] : \nabla_0 \vec{w} + \mathbf{P} : D\nabla_0 \vec{w}[\vec{u}]. \tag{4.42}$$

The test function $\vec{w}$ is an arbitrary constant therefore we have $D\nabla_0 \vec{w}[\vec{u}] = \mathbf{0}$ and so

$$D\left(\mathbf{P} : \nabla_0 \vec{w}\right)[\vec{u}] = D\mathbf{P}[\vec{u}] : \nabla_0 \vec{w}. \tag{4.43}$$

By definition we know $\mathbf{P} = \frac{\partial \Psi}{\partial \mathbf{F}}$ so using the chain rule we find

$$D\mathbf{P}_{\mathbb{I},\mathbb{J}}[\vec{u}] = \sum_{\mathbb{K}=1}^{3} \sum_{\mathbb{M}=1}^{3} \frac{\partial^2 \Psi}{\partial \mathbf{F}_{\mathbb{K},\mathbb{M}} \partial \mathbf{F}_{\mathbb{I},\mathbb{J}}} D\mathbf{F}_{\mathbb{K},\mathbb{M}}[\vec{u}]. \tag{4.44}$$

We define the fourth order elasticity tensor $\mathcal{A}$ as

$$\mathcal{A}_{\mathbb{K},\mathbb{M},\mathbb{I},\mathbb{J}} = \mathcal{A}_{\mathbb{I},\mathbb{J},\mathbb{K},\mathbb{M}} \equiv \frac{\partial^2 \Psi}{\partial \mathbf{F}_{\mathbb{K},\mathbb{M}} \partial \mathbf{F}_{\mathbb{I},\mathbb{J}}}. \tag{4.45}$$

Using our definition we may now write

$$D\mathbf{P}[\vec{u}] = \mathcal{A} : D\mathbf{F}[\vec{u}]. \tag{4.46}$$

Finally we observe that $D\mathbf{F}[\vec{u}] = \nabla_0 \vec{u}$ so putting it all together we find

$$D\left(\mathbf{P} : \nabla_0 \vec{w}\right)[\vec{u}] = \nabla_0 \vec{w} : \mathcal{A} : \nabla_0 \vec{u}. \tag{4.47}$$

Using the approximations $\nabla_0 \vec{w} \approx \sum_a \vec{w}_a \otimes \nabla_0 N_a$ and $\nabla_0 \vec{u} \approx \sum_b \vec{u}_b \otimes \nabla_0 N_b$ we find

$$\begin{aligned} \nabla_0 \vec{w} :& \mathcal{A} : \nabla_0 \vec{u} \approx \\ & \sum_a \sum_b (\vec{w}_a \otimes \nabla_0 N_a) : \mathcal{A} : (\vec{u}_b \otimes \nabla_0 N_b). \end{aligned} \tag{4.48}$$

Next we use the rules that for any second order tensor $\mathbf{B}$ and vectors $\vec{v}$ and $\vec{w}$ we have $(\vec{w} \otimes \vec{v}) : \mathbf{B} = \mathbf{B} : (\vec{w} \otimes \vec{v}) = \vec{w} \cdot \mathbf{B}\vec{v}$ so

$$\begin{aligned} (\vec{w}_a \otimes \nabla_0 N_a) : (\mathcal{A} : (\vec{u}_b \otimes \nabla_0 N_b)) = \\ \vec{w}_a \cdot \left((\mathcal{A} : (\vec{u}_b \otimes \nabla_0 N_b))\, \nabla_0 N_a \right). \end{aligned} \tag{4.49}$$

Next we change to index notation and use the definition of the double contraction so the right hand side becomes,

$$\sum_{\mathbb{I}=1}^{3} \sum_{\mathbb{J}=1}^{3} \sum_{\mathbb{K}=1}^{3} \sum_{\mathbb{M}=1}^{3} \vec{w}_{a,\mathbb{I}} \mathcal{A}_{\mathbb{I},\mathbb{J},\mathbb{K},\mathbb{M}} \nabla_0 N_{b,\mathbb{M}} \nabla_0 N_{a,\mathbb{J}} \vec{u}_{b,\mathbb{K}} \tag{4.50}$$

We may rearrange the order of terms into

$$\sum_{\mathbb{I}=1}^{3} \vec{w}_{a,\mathbb{I}} \sum_{\mathbb{K}=1}^{3} \left(\sum_{\mathbb{J}=1}^{3} \sum_{\mathbb{M}=1}^{3} \mathcal{A}_{\mathbb{I},\mathbb{J},\mathbb{K},\mathbb{M}} \nabla_0 N_{b,\mathbb{M}} \nabla_0 N_{a,\mathbb{J}} \right) \vec{u}_{b,\mathbb{K}}. \tag{4.51}$$

Finally we take the volume integral over the element

$$\vec{w}_a \cdot \mathbf{K}^e_{a,b}\vec{u}_b = \sum_{\mathbb{I}=1}^{3} \vec{w}_{a,\mathbb{I}} \left(\sum_{\mathbb{K}=1}^{3} \frac{\partial \vec{k}^e_{a,\mathbb{I}}}{\partial \vec{u}_{b,\mathbb{K}}} \vec{u}_{b,\mathbb{K}} \right) \tag{4.52}$$

where

$$\frac{\partial \vec{k}^e_{a,\mathbb{I}}}{\partial \vec{u}_{b,\mathbb{K}}} = \sum_{\mathbb{J}=1}^{3}\sum_{\mathbb{M}=1}^{3} \int_{V^e} \mathcal{A}_{\mathbb{I},\mathbb{J},\mathbb{K},\mathbb{M}} \nabla_0 N_{b,\mathbb{M}} \nabla_0 N_{a,\mathbb{J}} dV \tag{4.53}$$

If we compare our final result to (4.38) we observe that the formulas are identical.

4.5 Computing the Elasticity Tensor $\mathcal{A}$

We have derived a closed form solution for the local tangent stiffness matrix that requires us to compute the fourth order elasticity tensor $\mathcal{A}$ defined as

$$\mathcal{A}_{\mathbb{K},\mathbb{M},\mathbb{I},\mathbb{J}} = \mathcal{A}_{\mathbb{I},\mathbb{J},\mathbb{K},\mathbb{M}} \equiv \frac{\partial^2 \Psi}{\partial \mathbf{F}_{\mathbb{K},\mathbb{M}} \partial \mathbf{F}_{\mathbb{I},\mathbb{J}}}. \tag{4.54}$$

Any second order tensor can be written as a double contraction of a fourth order tensor and a second order tensor. We will try to exploit this knowledge to find an easy formula for $\mathcal{A}$. From $\mathbf{P} = \mathbf{F}\mathbf{S}$ we have

$$\mathbf{P}_{\mathbb{I},\mathbb{J}} = \sum_{\mathbb{M}=1}^{3} \mathbf{F}_{\mathbb{I},\mathbb{M}} \mathbf{S}_{\mathbb{M},\mathbb{J}} = \sum_{\mathbb{K}=1}^{3}\sum_{\mathbb{M}=1}^{3} \delta_{\mathbb{I},\mathbb{K}} \mathbf{S}_{\mathbb{M},\mathbb{J}} \mathbf{F}_{\mathbb{K},\mathbb{M}} \tag{4.55}$$

From this it appears that if we define the fourth order tensor $\mathcal{S}_{\mathbb{I},\mathbb{J},\mathbb{K},\mathbb{M}} = \delta_{\mathbb{I},\mathbb{K}} \mathbf{S}_{\mathbb{M},\mathbb{J}}$ then

$$\mathbf{P} = \mathcal{S} : \mathbf{F} \tag{4.56}$$

In general $\mathcal{S}$ depends on $\mathbf{F}$. For those special materials where $\mathcal{S}$ is a constant we have $\mathcal{A} = \mathcal{S}$.

The cases where one knows the strain energy function Ψ as a function of $\mathbf{F}$, are trivially dealt with using straightforward calculus. In some cases Ψ is given in terms of the Green strain tensor $\mathbf{E}$. For instance for Saint Venant–Kirchhoff model we have

$$\Psi(\mathbf{E}) = \frac{\lambda}{2} \left(\operatorname{tr}(\mathbf{E})\right)^2 + \mu \mathbf{E} : \mathbf{E} \tag{4.57}$$

where λ and μ are the Lamé coefficients. It follows that

$$\mathbf{S} = \frac{\partial \Psi(\mathbf{E})}{\partial \mathbf{E}} = \lambda \operatorname{tr}(\mathbf{E}) \mathbf{I} + 2\mu \mathbf{E}. \tag{4.58}$$

There are other cases where the right Cauchy–Green strain tensor are used to parameterize the strain energy function or the eigenvalues of the right Cauchy–Green strain tensor.

$$\mathbf{S} = 2\frac{\partial \Psi(\mathbf{C})}{\partial \mathbf{C}}, \tag{4.59a}$$

$$\mathbf{S} = \sum_{\mathbb{A}=1}^{3} 2\frac{\partial \Psi(\mathbf{\Lambda})}{\partial \lambda_{\mathbb{A}}^2}\left(\vec{v}_{\mathbb{A}} \otimes \vec{v}_{\mathbb{A}}\right), \tag{4.59b}$$

where $\vec{v}_{\mathbb{A}}$ is the $\mathbb{A}^{\text{th}}$ eigenvector for the eigenvalue $\lambda_{\mathbb{A}}^2$ so we have $\mathbf{C} = \mathbf{\Omega}\mathbf{\Lambda}\mathbf{\Omega}^T$ where

$$\mathbf{\Lambda} = \begin{bmatrix} \lambda_1^2 & 0 & 0 \\ 0 & \lambda_2^2 & 0 \\ 0 & 0 & \lambda_3^2 \end{bmatrix} \tag{4.60}$$

and

$$\mathbf{\Omega} = \begin{bmatrix} \vec{v}_1 & \vec{v}_2 & \vec{v}_3 \end{bmatrix}. \tag{4.61}$$

If we can compute $\mathbf{S}$ then we can always compute $\mathbf{P} = \mathbf{F}\mathbf{S}$. For the elasticity tensor $\mathcal{A}$ it seems hopeless. We may compute

$$\begin{aligned}
\mathcal{A}_{\mathbb{I},\mathbb{J},\mathbb{K},\mathbb{M}} &= \frac{\partial \mathbf{P}_{\mathbb{K},\mathbb{M}}}{\partial \mathbf{F}_{\mathbb{I},\mathbb{J}}}, && (4.62a)\\
&= \sum_{\mathbb{R}} \frac{\partial\left(\mathbf{F}_{\mathbb{K},\mathbb{R}}\mathbf{S}_{\mathbb{R},\mathbb{M}}\right)}{\partial \mathbf{F}_{\mathbb{I},\mathbb{J}}}, && (4.62b)\\
&= \sum_{\mathbb{R}} \frac{\partial \mathbf{F}_{\mathbb{K},\mathbb{R}}}{\partial \mathbf{F}_{\mathbb{I},\mathbb{J}}}\mathbf{S}_{\mathbb{R},\mathbb{M}} + \sum_{\mathbb{R}} \mathbf{F}_{\mathbb{K},\mathbb{R}}\frac{\partial \mathbf{S}_{\mathbb{R},\mathbb{M}}}{\partial \mathbf{F}_{\mathbb{I},\mathbb{J}}}, && (4.62c)\\
&= \delta_{\mathbb{I},\mathbb{K}}\mathbf{S}_{\mathbb{J},\mathbb{M}} + \sum_{\mathbb{R}} \mathbf{F}_{\mathbb{K},\mathbb{R}}\frac{\partial \mathbf{S}_{\mathbb{R},\mathbb{M}}}{\partial \mathbf{F}_{\mathbb{I},\mathbb{J}}}. && (4.62d)
\end{aligned}$$

Thus, we find ourselves looking for a formula for a $\frac{\partial \mathbf{S}_{\mathbb{R},\mathbb{M}}}{\partial \mathbf{F}_{\mathbb{I},\mathbb{J}}}$ term,

$$\frac{\partial \mathbf{S}_{\mathbb{R},\mathbb{M}}}{\partial \mathbf{F}_{\mathbb{I},\mathbb{J}}} = \sum_{\mathbb{V}=1}^{3}\sum_{\mathbb{W}=1}^{3} \frac{\partial \mathbf{S}_{\mathbb{R},\mathbb{M}}}{\partial \mathbf{C}_{\mathbb{V},\mathbb{W}}}\frac{\partial \mathbf{C}_{\mathbb{V},\mathbb{W}}}{\partial \mathbf{F}_{\mathbb{I},\mathbb{J}}}, \tag{4.63a}$$

$$\frac{\partial \mathbf{S}_{\mathbb{R},\mathbb{M}}}{\partial \mathbf{F}_{\mathbb{I},\mathbb{J}}} = \sum_{\mathbb{V}=1}^{3}\sum_{\mathbb{W}=1}^{3} \frac{\partial \mathbf{S}_{\mathbb{R},\mathbb{M}}}{\partial \mathbf{E}_{\mathbb{V},\mathbb{W}}}\frac{\partial \mathbf{E}_{\mathbb{V},\mathbb{W}}}{\partial \mathbf{F}_{\mathbb{I},\mathbb{J}}}. \tag{4.63b}$$

Here the terms $\mathcal{C} = \frac{\partial S}{\partial E} = 2\frac{\partial S}{\partial C}$ can be found from closed form solutions. So we are only seeking formulas for $\frac{\partial \mathbf{E}}{\partial \mathbf{F}}$ and $\frac{\partial \mathbf{C}}{\partial \mathbf{F}}$. From definition we have $\mathbf{E} = \frac{1}{2}\left(\mathbf{C} - \mathbf{I}\right)$ so we find

$$\frac{\partial \mathbf{E}_{\mathbb{V},\mathbb{W}}}{\partial \mathbf{F}_{\mathbb{I},\mathbb{J}}} = \frac{1}{2}\frac{\partial \mathbf{C}_{\mathbb{V},\mathbb{W}}}{\partial \mathbf{F}_{\mathbb{I},\mathbb{J}}}. \tag{4.64}$$

We know from definition that $\mathbf{C} = \mathbf{F}^T\mathbf{F}$ so

$$\frac{\partial \mathbf{C}_{\mathbb{V},\mathbb{W}}}{\partial \mathbf{F}_{\mathbb{I},\mathbb{J}}} = \sum_{\mathbb{L}=1}^{3} \left(\frac{\partial \mathbf{F}_{\mathbb{L},\mathbb{V}}}{\partial \mathbf{F}_{\mathbb{I},\mathbb{J}}} \mathbf{F}_{\mathbb{L},\mathbb{W}} + \mathbf{F}_{\mathbb{L},\mathbb{V}} \frac{\partial \mathbf{F}_{\mathbb{L},\mathbb{W}}}{\partial \mathbf{F}_{\mathbb{I},\mathbb{J}}} \right), \tag{4.65a}$$

$$= \sum_{\mathbb{L}=1}^{3} \left(\delta_{\mathbb{L},\mathbb{I}} \delta_{\mathbb{V},\mathbb{J}} \mathbf{F}_{\mathbb{L},\mathbb{W}} + \delta_{\mathbb{L},\mathbb{I}} \delta_{\mathbb{W},\mathbb{J}} \mathbf{F}_{\mathbb{L},\mathbb{V}} \right), \tag{4.65b}$$

$$= \delta_{\mathbb{V},\mathbb{J}} \mathbf{F}_{\mathbb{I},\mathbb{W}} + \delta_{\mathbb{W},\mathbb{J}} \mathbf{F}_{\mathbb{I},\mathbb{V}}. \tag{4.65c}$$

Let us define the fourth order tensor $\mathcal{J}_{\mathbb{V},\mathbb{W},\mathbb{I}\mathbb{J}} = \delta_{\mathbb{V},\mathbb{J}} \mathbf{F}_{\mathbb{I},\mathbb{W}} + \delta_{\mathbb{W},\mathbb{J}} \mathbf{F}_{\mathbb{I},\mathbb{V}}$ then we may combine our derivations into

$$\begin{aligned} \mathcal{A}_{\mathbb{I},\mathbb{J},\mathbb{K},\mathbb{M}} &= \delta_{\mathbb{I},\mathbb{K}} \mathbf{S}_{\mathbb{J},\mathbb{M}} \\ &\quad + \frac{1}{2} \sum_{\mathbb{R}} \mathbf{F}_{\mathbb{K},\mathbb{R}} \sum_{\mathbb{W}} \sum_{\mathbb{V}} \mathcal{C}_{\mathbb{R},\mathbb{M},\mathbb{V},\mathbb{W}} \mathcal{J}_{\mathbb{V},\mathbb{W},\mathbb{I},\mathbb{J}}. \end{aligned} \tag{4.66}$$

To deal with the case of eigenvalue parameterizations we compute the fourth order elasticity tensor $\mathcal{C}$ in terms of the right Cauchy–Green strain censor as

$$\mathcal{C}_{\mathbb{I},\mathbb{J},\mathbb{K},\mathbb{M}} = 2 \frac{\partial \mathbf{S}_{\mathbb{I},\mathbb{J}}}{\partial \mathbf{C}_{\mathbb{K},\mathbb{M}}} \tag{4.67a}$$

$$= 2 \sum_{\mathbb{A}=1}^{3} \sum_{\mathbb{B}=1}^{3} 2 \frac{\partial^2 \Psi(\mathbf{\Lambda})}{\partial \lambda_{\mathbb{B}}^2 \partial \lambda_{\mathbb{A}}^2} \frac{\partial \lambda_{\mathbb{B}}^2}{\partial \mathbf{C}_{\mathbb{K},\mathbb{M}}} \left(\vec{v}_{\mathbb{A}} \otimes \vec{v}_{\mathbb{A}} \right)_{\mathbb{I},\mathbb{J}}. \tag{4.67b}$$

Now from $\mathbf{\Lambda} = \mathbf{\Omega}^T \mathbf{C} \mathbf{\Omega}$ we find

$$\lambda_{\mathbb{B}}^2 = \vec{v}_{\mathbb{B}} \cdot \mathbf{C} \vec{v}_{\mathbb{B}}. \tag{4.68}$$

Letting the Euclidean basis be given by the orthonormal vectors $\vec{e}_1$, $\vec{e}_2$ and $\vec{e}_3$ then $\mathbf{C} = \sum_{\mathbb{I}=1}^{3} \sum_{\mathbb{J}=1}^{3} \mathbf{C}_{\mathbb{I},\mathbb{J}} \left(\vec{e}_{\mathbb{I}} \otimes \vec{e}_{\mathbb{J}} \right)$ and we find

$$\lambda_{\mathbb{B}}^2 = \vec{v}_{\mathbb{B}} \cdot \left(\sum_{\mathbb{I}=1}^{3} \sum_{\mathbb{J}=1}^{3} \mathbf{C}_{\mathbb{I},\mathbb{J}} \left(\vec{e}_{\mathbb{I}} \otimes \vec{e}_{\mathbb{J}} \right) \vec{v}_{\mathbb{B}} \right). \tag{4.69}$$

From the above it follows that

$$\frac{\partial \lambda_{\mathbb{B}}^2}{\partial \mathbf{C}_{\mathbb{K},\mathbb{M}}} = \vec{v}_{\mathbb{B}} \cdot \left(\vec{e}_{\mathbb{K}} \otimes \vec{e}_{\mathbb{M}} \right) \vec{v}_{\mathbb{B}}, \tag{4.70a}$$

$$= \left(\vec{v}_{\mathbb{B}} \cdot \vec{e}_{\mathbb{M}} \right) \left(\vec{v}_{\mathbb{B}} \cdot \vec{e}_{\mathbb{K}} \right), \tag{4.70b}$$

$$= \left(\vec{v}_{\mathbb{B}} \otimes \vec{v}_{\mathbb{B}} \right)_{\mathbb{K},M}. \tag{4.70c}$$

Since $\left(\vec{v}_{\mathbb{A}} \otimes \vec{v}_{\mathbb{A}} \right)_{\mathbb{I},\mathbb{J}} \left(\vec{v}_{\mathbb{B}} \otimes \vec{v}_{\mathbb{B}} \right)_{\mathbb{K},\mathbb{M}} = \left(\vec{v}_{\mathbb{A}} \otimes \vec{v}_{\mathbb{A}} \otimes \vec{v}_{\mathbb{B}} \otimes \vec{v}_{\mathbb{B}} \right)_{\mathbb{I},\mathbb{J},\mathbb{K},\mathbb{M}}$ we finally have

$$\mathcal{C} = \sum_{\mathbb{A},\mathbb{B}} 4 \frac{\partial^2 \Psi(\mathbf{\Lambda})}{\partial \lambda_{\mathbb{A}}^2 \partial \lambda_{\mathbb{B}}^2} \mathcal{N}_{\mathbb{A},\mathbb{A},\mathbb{B},\mathbb{B}}, \tag{4.71}$$

where

$$\mathcal{N}_{\mathbb{A},\mathbb{A},\mathbb{B},\mathbb{B}} = \vec{v}_{\mathbb{A}} \otimes \vec{v}_{\mathbb{A}} \otimes \vec{v}_{\mathbb{B}} \otimes \vec{v}_{\mathbb{B}} \tag{4.72}$$

Having computed $\mathcal{C}$ we can use the formula in (4.66) to obtain $\mathcal{A}$.

4.5.1 Material Examples

Here follows some examples of common strain energy functions: volume preservation, Riemannian elasticity, Saint Venant–Kirchhoff material, and Ogden model.

$$\Psi_{\text{vol}} = \left(\prod_{\mathbb{A}} \lambda_{\mathbb{A}}^{\frac{1}{2}} - 1 \right)^2 , \tag{4.73a}$$

$$\Psi_{\text{rie}} = \frac{\mu}{4} \sum_{\mathbb{A}} \log^2 \lambda_{\mathbb{A}} + \frac{\lambda}{8} \left(\sum_{\mathbb{A}} \log \lambda_{\mathbb{A}} \right)^2 , \tag{4.73b}$$

$$\Psi_{\text{svk}} = \frac{\mu}{4} \sum_{\mathbb{A}} \left(\lambda_{\mathbb{A}}^{\frac{1}{2}} - 1 \right)^2 + \frac{\lambda}{8} \left(\sum_{\mathbb{A}} \left(\lambda_{\mathbb{A}}^{\frac{1}{2}} - 1 \right) \right)^2 , \tag{4.73c}$$

$$\Psi_{\text{ogden}} = \sum_{\mathbb{P}=1}^{N} \frac{\mu_{\mathbb{P}}}{\alpha_{\mathbb{P}}} \left(\lambda_1^{\frac{\alpha_{\mathbb{P}}}{2}} + \lambda_2^{\frac{\alpha_{\mathbb{P}}}{2}} + \lambda_3^{\frac{\alpha_{\mathbb{P}}}{2}} - 3 \right) , \tag{4.73d}$$

where μ and λ are the Lamé constants and $\alpha_{\mathbb{P}}$ and $\mu_{\mathbb{P}}$ are often determined experimentally.

In [McAdams et al.(2011)McAdams, Zhu, Selle, Empey, Tamstorf, Teran, and Sifakis] a corotational elastic strain energy is presented as

$$\Psi_{\text{corot}} = \mu \parallel \mathbf{F} - \mathbf{R} \parallel_F^2 + \frac{\lambda}{2} \left(\text{tr} \left(\mathbf{R}^T \mathbf{F} - \mathbf{I} \right) \right)^2 , \tag{4.74}$$

where μ and λ are the Lamé constants, $\mathbf{F}$ is the deformation gradient and $\mathbf{R}$ is the rotation tensor from the polar decomposition $\mathbf{F} = \mathbf{R}\mathbf{U}$.

4.6 Summary

In this paper we treated the implicit time integration scheme of the non-linear finite element method we introduced in [Erleben(2011a)]. The result was a root search problem (5.1) that could be solved iteratively. In each iteration a velocity update is computed by (2.57) and used to find a position update (2.52). In order to compute the velocity update one must compute the tangent stiffness matrix for which we have derived a closed form formula given by (4.38). The formula uses a fourth order elasticity censor $\mathcal{A}$ that is not always easily obtained. To deal with this we derived a conversion formula (4.66) that allows one to compute $\mathcal{A}$ from the material elasticity tensor $\mathcal{C}$. For the cases where $\mathcal{C}$ is not known we have shown how to obtain it knowing only the principal stretches of the material (4.71).

Chapter 5

Implementation Tricks and Tips for Finite Element Modeling of Hyper Elastic Materials

In a series of papers we have introduced the finite element method for hyper elastic deformable models undergoing large deformations. Although we have derived pseudo code of the algorithms we have introduced throughout our writings we have largely omitted implementation specific details. In this final work on hyper elastic models we shift aim and focus on implementation tips and tricks.

We have chosen to use Matlab as the example programming language to allow the casual reader to focus more on principles and ideas rather than programming language specifics. This provide the readers with a cross-platform and easy up to run code snippets. Our points generalize to other programming languages as well such as NumPhy, SciPhy or even C and C++.

Finally, we compare our own implementations of the various methods in particular we investigate performance as well as mesh and time step convergence studies.

5.1 Introduction

In previous work we have attacked the most wide spread accurate physics-based simulation methods for hyper elastic materials used in the field of computer graphics [Erleben(2011c), Erleben(2011a), Erleben(2011b)]. Our approach has so far been mathematical and theoretical with great focus on the proper arguments and stating the physical assumptions used throughout the discretization processes. In this final paper we wish to close our work on these hyper elastic simulation methods by turning our attention towards the more practical matters such as how to implement various difficult parts of these methods. To make the knowledge more accessible to the readers we supplement our work with an Open Source project [Erleben(2011e)] covering all the methods and variations we have covered in our series of papers.

The details we cover here is on large part the details of the numerical method for the implicit non-linear finite element method. This may be found in Section 5.2. When setting up a simulation one needs to specify proper boundary conditions. We will explain in Section 5.3 how one may introduce fixed position type of boundary conditions. In simulations one wishes to examine what happens under different external loading. Thus, in Section 5.4 we derive the necessary formalism needed to apply arbitrary surface traction to all the methods. Following this we analyze what kind of preconditioner one should apply to the iterative solvers in Section 5.5. Even though our work has put much focus on getting the numerical methods correct one should not omit the importance of a good quality mesh. Thus, in Section 5.6 we introduce various quality measures that are useful for inspecting ones mesh. Before our conclusion in Section 5.8 we will investigate numerical properties and compare the different methods in Section 5.7.

5.2 The Non-linear Newton Method

The implicit time stepping of the non-linear finite element method is given by solutions to the root search problem

$$\Phi(\vec{x}, \vec{v}) = \vec{0}. \tag{5.1}$$

We apply an iterative Newton method that produces a sequence of iterates $\{\vec{x}^k, \vec{z}^k\}_{k=1}^{\infty}$ that hopefully converges towards a solution $\vec{x}^*, \vec{v}^*$ defined by $\Phi(\vec{x}^*, \vec{v}^*) = \vec{0}$. To keep notation simple we write Φ^k to mean $\Phi(\vec{x}^k, \vec{v}^k)$ and similar for other functions taking our iterates as arguments. Thus, in general it holds that $\Phi^k \neq 0$ otherwise the k^{th} iterate would be a solution to the problem. We define the gradient operator ∇ as follows

$$\nabla \equiv \begin{bmatrix} \frac{\partial}{\partial \vec{x}} & \frac{\partial}{\partial \vec{v}} \end{bmatrix} \tag{5.2}$$

Observe that if this is applied to a scalar function ψ^k then $\nabla\psi^k$ is a row vector and not a column vector. The Jacobian of the root search function Φ at the k^{th} iterate is

$$\begin{aligned} \mathbf{J}^k = \nabla\Phi^k &= \begin{bmatrix} \frac{\partial \Phi^k}{\partial \vec{x}} & \frac{\partial \Phi^k}{\partial \vec{v}} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{I} & -\Delta t \mathbf{I} \\ \Delta t \mathbf{K} & (\mathbf{M} + \Delta t \mathbf{C}) \end{bmatrix}. \end{aligned} \tag{5.3}$$

Hirota [Hirota(2002)] presented an implicit method for a non-linear finite element method which is similar to our formulation using implicit time stepping. Hirota used a two point predictor-corrector to generate initial iterates for his

Newton method. Incremental loading was applied and a backtracking line–search method that guarantees non inverted tetrahedra. Our approach differs in that we use a different initial iterate and a different line–search method.

In Section 5.2.1 we address the issue of whether the Newton system is singular or not. Then in Section 5.2.2 we will prove that an exact Newton direction always exist and even in case only an approximate Newton direction is computed one can ensure that the Newton direction is always a descent direction.

5.2.1 Solving the Newton System

The Newton direction is a solution to the Newton system

$$-\Phi^k = \mathbf{J}^k \begin{bmatrix} \Delta\vec{x}^k \\ \Delta\vec{v}^k \end{bmatrix}. \tag{5.4}$$

The question we have is whether we always can find a Newton direction. This means we need to consider if the Jacobian is non-singular. We immediately observe that both diagonal blocks of the Jacobian are symmetric positive definite matrices. This means that both diagonal blocks are always non-singular. From this it follows that the Jacobian is always non-singular.

If we apply a Shur Complement method [Saad(2003)] then from the first row of our Newton system we obtain

$$\Delta\vec{x}^k = \Delta t \Delta\vec{v}^k - \Phi^k_{\vec{x}}. \tag{5.5}$$

From the second row we have

$$-\Phi^k_{\vec{v}} = \Delta t \mathbf{K} \Delta\vec{x}^k + (\mathbf{M} + \Delta t \mathbf{C})\, \Delta\vec{v}^k. \tag{5.6}$$

Substituting the first equation we find

$$\underbrace{\left(\mathbf{M} + \Delta t \mathbf{C} + \Delta t^2 \mathbf{K}\right)}_{\mathbf{A}} \Delta\vec{v}^k = \underbrace{\Delta t \mathbf{K} \Phi^k_{\vec{x}} - \Phi^k_{\vec{v}}}_{\vec{b}}. \tag{5.7}$$

Recall $\mathbf{M}$ and $\mathbf{C}$ are symmetric positive definite matrices and $\mathbf{K}$ is a symmetric matrix (not necessarily positive definite). Thus, the $\mathbf{A}$-matrix is a symmetric positive definite matrix and as such a preconditioned conjugate gradient (PCG) method will be suitable for obtaining an approximation for $\Delta\vec{v}^k$. Once $\Delta\vec{v}^k$ has been computed we may compute $\Delta\vec{x}^k$ using the equation we obtained from the first row of the Newton system.

In our implementation we use the full system from (5.4) rather than the Shur reduced system in (5.7). Observe that the Shur reduced system can be solved efficiently using PCG. The full Newton system is non-symmetric although non-singular so here generalized minimum residual (GMRES) method is a good candidate.

5.2.2 Is the Newton Direction a Descent Direction?

The natural merit function is defined by

$$\psi(\vec{x},\vec{v}) \equiv \frac{1}{2} \parallel \Phi(\vec{x},\vec{v}) \parallel^2 = \frac{1}{2}\Phi(\vec{x},\vec{v})^T\Phi(\vec{x},\vec{v}). \tag{5.8}$$

From calculus we find the gradient of merit function to be

$$\nabla\psi^k = \left(\Phi^k\right)^T \mathbf{J}^k. \tag{5.9}$$

Observe that from the properties of $\mathbf{J}^k$ that $\nabla\psi^k = \vec{0}^T$ if and only if $\Phi^k = \vec{0}$. In this case we already have $\psi^k = 0$ implying that the k^{th} iterate is a solution of the root search problem. The directional derivative in the Newton direction is given by

$$D\psi^k = \nabla\psi^k \begin{bmatrix}\Delta\vec{x}^k\\ \Delta\vec{v}^k\end{bmatrix}, \tag{5.10a}$$

$$= \left(\Phi^k\right)^T \mathbf{J}^k \begin{bmatrix}\Delta\vec{x}^k\\ \Delta\vec{v}^k\end{bmatrix}, \tag{5.10b}$$

$$= -\left(\Phi^k\right)^T \Phi^k. \tag{5.10c}$$

As can be seen the exact Newton direction is always a descent direction for the natural merit function. Given that we only solve the Newton system approximately then we may define the residual as

$$\vec{r}^k = -\Phi^k - \mathbf{J}^k \begin{bmatrix}\Delta\vec{x}^k\\ \Delta\vec{v}^k\end{bmatrix}. \tag{5.11}$$

Giving us

$$\mathbf{J}^k \begin{bmatrix}\Delta\vec{x}^k\\ \Delta\vec{v}^k\end{bmatrix} = -\left(\Phi^k + \vec{r}^k\right). \tag{5.12}$$

The directional derivative of the merit function now becomes

$$D\psi^k = -\left(\Phi^k\right)^T \left(\Phi^k + \vec{r}^k\right). \tag{5.13}$$

If $\left(\Phi^k\right)^T \vec{r}^k \geq 0$ then the Newton direction will always be a descent direction. On the other hand if $\left(\Phi^k\right)^T \vec{r}^k < 0$ then the residual term could cause problems. If $\parallel \vec{r}^k \parallel < \parallel \Phi^k \parallel$ then we always have a descent direction. Thus, if an iterative solver is used for the Newton system then we require the stopping criteria

$$\parallel \vec{r}^k \parallel < \gamma \parallel \Phi^k \parallel \tag{5.14}$$

for some tolerance $0 < \gamma < 1$.

5.2.3 The Line–Search Method

Although we know we have a descent direction it may be that the non-linear nature of the root search problem Φ results in overshooting. Thus, to globalize the Newton method and make it more robust a line–search method can be employed. The idea is to make sure the line–search method finds a step length τ^k that results in a sufficient decrease of the merit function ψ when looking in the Newton direction. The Newton update is given by

$$\vec{x}^{k+1} = \vec{x}^k + \tau^k \Delta \vec{x}^k, \tag{5.15a}$$

$$\vec{v}^{k+1} = \vec{v}^k + \tau^k \Delta \vec{v}^k. \tag{5.15b}$$

If we look at the behavior of ψ^{k+1} then we have

$$\psi(\vec{x}^{k+1}, \vec{v}^{k+1}) = \psi(\vec{x}^k + \tau^k \Delta \vec{x}^k, \vec{v}^k + \tau^k \Delta \vec{v}^k). \tag{5.16}$$

If we use a linear approximation then we find

$$\psi^{k+1} \approx \underbrace{\psi^k + \left(\beta D \psi^k\right) \tau^k}_{f(\tau^k)}, \tag{5.17}$$

for now one can think of $\beta = 1$. We will explain the usage of β later. Observe the right-hand side is a simple linear real function of the step length. Whereas the left hand side is a highly non-linear real function of the step length. The idea is now to use a relaxed version of the linear approximation that is one choose $0 < \beta \ll 1$ and then one wishes to find maximum τ^k-values such that $\psi^{k+1}(\tau^k) < f(\tau^k)$. In which case we say we have a sufficient decrease. When we look for the step length we use an iterative approach. Initially we use the optimal Newton step length $\tau^k = 1$. If the sufficient decrease test fails then we reduce the step length by a constant fraction $\tau^k \leftarrow \alpha \tau^k$. Where the step reduction parameter α fulfills $\beta < \alpha < 1$. To avoid iterating forever a minimum step length value $0 < \tau_{\text{min}} \ll 1$ may be specified or a maximum iteration limit. The line–search method we have outlined is called an Armijo backtracking line–search method [Nocedal and Wright(1999)].

In excess to the sufficient decrease condition we may require that tetrahedra do not invert or we may even test if the tetrahedra do not change their volume too much from iteration to iteration. We can summarize our line–search algorithm in the following pseudo code,

```
1 :  Algorithm line–search
2 :    $\tau \leftarrow 1$
3 :    while forever
4 :      $v \leftarrow$ minimum tetrahedra volume at $\tau$
5 :      $\rho \leftarrow$ minimum tetrahedra volume ratio at $\tau$
6 :      if $\psi(\tau) < f(\tau)$ and $v > 0$ and $\rho > 0.1$ then
7 :        return $\tau$
8 :      end
9 :      $\tau \leftarrow \alpha\tau$
10 :   end
11 : end
```

5.2.4 The Stopping Criteria

It may be beneficial to use many different stopping criteria during the Newton iterations. In order to make sure the numerical method is efficient in the sense that computations give a sufficient improvement in the approximate solution and to guard towards potential numerical problems.

An absolute stopping criteria can be use to stop iteration as soon as one is close enough to a root. As the merit function is bounded from below by zero such a test can be stated as

$$\psi(\vec{x}^k, \vec{v}^k) \leq \varepsilon_{\text{abs}} \tag{5.18}$$

where $\varepsilon_{\text{abs}} > 0$ is a user specified tolerance. A relative stopping criteria can be used to make sure that one bails out if there is no prospect of any significant improvement,

$$\left|\psi^{k+1} - \psi^k\right| \leq \varepsilon_{\text{rel}}\psi^k \tag{5.19}$$

where $\varepsilon_{\text{rel}} > 0$ is a user specified tolerance. This kind of criteria could suggest a local minima could be near by. For our problem this would suggest that we are near a root for ϕ, so a close to zero gradient test might be useful,

$$\| \nabla\psi(\vec{x}^k, \vec{v}^k) \|_2 \leq \varepsilon_{\nabla} \tag{5.20}$$

where $\varepsilon_{\nabla} > 0$ is a user specified tolerance. It may be that the Newton direction is too small to make any significant change in the iterative value in which case it make sense to give up,

$$\| \begin{bmatrix} (\Delta\vec{x}^k)^T & (\Delta\vec{v}^k)^T \end{bmatrix}^T \|_\infty \leq \varepsilon_{\Delta} \tag{5.21}$$

where $\varepsilon_{\Delta} > 0$. It may be worthwhile to guard against a non descent Newton direction. Although our theory guarantees this never can occur numerical issues may cause trouble. That means we always require that

$$\nabla\psi(\vec{x}^k, \vec{v}^k) \begin{bmatrix} \Delta\vec{x}^k \\ \Delta\vec{v}^k \end{bmatrix} < 0. \tag{5.22}$$

To guard against infinite looping one should ensure to specify a maximum iteration count for the Newton method. Finally, it may be a good idea to test for stagnation which means that after having performed the line–search it should be tested if a significant change is being made to the current iterate,

$$\| \tau^k \begin{bmatrix} \left(\Delta\vec{x}^k\right)^T & \left(\Delta\vec{v}^k\right)^T \end{bmatrix}^T \|_\infty \leq \varepsilon_\tau \tag{5.23}$$

where $\varepsilon_\tau > 0$. Selecting the values of the ε tolerance parameters should not be done too aggressively. For instance we use a absolute tolerance of no more than 10^{-2}. Although our stagnation tolerances go as low as 10^{-15} we often only use 10^{-5} to test for relative convergence. The maximum iteration count we often select to be no more than 10 iterations.

5.2.5 Generating Good Starting Iterates

In [Hirota(2002)] a two point predictor-corrector method is used to generate an initial starting iterate for his non-linear implicit finite element method. We found this approach a little awkward to implement and opted for a different approach taking advantage of the optimization setting of the implicit method. The idea is to use a fixed number of gradient descent steps on the natural merit function (5.8). We already derived a closed form solution for the gradient (5.9). Thus, in each iteration of the gradient descent method we perform the update

$$\begin{bmatrix} \vec{x}^{k+1} \\ \vec{v}^{k+1} \end{bmatrix} = \begin{bmatrix} \vec{x}^{k} \\ \vec{v}^{k} \end{bmatrix} - \tau^k \nabla \psi^k. \tag{5.24}$$

To make sure the gradient descent method does not overshoot we apply a damped version. Thus, we find the step length τ^k using a backtracking line–search method where we test if the new iterate gives a lower ψ-value than ψ^k. This corresponds to picking a sufficient decrease parameter value of $\beta = 0$ and a step-reduction parameter value of $\alpha = \frac{1}{2}$. No other extra tests were done and we always run a fixed number of gradient descent iterations, 3 in our current implementation.

5.2.6 Incremental Loading

The idea of incremental loading means that one slowly increases the load forces $\vec{f}$. For each increment one re-invokes the Newton method using the previous solution as an initial guess for the new load. This can be summarized as follows

```
1 : **Algorithm incremental-load**
2 :   $\Delta\vec{f} \leftarrow$ some fraction of $\vec{f}$
3 :   $\vec{f} \leftarrow \vec{0}$
4 :   **while** $\vec{f}$ **not full load do**
5 :     $\vec{f} \leftarrow \vec{f} + \Delta\vec{f}$
6 :     $(\vec{x}, \vec{v}) \leftarrow$ **Newton–Method**$(\vec{x}, \vec{v}, \vec{f})$
7 :   **End**
8 :   **return** $(\vec{x}, \vec{v})$
**End**
```

In our test runs we often only used three equal sized increments.

5.2.7 Computing the Elasticity Tensor

For the implicit non-linear Finite element method we need a fourth order tensor. The fourth order tensor is not easily stored in a Matlab array. It is possible to flattern the tensor into a two dimensional array. The trick is to use the first two indices as block indices and the last two indices as entries into a sub-block. Thus, we store the fourth order tensor as a 3×3 array of matrix blocks each of dimension 3×3. We may write this as

$$\mathcal{A} \equiv \mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & \mathbf{A}_{13} \\ \mathbf{A}_{21} & \mathbf{A}_{22} & \mathbf{A}_{23} \\ \mathbf{A}_{31} & \mathbf{A}_{32} & \mathbf{A}_{33} \end{bmatrix} \tag{5.25}$$

where for any $\mathbb{I}, \mathbb{J}$ we have $\mathbf{A}_{\mathbb{I},\mathbb{J}} \in \mathbb{R}^{3\times3}$. In practice this means that the storage for $\mathcal{A}$ is a 9×9 matrix. Knowing this storage layout we may now access elements of the fourth order tensor as follows

$$\mathcal{A}_{\mathbb{I},\mathbb{J},\mathbb{K},\mathbb{M}} = \mathbf{A}_{(3(\mathbb{I}-1)+\mathbb{K}),(3(\mathbb{J}-1)+\mathbb{M})}. \tag{5.26}$$

When using Matlab then implementing the equations derived by Erleben in [Erleben(2011b)] can be computational expensive as they results in deep nested for loops that must be evaluated for all elements in a mesh. Using symbolic differentiation and code generation tools can help a lot in these circumstances. For instance in Matlab one can use the symbolic package to let Matlab derive a closed form solution for the elasticity tensor as well as the tangent stiffness matrix. Once Matlab have processed this one can ask Matlab to generate code matching the closed form formulas that Matlab has derived.

In the case of a Saint Venant–Kirchhoff material one may proceed as follows.

```
syms lambda mu real;
syms F11 F12 F13...
         F21 F22 F23...
         F31 F32 F33 real;
```

```
F = [F11 F12 F13;...
       F21 F22 F23;...
       F31 F32 F33...
      ];
C = (F')*F;
E = (1/2) * (C-eye(3,3));

% Strain energy function
psi = (1/2) * lambda* trace(E)^2 +...
          mu * trace(E*E);

% First Piola--Kirchhoff stress tensor
P = sym('TEMP')*ones(3,3);
for i=1:3
  for j=1:3
    P(i,j) = diff(psi, F(i,j));
  end
end

% Material Elasticity Censor
A = sym('TEMP')*ones(9,9);
for i=1:3
  for j=1:3
    for k=1:3
      for m=1:3
        r = (i-1)*3 + k;
        c = (j-1)*3 + m;
        A( r, c ) = diff( P(i,j) , F(k,m));
      end
    end
  end
end
A = simplify(A);
matlabFunction(A,'file','compute_A.m');

% Tangent stiffness element
syms Ga1 Ga2 Ga3 real;
syms Gb1 Gb2 Gb3 real;
Ga = [Ga1; Ga2; Ga3];
Gb = [Gb1; Gb2; Gb3];

Kab = sym('TEMP')*ones(3,3);
Kab(:,:) = 0;
for i=1:3
  for k=1:3
    for j=1:3
      for m=1:3
        r = (i-1)*3 + k;
        c = (j-1)*3 + m;
        Kab( i, k ) = Kab( i, k ) +...
                          A(r,c)*Gb(m)*Ga(j);
      end
    end
  end
end
Kab = simplify(Kab);
matlabFunction(Kab,'file','compute_K.m');
```

One may generate c-code using `ccode` function instead of using `matlabFunction`. Overall one can think of this work process as an efficient way to perform full loop-unrolling and exploiting any common sub-expressions in any computations. The main challenge now lies in writing the strain energy in terms of the deformation gradient components.

5.2.8 Built-in Matlab Support

Rather than implementing a Newton method from scratch one can take advantage of numerical methods already in Matlab. Matlab already has in built root search solver `fsolve` which can be used for fast prototyping or lazy programming. To use the built in solver one must first make a Matlab function that returns the current value of Φ as well as the current Jacobian. This could look like this,

```
function [phi J] = myfun( z, .... )
  phi = call_phi( z, .... );
  J = call_nabla_phi( z, .... );
end
```

Afterwards one may set up specific choices and tolerances for the numerical method, before creating a function that can be passed to `fsolve`. This could be written as,

```
  options = optimset('Jacobian','on',...
                     'Display','iter',...
                     'TolFun', 10e-2,...
                     'TolX',10e-5 );

  f  = @(z)  myfun( z, .... );
  z = [ p; v ];
  z  = fsolve( f, z, options);

  p = z(1:length(p));
  v = z(length(p)+1:end);
```

The advantage of writing ones own solver is of course that one can specialize the implementation for the specific task at hand whereas the built-in functions of Matlab may solve more general problems and may not be as fast.

5.3 The Influence of Boundary Conditions

Often we wish to specify boundary conditions in order to fixate parts of the surface of a deformable model. This type of boundary condition is a Dirichlet boundary condition. Assume we apply such a boundary condition to the i^{th} node of our computational mesh. Then the boundary condition can be written as

$$\vec{x}_i = \vec{c}_i, \tag{5.27a}$$
$$\vec{v}_i = \vec{0}, \tag{5.27b}$$

where $\vec{c}_i \in \mathbb{R}^3$ is some known fixed position. We will now explain the details in applying this boundary condition in the implicit non-linear finite element method. If we look at our root search problem then we should drop all variables with such boundary conditions as these are always fulfilled. Obviously we have $\Delta\vec{x}_i^k = \vec{0}$ and $\Delta\vec{v}_i^k = \vec{0}$. The boundary condition influence our computations when we compute the Newton direction. If we let $\mathcal{A}$ be the nodal index set of active boundary conditions and $\mathcal{F}$ the nodal index set of free nodes. Then the implicit function defining the root search problem can be imaginary partitioned as

$$\Phi(\vec{x}_\mathcal{F}, \vec{v}_\mathcal{F}, \vec{x}_\mathcal{A}, \vec{v}_\mathcal{A}) = \begin{bmatrix} \Phi_{\vec{x}_\mathcal{F}} \\ \Phi_{\vec{v}_\mathcal{F}} \\ \Phi_{\vec{x}_\mathcal{A}} \\ \Phi_{\vec{v}_\mathcal{A}} \end{bmatrix} = \vec{0} \tag{5.28}$$

where

$$\Phi_{\vec{x}_\mathcal{F}} = \vec{x}_\mathcal{F} - \vec{x}_\mathcal{F}^t - \Delta t \vec{v}_\mathcal{F}, \tag{5.29a}$$
$$\begin{aligned}\Phi_{\vec{v}_\mathcal{F}} &= \left(\mathbf{M}_{\mathcal{F},\mathcal{F}} + \Delta t \mathbf{C}_{\mathcal{F},\mathcal{F}}\right)\vec{v}_\mathcal{F} \\ &\quad - \mathbf{M}_{\mathcal{F},\mathcal{F}}\vec{v}_\mathcal{F}^t - \Delta t \vec{f}_\mathcal{F} + \Delta t \vec{k}_\mathcal{F},\end{aligned} \tag{5.29b}$$
$$\Phi_{\vec{x}_\mathcal{A}} = \vec{x}_\mathcal{A} - \vec{c}, \tag{5.29c}$$
$$\Phi_{\vec{v}_\mathcal{A}} = \vec{v}_\mathcal{A}. \tag{5.29d}$$

This results in the Newton system

$$\begin{bmatrix} -\Phi_{\vec{x}_\mathcal{F}} \\ -\Phi_{\vec{v}_\mathcal{F}} \\ -\Phi_{\vec{x}_\mathcal{A}} \\ -\Phi_{\vec{v}_\mathcal{A}} \end{bmatrix} = \mathcal{J} \begin{bmatrix} \Delta\vec{x}_\mathcal{F} \\ \Delta\vec{v}_\mathcal{F} \\ \Delta\vec{x}_\mathcal{A} \\ \Delta\vec{v}_\mathcal{A} \end{bmatrix}, \tag{5.30}$$

where

$$\mathcal{J} = \begin{bmatrix} \mathbf{I}_{\mathcal{F},\mathcal{F}} & -\Delta t \mathbf{I}_{\mathcal{F},\mathcal{F}} & \mathbf{0} & \mathbf{0} \\ \Delta t \mathbf{K}_{\mathcal{F},\mathcal{F}} & \left(\mathbf{M}_{\mathcal{F},\mathcal{F}} + \Delta t \mathbf{C}_{\mathcal{F},\mathcal{F}}\right) & \Delta t \mathbf{K}_{\mathcal{F},\mathcal{A}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & I_{\mathcal{A},\mathcal{A}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_{\mathcal{A},\mathcal{A}} \end{bmatrix}.$$

For our specific boundary condition we have $-\Phi_{\vec{x}_\mathcal{A}} = -\Phi_{\vec{v}_\mathcal{A}} = \Delta\vec{x}_\mathcal{A} = \Delta\vec{v}_\mathcal{A} = \vec{0}$. Substitution this prior knowledge gives

$$\begin{bmatrix} -\Phi_{\vec{x}_\mathcal{F}} \\ -\Phi_{\vec{v}_\mathcal{F}} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_{\mathcal{F},\mathcal{F}} & -\Delta t \mathbf{I}_{\mathcal{F},\mathcal{F}} \\ \Delta t \mathbf{K}_{\mathcal{F},\mathcal{F}} & \left(\mathbf{M}_{\mathcal{F},\mathcal{F}} + \Delta t \mathbf{C}_{\mathcal{F},\mathcal{F}}\right) \end{bmatrix} \begin{bmatrix} \Delta\vec{x}_\mathcal{F} \\ \Delta\vec{v}_\mathcal{F} \end{bmatrix}. \tag{5.31}$$

Applying a Shur complement method we observe that this would be equivalent to solving

$$\mathbf{A}_{\mathcal{F},\mathcal{F}} \Delta\vec{v}_\mathcal{F}^k = \vec{b}_\mathcal{F} \tag{5.32}$$

and then afterwards compute

$$\Delta\vec{x}^k_{\mathcal{F}} = \Delta t \Delta\vec{v}^k_{\mathcal{F}} - \Phi_{\vec{x}_{\mathcal{F}}}. \tag{5.33}$$

Following the same procedure other boundary conditions can be applied. For our particular case we observe that we can solve a sub-part of our original full system from Section 5.2.1.

If we consider the corotational linear finite element method and the semi-implicit time stepper of the non-linear finite element method then they both first compute a velocity update solving a linear system that we here write abstractly as

$$\mathbf{A}\mathbf{v} = \vec{b}. \tag{5.34}$$

Following this the methods compute the position update written abstractly as

$$\vec{x}^{t+1} = \vec{x}^t + \Delta t \vec{v}. \tag{5.35}$$

Applying the boundary conditions and using the imaginary partitions we have

$$\mathbf{A}_{\mathcal{F}\mathcal{F}}\mathbf{v}_{\mathcal{F}} = \vec{b}_{\mathcal{F}} - \mathbf{A}_{\mathcal{F}\mathcal{A}}\vec{v}_{\mathcal{A}}. \tag{5.36}$$

Followed by

$$\vec{x}^{t+1}_{\mathcal{F}} = \vec{x}^t_{\mathcal{F}} + \Delta t \vec{v}_{\mathcal{F}}. \tag{5.37}$$

The finite volume method is even more easy to apply the boundary conditions to as one here works directly on the variables in $\mathcal{F}$.

5.4 Applying Surface Traction

Let us first consider the finite element methods when applying a surface traction. Recall that after putting our equations of motion into weak form we have [Erleben(2011a)]

$$P_\rho + P_b + P_e + P_t = 0, \tag{5.38}$$

where the P_t term is given by the spatial integral

$$P_t = -\int_{\partial v_t} \vec{t} \cdot \vec{w} ds. \tag{5.39}$$

Here $\vec{w}$ is an arbitrary test function. P_t can be interpreted as the power of the traction. For ease of derivation let us consider a single triangle with label e that is part of the spatial boundary surface ∂v_t. Let the triangle area be A^e and let the nodes be labelled by i, j, and k.

Next step is to apply the Galerkin approximations to discretize the traction term. That is we make the substitutions

$$\vec{w} \approx N_i\vec{w}_i + N_j\vec{w}_j + N_k\vec{w}_k = \sum_{a\in\{i,j,k\}} N_a\vec{w}_a, \tag{5.40a}$$

$$\vec{t} \approx N_i\vec{t}_i + N_j\vec{t}_j + N_k\vec{t}_k = \sum_{b\in\{i,j,k\}} N_b\vec{t}_b. \tag{5.40b}$$

Here N's are the linear shape functions for the triangle and $\vec{w}_a$'s are arbitrary unknown nodal virtual displacements. The $\vec{t}_b$ are known nodal surface tractions that is being applied to the mesh. After substitution we have,

$$P_t^e \approx -\int_{\partial v_t^e} \left(\sum_a N_a \vec{w}_a\right) \cdot \left(\sum_b N_b \vec{t}_b\right) ds. \tag{5.41}$$

We may rewrite this approximation into matrix notation

$$P_t^e \approx -\begin{bmatrix}\vec{w}_i\\ \vec{w}_j\\ \vec{w}_k\end{bmatrix}^T \mathbf{L}^e \begin{bmatrix}\vec{t}_i\\ \vec{t}_j\\ \vec{t}_k\end{bmatrix} \tag{5.42}$$

where

$$\mathbf{L}^e = \int_{\partial v_t^e} \begin{bmatrix} N_i^2\mathbf{I} & N_iN_j\mathbf{I} & N_iN_k\mathbf{I}\\ N_jN_i\mathbf{I} & N_j^2\mathbf{I} & N_jN_k\mathbf{I}\\ N_kN_i\mathbf{I} & N_kN_j\mathbf{I} & N_k^2\mathbf{I}\end{bmatrix} ds. \tag{5.43}$$

Following the standard finite element procedure by using the fact that the $\vec{w}$ terms are arbitrary and all power terms sum to zero we can drop the displacements and get the final nodal element traction force term,

$$\vec{f}_t^e = \mathbf{L}^e \begin{bmatrix}\vec{t}_i\\ \vec{t}_j\\ \vec{t}_k\end{bmatrix}. \tag{5.44}$$

The global nodal traction force is found by assembling all the element traction forces. That is one sums over all e that is part of ∂v_t. We name $\mathbf{L}$ the nodal load distribution matrix and all that remains is to find a closed-form solution for evaluating its value.

To derive a closed form solution we will make a change of variables into the isoparametric space $(\mathcal{X}, \mathcal{Y}, \mathcal{Z})$ where the triangle node i is placed at $(0,0,0)$, j at $(1,0,0)$ and k at $(0,1,0)$. The isoparametric triangle has constant area equal to $\frac{1}{2}$. Further, since the shape functions are the barycentric coordinates ie. the area weighted coordinates then in isoparametric space we have

$$N_j = \mathcal{X}, \tag{5.45a}$$
$$N_k = \mathcal{Y}, \tag{5.45b}$$
$$N_i = 1 - \mathcal{X} - \mathcal{Y}. \tag{5.45c}$$

The change of variables is then

$$\mathbf{L}^e = \int_{\partial v_t^e} \begin{bmatrix} \mathcal{L}_{ii}\mathbf{I} & \mathcal{L}_{ij}\mathbf{I} & \mathcal{L}_{ik}\mathbf{I}\\ \mathcal{L}_{ji}\mathbf{I} & \mathcal{L}_{jj}\mathbf{I} & \mathcal{L}_{jk}\mathbf{I}\\ \mathcal{L}_{ki}\mathbf{I} & \mathcal{L}_{kj}\mathbf{I} & \mathcal{L}_{kk}\mathbf{I}\end{bmatrix} j d\mathcal{S}. \tag{5.46}$$

For linear triangular elements we have $j = A^e$ and

$$\mathcal{L}_{ii} = (1 - \mathcal{X} - \mathcal{Y})^2\,, \tag{5.47a}$$
$$\mathcal{L}_{ij} = (1 - \mathcal{X} - \mathcal{Y})\,\mathcal{X}, \tag{5.47b}$$
$$\mathcal{L}_{ik} = (1 - \mathcal{X} - \mathcal{Y})\,\mathcal{Y}, \tag{5.47c}$$
$$\mathcal{L}_{ji} = \mathcal{X}\,(1 - \mathcal{X} - \mathcal{Y})\,, \tag{5.47d}$$
$$\mathcal{L}_{jj} = \mathcal{X}^2, \tag{5.47e}$$
$$\mathcal{L}_{jk} = \mathcal{X}\mathcal{Y}, \tag{5.47f}$$
$$\mathcal{L}_{ki} = \mathcal{Y}\,(1 - \mathcal{X} - \mathcal{Y})\,, \tag{5.47g}$$
$$\mathcal{L}_{kj} = \mathcal{Y}\mathcal{X}, \tag{5.47h}$$
$$\mathcal{L}_{kk} = \mathcal{Y}^2. \tag{5.47i}$$

These are all simple polynomials in the isoparametric space and can be algebraically integrated to yield the final result

$$\mathbf{L}^e = \frac{A^e}{24}\begin{bmatrix} 2\mathbf{I} & \mathbf{I} & \mathbf{I} \\ \mathbf{I} & 2\mathbf{I} & \mathbf{I} \\ \mathbf{I} & \mathbf{I} & 2\mathbf{I} \end{bmatrix}. \tag{5.48}$$

Now we will turn our attention to the finite volume method. Here we must look at the control area s_i of a surface node i. Then we have

$$\vec{f_t^i} = \int_{s_i} \vec{t} ds. \tag{5.49}$$

The control area will consist of sub parts of all triangles sharing the node i. Thus, we can replace the integral with a summation over the triangles

$$\vec{f_t^i} = \sum_a \underbrace{\int_{s_i^a} \vec{t} ds}_{\vec{h}_a}. \tag{5.50}$$

Let us develop the sub-term corresponding to the triangle e that has two other nodes labelled j and k. That is the surface integral $\vec{h}_e = \int_{s_i^e} \vec{t} ds$. The corners of the sub control area s_i^e wrt. the i^{th} vertex is

$$\vec{p}_1 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T \tag{5.51a}$$
$$\vec{p}_2 = \begin{bmatrix} \frac{3}{4} & \frac{1}{4} & 0 \end{bmatrix}^T \tag{5.51b}$$
$$\vec{p}_3 = \begin{bmatrix} \frac{3}{4} & 0 & \frac{1}{4} \end{bmatrix}^T \tag{5.51c}$$
$$\vec{p}_4 = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix}^T \tag{5.51d}$$

where we used barycentric coordinates. The mid point of the sub control area is

$$\vec{p} = \frac{1}{2}\left(\vec{p}_1 + \vec{p}_2 + \vec{p}_3 + \vec{p}_4\right) = \begin{bmatrix} \frac{34}{48} & \frac{7}{48} & \frac{7}{48} \end{bmatrix}^T. \tag{5.52}$$

Thus, assuming that we can linear interpolate the surface traction from nodal traction values (as in the finite element method) and using a mid point rule approximation we have

$$\vec{h}_e \approx \frac{A^e}{3}\left(\frac{34}{48}\vec{t}_i + \frac{7}{48}\vec{t}_j + \frac{7}{48}\vec{t}_k\right). \tag{5.53}$$

Rather than evaluating (5.50) by iterating over nodes we can instead iterate over triangles and scatter the contribution from each triangle to the nodal summation in (5.50). Thus, we find

$$\vec{f}_t^e = \mathbf{L}^e \begin{bmatrix} \vec{t}_i \\ \vec{t}_j \\ \vec{t}_k \end{bmatrix}. \tag{5.54}$$

where

$$\mathbf{L}^e = \frac{A^e}{144}\begin{bmatrix} 34\mathbf{I} & 7\mathbf{I} & 7\mathbf{I} \\ 7\mathbf{I} & 34\mathbf{I} & 7\mathbf{I} \\ 7\mathbf{I} & 7\mathbf{I} & 34\mathbf{I} \end{bmatrix}. \tag{5.55}$$

We can now apply any nodal surface traction distribution to a finite element method using the load matrix in (5.48) or to the finite volume method using the load matrix in (5.55).

5.5 Using Preconditioning

In the case of the semi-implicit non-linear finite element method the coefficient matrix of the velocity update is the mass matrix $\mathbf{M}$. This is known to be a symmetric and positive definite matrix. Often it will have larger diagonal values than off-diagonal values. Thus, a good choice for a preconditioner is to use the diagonal of the mass matrix. In particular if a lumped mass matrix is used then this preconditioner will result in a solution in one iteration of PCG.

When analyzing the velocity update of the corotational linear finite element method we observe that the coefficient matrix is a sum of three terms $\mathbf{A} = \mathbf{M} + \Delta t\mathbf{C} + \Delta t^2\mathbf{K}$. When Δt becomes very small (as it should be to ensure a good finite difference approximation) the mass matrix will dominate the expression. Thus, again we note that using the diagonal of the mass matrix will be a good choice for a preconditioner.

The same analysis applies in the case of the Shur reduced version of the Newton system for the implicit non-linear finite element method (5.7). For the full Newton system (5.4) it is less obvious what preconditioner to use.

5.6 Mesh Quality Measures

The numerical conditioning of the matrix equations that we form in the numerical methods we have derived is highly dependent on the shape of the individual tetrahedra in the mesh. Thus, to ensure that ones mesh is well shaped one often uses a histogram of tetrahedron quality measures to inspect ones mesh. In essence a quality measure tries to measure how well shaped a given tetrahedron really is. That is how close a tetrahedron is to being an equilateral tetrahedron. Often the quality measure is designed to give a specific value say 1 if a "perfect" tetrahedron is found or the value 0 is the worst case tetrahedron is found. Observe that different quality measures might have different values but the principle stays the same. Thus, a histogram shows the distribution of good and bad tetrahedra in a given mesh.

We will now present a few quality measure definitions. In the following we assume that we are given a single tetrahedron with local node indices 1, 2, 3 and 4. The radius ratio quality of a tetrahedron is 2 times the ratio of the radius of the inscribed sphere r_{in} divided by the radius of the circumscribed sphere r_{out} [Shewchuk(2002)],

$$Q_{rr} \equiv 2\frac{r_{\text{in}}}{r_{\text{out}}}. \tag{5.56}$$

An equilateral tetrahedron achieves the maximum possible quality of 1. The inscribed radius and maximum edge length ratio is given by [Shewchuk(2002)],

$$Q_{rl} \equiv 2\sqrt{6}\frac{r_{\text{in}}}{\max_{1\leq i<j\leq 4} l_{ij}} \tag{5.57}$$

where l_{ij} is the length of the edge between the nodes i and j. The minimum sine of the dihedral angles of a tetrahedron is given by [Shewchuk(2002)]

$$Q_{\theta} \equiv \frac{9\sqrt{2}}{8}V \min_{1\leq i<j\leq 4}\left\{\frac{l_{ij}}{A_k A_l}\right\}. \tag{5.58}$$

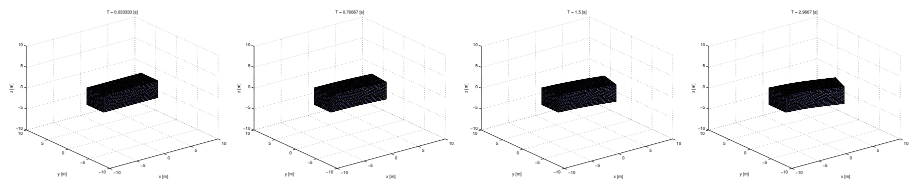
Here we assumed the traditional convention and labeled the tetrahedron nodes by i, j, k and m. Here A_k is the signed area of the triangle opposing the k^{th} node. V is the signed volume of the tetrahedron. If any of the A's are zero then we will have a division by zero when computing the quality measure. If A becomes zero then we have a flat tetrahedron and dihedral angles must then be 0 and 180 degrees, implying that the sine of the dihedral angles is always zero. Thus, by definition the quality measure should be zero. The eigenvalue or mean ratio of a tetrahedron known as the volume-length ratio measure is defined by [Joe(1991)]

$$Q_{vl} \equiv 12\frac{(3V)^{\frac{2}{3}}}{\sum_{1\leq i<j\leq 4} l_{ij}^2}. \tag{5.59}$$

This value may be used as a shape quality measure for the tetrahedron. For an equilateral tetrahedron, the value of this quality measure will be 1. For any other tetrahedron, the value will be between 0 and 1.

5.7 Numerical Experiments

We have implemented all the numerical methods in Matlab (R2010a) and run our experiments on a cluster with 8 Intel(R) Xeon(R) CPU X5355 @ 2.66GHz


(a) Bend

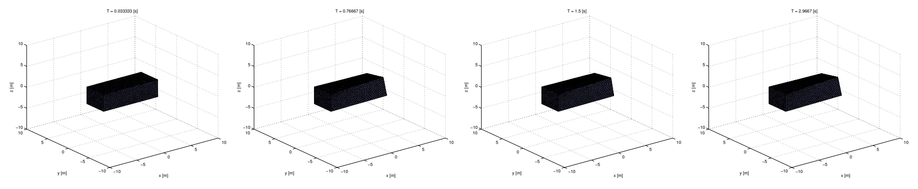

(b) Twist

Figure 5.1: Still frames for the bend and twist motions taken from the finite volume method simulation.

and 18 GB shared ram running Linux version 2.6.20-gentoo-r8. Each experiment only used a single CPU core.

In our experiments we have used two different test setups. In both cases we use a cantilever beam. In the first case a downward constant uniform traction $\vec{t} = \begin{bmatrix} 0 & -5 & 0 \end{bmatrix}^T$ (kN m^{-2}) is applied at the end surface of the beam. The load is pulling the beam downward in a bending motion. In the other test case we apply a rotational traction $\vec{t} = 5 \begin{bmatrix} 0 & z & -y \end{bmatrix}^T$ (kN m^{-2}) at the end surface of the beam. Here y and z are the current spatial coordinate values of any point on the surface of the end. This causes the beam to undergo a twisting motion. We refer to the test cases as *bend* and *twist*. Figure 5.1 shows examples of the twist and bend motions that is being simulated.

For sake of readability we introduce some abbreviations for the methods we have examined. The corotational linear elastic finite element method is denoted COR, the finite volume method is referenced by FVM, the semi-implicit non-linear finite element method is given by SI-FEM and the implicit version by I-FEM. The lumped version of SI-FEM is labelled with L-FEM.

5.7.1 Meshing Issues

For our convergence studies we created a series of meshes for our beam. Thus, the beam shape is an oblong box shape with increasing resolution. The box shape was chosen to be $10 \times 4 \times 4$ meters in size and centered around the origin. We used DistMesh [Persson and Strang(2004)] to create the tetrahedral meshes. We varied the mesh resolution by using different values for the minimum edge lengths of the resulting tetrahedra mesh,

$$L_{\text{min}} = \{0.80, 0.75, 0.65, 0.5, 0.45, 0.4, 0.35, 0.3\}. \tag{5.60}$$

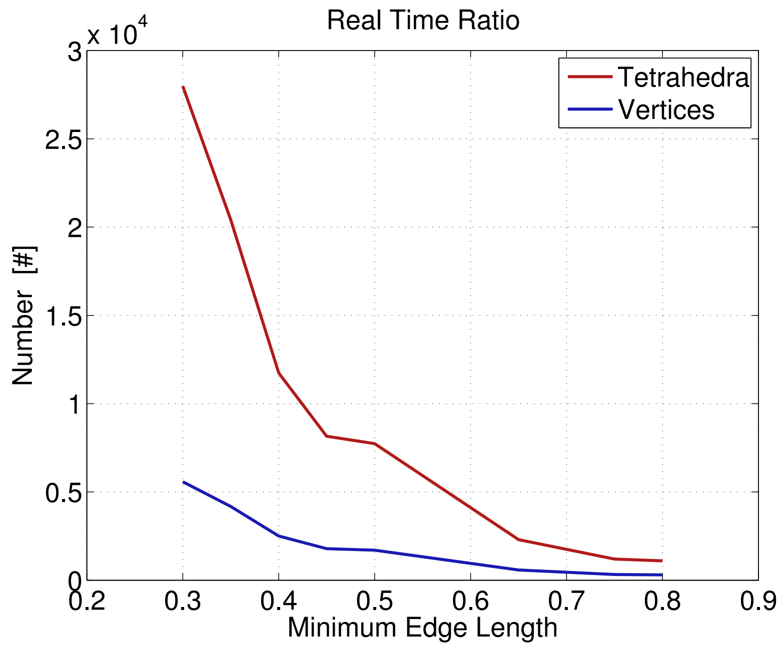


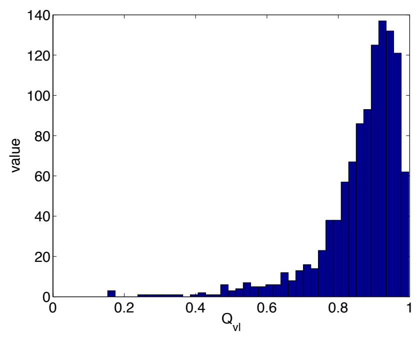
Figure 5.2: Mesh size as a function of minimum edge length parameter for DistMesh. Observe that mesh sizes range from 1000 tetrahedra up to 27000 tetrahedra.

We set DistMesh to generate a uniform mesh resulting in a final mesh having similar sized tetrahedron. Figure 5.2 shows how our mesh sizes change.

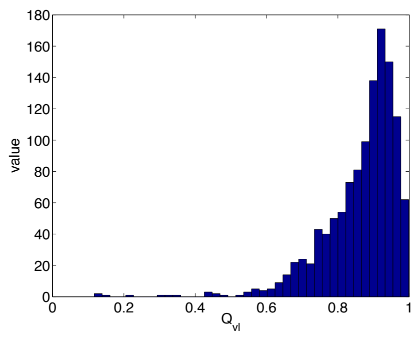
We inspected the mesh quality measures of the our generated meshes to ensure that they were not too bad for our convergence studies. The histograms of the volume length quality measure are shown in Figure 5.3. From our histograms we conclude that the mesh quality is similar for all resolutions and the mesh elements are good conditioned as they all have values close to one. The other quality measures we listed in Section 5.6 shows the same tendencies. We omitted showing the histograms here due to space considerations.

5.7.2 Parameter Selection

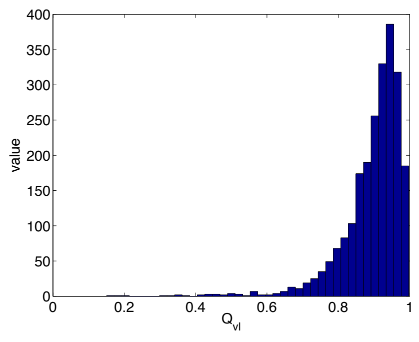
When using the SI-FEM, I-FEM, L-FEM and FVM we apply a Saint Venant–Kirchhoff material and use the material parameters: mass density $\rho = 1000$ (Kg m^{-3}), Young modulus of $E = 10 \cdot 10^5$ (Pa), Poisson ratio $\nu = 0.3$, and viscous damping $c = 0.0004$ (s^{-1}). COR uses a different material model with a similar expression to the Saint Venant–Kirchhoff model except it uses the small strain tensor in place of the large strain tensor [Erleben(2011a)]. We observed that materials behave more “stiff” using this material model. Thus, we use a low Young Modulus value $E = 5000$ (Pa) in our tests to create motion that can be



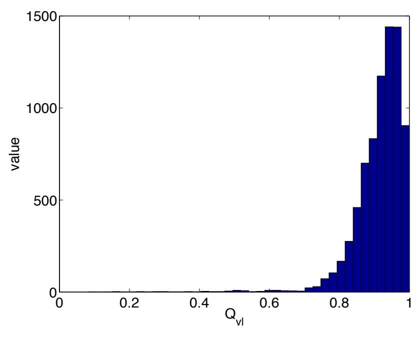
(a) $l_{\min} = 0.80$ m



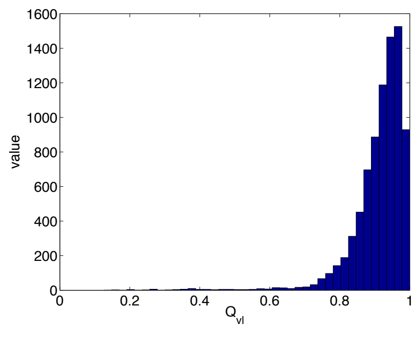
(b) $l_{\min} = 0.75$ m



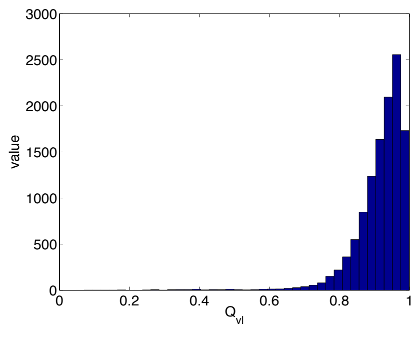
(c) $l_{\min} = 0.65$ m



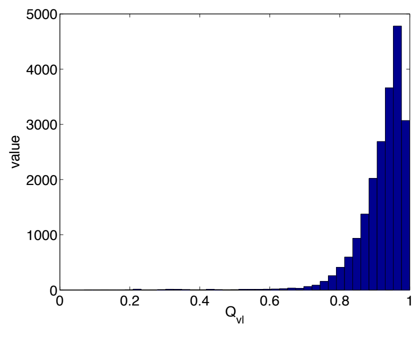
(d) $l_{\min} = 0.5$ m



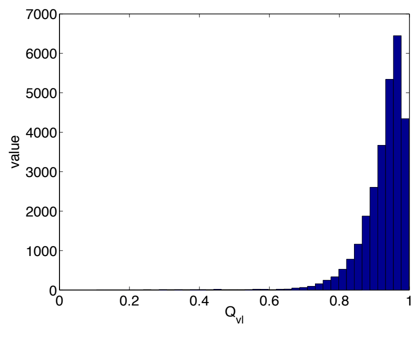
(e) $l_{\min} = 0.45$ m



(f) $l_{\min} = 0.4$ m



(g) $l_{\min} = 0.35$ m



(h) $l_{\min} = 0.3$ m

Figure 5.3: Histograms of volume-length ratio quality measures for our test meshes with varying minimum edge length size $l_{\min}$. Observe that in all cases the quality distribution looks the same and that many elements have a quality close to one.

Name	E (Pa)	ν	ρ (Kg m^{-3})
Cartilage	$0.69 \cdot 10^6$	0.018	$^\dagger 1000$
Cortical bone	$16.16 \cdot 10^9$	0.33	$^\diamond 1600$
Cancellous bone	$452 \cdot 10^6$	0.3	$^\diamond 1600$
Rubber	$0.01 \cdot 10^9$	0.48	1050
Concrete	$30 \cdot 10^9$	0.20	2320
Copper	$125 \cdot 10^9$	0.35	8900
Steel	$210 \cdot 10^9$	0.31	7800
Aluminium	$72 \cdot 10^9$	0.34	2700
Glass	$50 \cdot 10^9$	0.18	2190

Table 5.1: Material parameter values for common types of materials. Young Modulus E, Poisson Ratio ν and mass density ρ. Observe ($^\diamond$) is for dense bone and (†) could not be found.

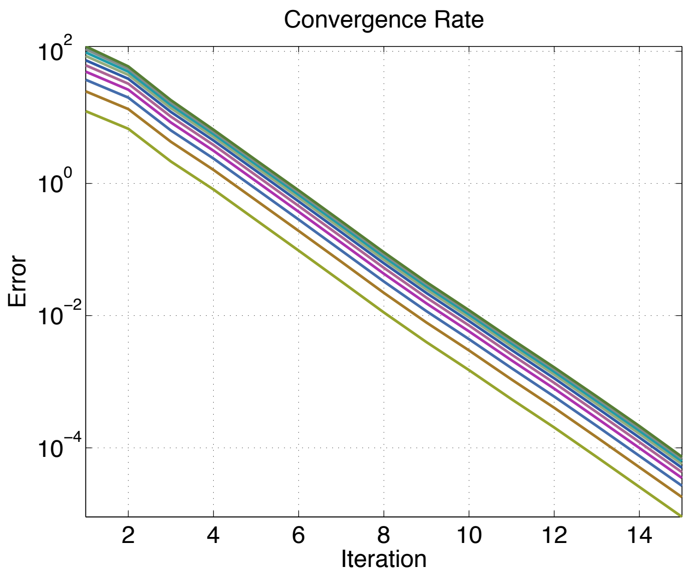
visually inspected. The remaining material parameters are chosen in the same fashion as for the FEM and FVM methods. The chosen material parameters do not correspond to any real-life material we know of. We just selected some values that would allow us to compare motions across the different methods. One may refer to Table 5.1 for more real life values ([1]).

We compared the convergence rate of a PCG solver against a non preconditioned version. Figure 5.4 shows our results for the SI-FEM method. We observe that the preconditioned version has approximately two orders of magnitude more accuracy. We observed the same tendency for the COR method. We omitted plots due to space considerations. We have not applied preconditioner to the GMRES solver for the I-FEM method as it is not obvious which preconditioner to use for (5.4). In all cases of PCG and GMRES we used default tolerance set by Matlab. According to Matlab documentation PCG uses a default tolerance of $1.0 \cdot 10^{-6}$ and a maximum number of iterations of 20. GMRES uses the same tolerance and no restarts by default and a maximum iteration limit of 10 iterations.

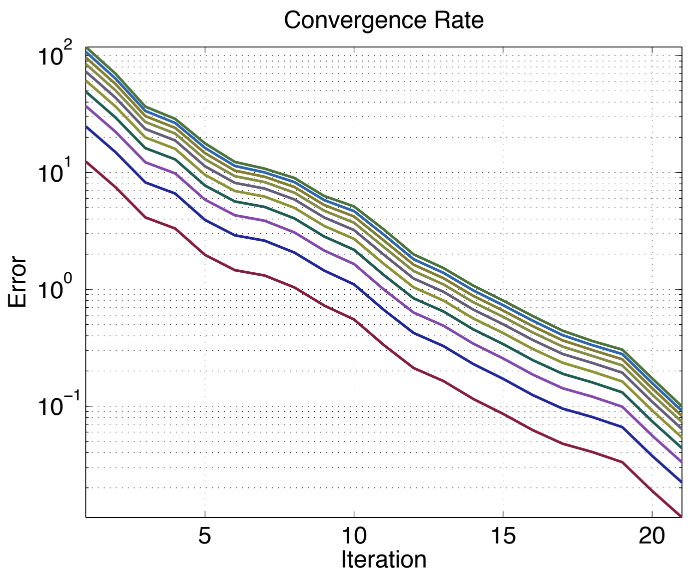
For the I-FEM method we use the Newton method outlined in Section 5.2. For our testing we use the default values given in Table 5.2. Most of the tolerances are picked not to be too aggressive. In all our tests we observed that the Newton method gave up reaching its maximum number of iterations. Three of the numerical parameters control the algorithmic behavior of our Newton method: the number of incremental loads (n_{load}) ie. the number of times the

[1] Values are taken from

- `http://en.wikipedia.org/wiki/Density`,
- `http://en.wikipedia.org/wiki/Young%27s_modulus`,
- `http://en.wikipedia.org/wiki/Poisson%27s_ratio`,
- `http://www.engineeringtoolbox.com/poissons-ratio-d_1224.html`, and
- `http://www.engineeringtoolbox.com/young-modulus-d_417.html`.



(a) With preconditioner



(b) Without preconditioner

Figure 5.4: Convergence rate comparison of conjugate gradient method for the semi-implicit non-linear finite element method with and with out preconditioner. The plots show convergence rates for 10 time steps.

Description	Value
Absolute tolerance (ε_{abs})	$1.0 \cdot 10^{-2}$
Relative tolerance (ε_{rel})	$1.0 \cdot 10^{-5}$
Zero gradient tolerance (ε_{∇})	$1.0 \cdot 10^{-15}$
Too small Newton direction (ε_{Δ})	$1.0 \cdot 10^{-7}$
Stagnation tolerance (ε_{τ})	$2.2204 \cdot 10^{-15}$
Maximum Newton iterations (n_{max})	10
Gradient descent iterations for warm start (n_{grad})	3
Incremental loads (n_{load})	3

Table 5.2: Default parameter values for the Newton method used to solve the implicit non-linear finite element method.

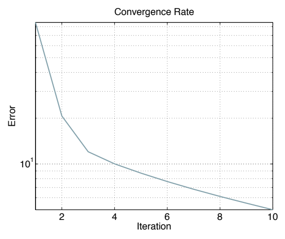
Newton method is restarted, the number of gradient descent iterations used for warm starting the Newton method (n_{grad}), and the maximum number of allowed Newton iterations (n_{max}) per restart.

To investigate the parameters influence on the convergence properties of the Newton method we varied their values and compared the convergence rate plots shown in Figure 5.5. For the comparison study we used a single time step of the twist test case. From our plots it shows that using incremental loading seems to worsen the final accuracy of the Newton method. Further, it seems that the gradient descent warm start has little influence on the final accuracy. In all cases we observe an initial super linear convergence rate that seems to deteriorate into a linear convergence rate.

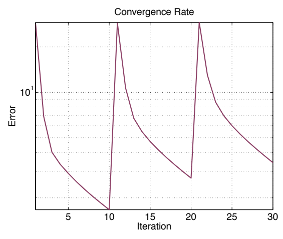
5.7.3 Performance Measurements

We have measured the average wall clock time per time step for both test cases under varying mesh sizes. Figure 5.6 shows our measurements. In our experiments we used $\Delta t = 10^{-4}$ (s) for COR, SI-FEM, L-FEM and FVM. For I-FEM we used $\Delta t = 10^{-2}$. Thus, the measurements are based on approximately 1000 time steps and 100 time steps respectively. We observe that I-FEM is the most computational expensive method up to four orders of magnitude slower than the fastest method which is the FVM method. The COR, SI-FEM and L-FEM shows similar performance and is roughly two orders of magnitude slower than FVM. There is a tendency showing that COR is slightly more expensive than SI-FEM and L-FEM. All methods scale linear in the number of mesh elements which is expected.

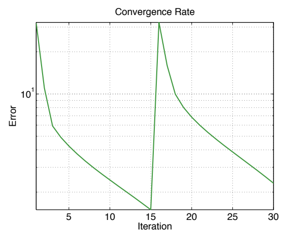
We have investigated the details of how the wall-clock time per time step behaves under varying parameters. Figure 5.7 shows selected plots of our measurements when varying the mesh element size. As can be observed in the figure our results indicate that each simulation run has a similar behavior through-out the simulation. We observe there is great variation across methods and mesh element sizes. Not surprising has the mesh element size great influence on the performance as this directly influence the size of the matrix equations. Nor is



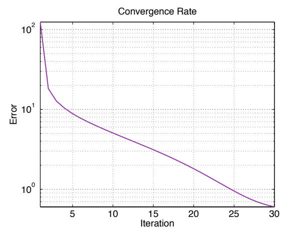
(a) $n_{\text{max}} = 10, n_{\text{grad}} = 3, n_{\text{load}} = 1$



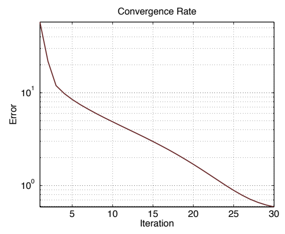
(b) $n_{\text{max}} = 10, n_{\text{grad}} = 3, n_{\text{load}} = 3$



(c) $n_{\text{max}} = 16, n_{\text{grad}} = 6, n_{\text{load}} = 2$

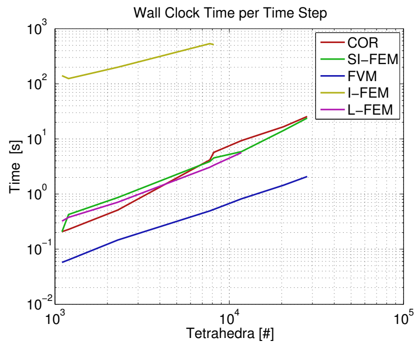


(d) $n_{\text{max}} = 30, n_{\text{grad}} = 0, n_{\text{load}} = 1$

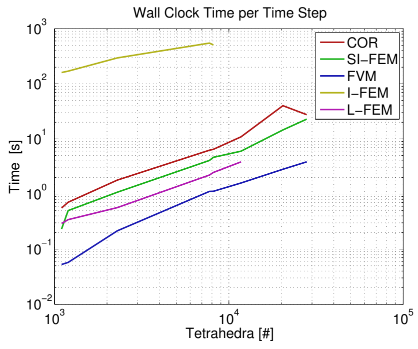


(e) $n_{\text{max}} = 30, n_{\text{grad}} = 9, n_{\text{load}} = 1$

Figure 5.5: Comparison study of numerical parameters used for the Newton method that solves the implicit non-linear finite element method. The number of incremental loads n_{load} is varied as well as the number of gradient descent steps n_{grad} and number of Newton iterations n_{max}.

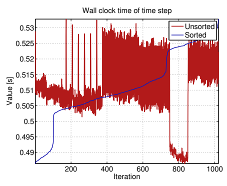


(a) Bend

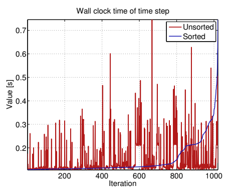


(b) Twist

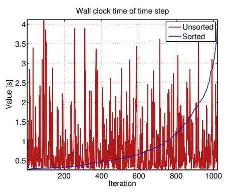
Figure 5.6: Performance measurements for both the bend and twist test cases for the first second of simulated time. Observe that I-FEM is four orders of magnitude slower than FVM and that FVM is the fastest method.



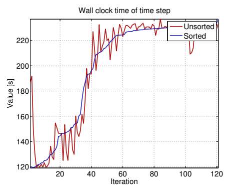
(a) COR $\Delta t = 10^{-4}$ s, $l_{\text{min}} = 0.65$ m



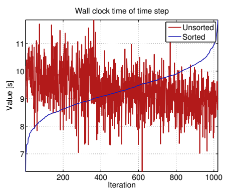
(b) FVM $\Delta t = 10^{-4}$ s, $l_{\text{min}} = 0.65$ m



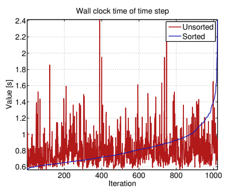
(c) SI-FEM $\Delta t = 10^{-4}$ s, $l_{\text{min}} = 0.65$ m



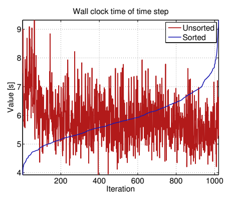
(d) I-FEM $\Delta t = 10^{-4}$ s, $l_{\text{min}} = 0.65$ m



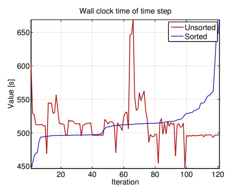
(e) COR $\Delta t = 10^{-4}$ s, $l_{\text{min}} = 0.4$ m



(f) FVM $\Delta t = 10^{-4}$ s, $l_{\text{min}} = 0.4$ m



(g) SI-FEM $\Delta t = 10^{-4}$ s, $l_{\text{min}} = 0.4$ m



(h) I-FEM $\Delta t = 10^{-4}$ s, $l_{\text{min}} = 0.45$ m

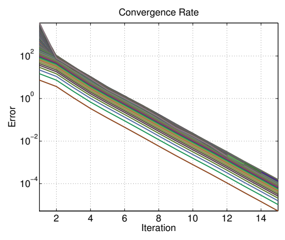
Figure 5.7: Wall clock comparison for the bend test case with varying mesh element sizes. The plots show measurements of the first 1 second of simulated time. Observe the large variation.

it surprising that the different methods have different computational cost. For the same mesh size and time step size the FVM method has the lowest computational cost per time step. The SI-FEM method and the COR method are more even whereas the I-FEM method is the most expensive.

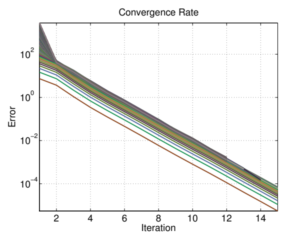
5.7.4 Convergence Rates

The COR method and the SI-FEM method both use PCG for their velocity update and the I-FEM method uses a Newton method. We have investigated the convergence rate of the PCG solvers and the Newton method across varying mesh element sizes and time step sizes. Figure 5.8 shows selected convergence error plots for varying mesh element size and fixed time step size.

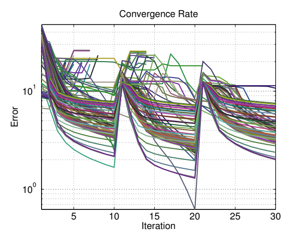
Figure 5.9 shows selected convergence plots with fixed mesh element size and varying time step size. We observe in the shown plots that for the COR



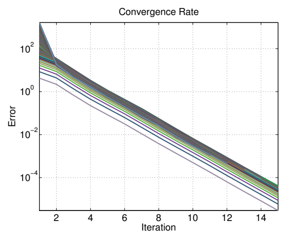
(a) COR $\Delta t = 10^{-4}$ s, $l_{\min} = 0.65$ m



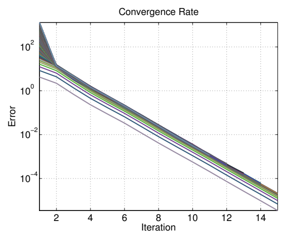
(b) SI-FEM $\Delta t = 10^{-4}$ s, $l_{\min} = 0.65$ m



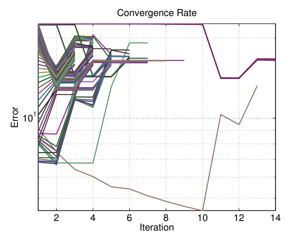
(c) I-FEM $\Delta t = 10^{-4}$ s, $l_{\min} = 0.65$ m



(d) COR $\Delta t = 10^{-4}$ s, $l_{\min} = 0.4$ m

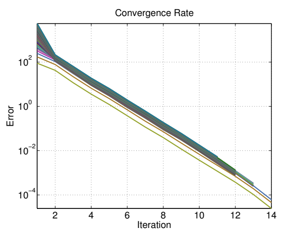


(e) SI-FEM $\Delta t = 10^{-4}$ s, $l_{\min} = 0.4$ m

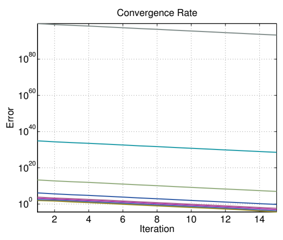


(f) I-FEM $\Delta t = 10^{-4}$ s, $l_{\min} = 0.45$ m

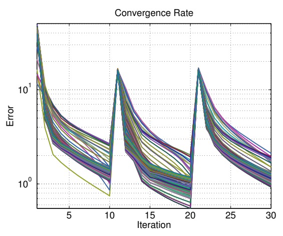
Figure 5.8: Selected convergence plots for the bend test case with varying mesh element size. Notice that convergence rate is independent of the mesh element size. Convergence plots are shown for all time steps during the first 1 second of simulated time.



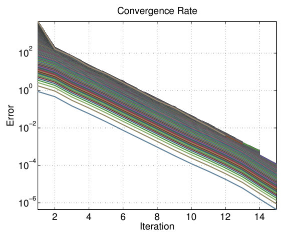
(a) COR $\Delta t = 10^{-2}$ s, $l_{\text{min}} = 0.8$ m



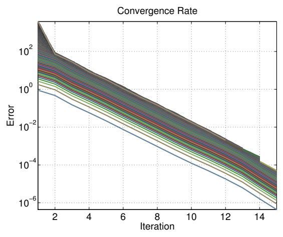
(b) SI-FEM $\Delta t = 10^{-2}$ s, $l_{\text{min}} = 0.8$ m



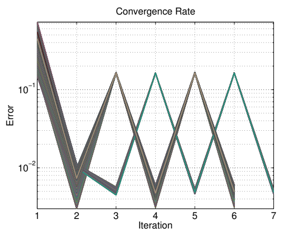
(c) I-FEM $\Delta t = 10^{-2}$ s, $l_{\text{min}} = 0.8$



(d) COR $\Delta t = 10^{-4}$ s, $l_{\text{min}} = 0.8$ m



(e) SI-FEM $\Delta t = 10^{-4}$ s, $l_{\text{min}} = 0.8$ m



(f) I-FEM $\Delta t = 10^{-4}$ s, $l_{\text{min}} = 0.8$ m

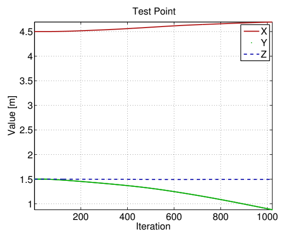
Figure 5.9: Selected convergence plots for the bend test case with varying time step size. Convergence plots are shown for all time steps during the first 1 second of simulated time.

and SI-FEM methods the convergence rates appear to be independent of the mesh element size. The time step size has great influence on SI-FEM too low a value seems to yield a poor convergence constant. We believe this is related to an unstable time integration as we generally observe simulation blow-ups for SI-FEM when using $\Delta t = 10^{-2}$ seconds. The most efficient constant for SI-FEM seems to be obtained with $\Delta t = 10^{-3}$ seconds or lower. The I-FEM method is sensitive towards changes in mesh element size and time step size. It seems to work best for a coarse mesh and with a time step size in the order of $\Delta t = 10^{-2}$ seconds. Due to space considerations we have omitted showing all our plots of all test simulations. Our observations generalize to all our tests.

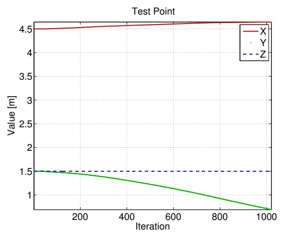
5.7.5 Mesh Convergence Experiment

We have traced the motion curves of the test point $(4.5, 1.5, 1.5)$ during the first one second of simulated time while varying the mesh element size. We then visually inspected if the motion curve changed when we increased the mesh resolution. We compared the motion curves at finest resolution between each simulation method to see if the methods differ in their results.

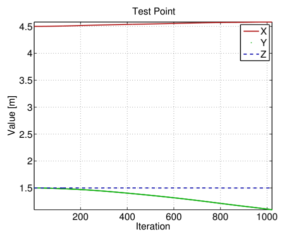
We observed that the motion curves did not seem to vary greatly when increasing the mesh element size. Thus, Figure 5.10 shows only the final motion



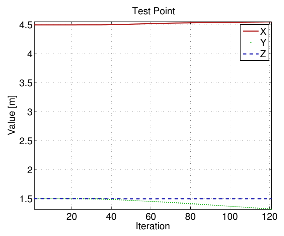
(a) COR $\Delta t = 10^{-4}$ s, $l_{\text{min}} = 0.3$ m



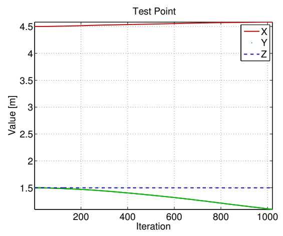
(b) FVM $\Delta t = 10^{-4}$ s, $l_{\text{min}} = 0.3$ m



(c) SI-FEM $\Delta t = 10^{-4}$ s, $l_{\text{min}} = 0.3$ m



(d) I-FEM $\Delta t = 10^{-2}$ s, $l_{\text{min}} = 0.65$ m



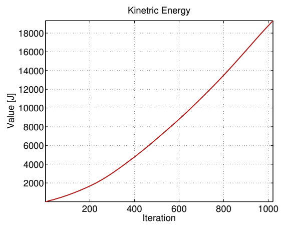
(e) L-FEM $\Delta t = 10^{-4}$ s, $l_{\text{min}} = 0.4$ m

Figure 5.10: The motion curves of the test point $(4.5, 1.5, 1.5)$ at the finest mesh resolution for one second of simulated time of the bend test case. Observe that all methods produce similar motion.

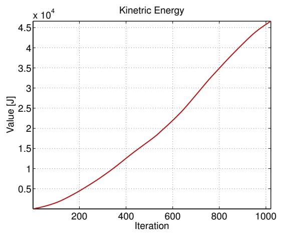
curves at the finest mesh resolution where we obtained usable results. When comparing motion curves across methods we do see some differences although all curves have similar shapes. The L-FEM and SI-FEM methods are similar. The FVM method seems to overshoot and the I-FEM method undershoots. This is not surprisingly as this is emergent behavior of the discretization methods and depends on the time step size. The COR method is different from the others this too is not surprisingly given that its material model is different from the other methods.

We recorded the total kinetic energy and compared this for different mesh element sizes. Figure 5.11 shows the kinetic energy plots at the finest resolution mesh for the bend test case. Again we observe little dependency on varying the mesh element size. When we compare between different methods we see similar shaped plots, but with a different energy rate. The kinetic energy of FVM and I-FEM is five times larger than SI-FEM and L-FEM whereas COR is almost twice that of SI-FEM and L-FEM.

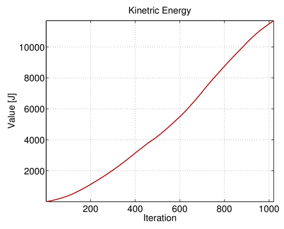
We investigated the volume gain while varying the minimum mesh element size. Figure 5.12 shows the volume gain plots for selected mesh element size in the case of bend test case. We observe that FVM, SI-FEM and L-FEM have similar shaped curves. The minimum and maximum element volume gain for FVM is twice that of SI-FEM and L-FEM. The COR method have a minimum and



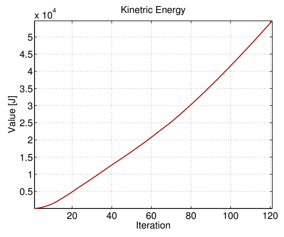
(a) COR $\Delta t = 10^{-4}$ s, $l_{\text{min}} = 0.3$ m



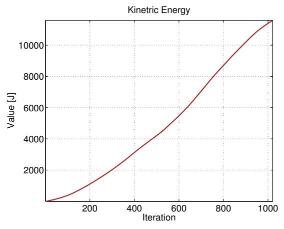
(b) FVM $\Delta t = 10^{-4}$ s, $l_{\text{min}} = 0.3$ m



(c) SI-FEM $\Delta t = 10^{-4}$ s, $l_{\text{min}} = 0.3$ m

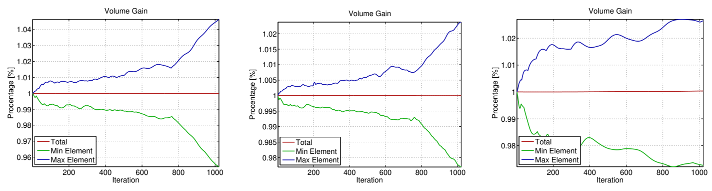


(d) I-FEM $\Delta t = 10^{-2}$ s, $l_{\text{min}} = 0.45$ m

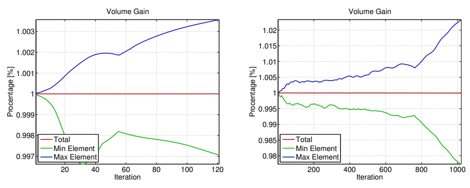


(e) L-FEM $\Delta t = 10^{-4}$ s, $l_{\text{min}} = 0.4$ m

Figure 5.11: The kinetic energy at the finest mesh resolution for one second of simulated time of the bend test case. Observe that all methods produce similar energy behavior.



(a) $\Delta t = 10^{-4}$ s, $l_{\text{min}} =$ 0.45 m (b) $\Delta t = 10^{-4}$ s, $l_{\text{min}} =$ 0.45 m (c) $\Delta t = 10^{-4}$ s, $l_{\text{min}} =$ 0.45 m



(d) $\Delta t = 10^{-4}$ s, $l_{\text{min}} =$ 0.45 m (e) $\Delta t = 10^{-4}$ s, $l_{\text{min}} =$ 0.45 m

Figure 5.12: Volume gain of various methods for the bend test case. The finest mesh resolution was shown where we have results for all methods.

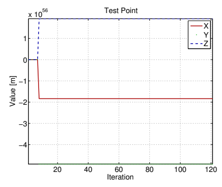
maximum volume gain of the same order as the SI-FEM and L-FEM methods but a different plot shape which is more similar to the I-FEM method. The I-FEM method shows the smallest volume gain for the minimum and maximum elements. It is approximately an order of magnitude smaller. We have repeated all of the above experiments for the twist test case and found similar conclussions. Due to space considerations we have omitted showing the plots of these tests.

5.7.6 Time Step Convergence Experiment

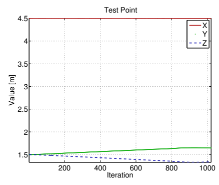
We repeated the same test setups as in our mesh convergence experiments. We kept the minimum mesh element size fixed and varied the time step size instead. As before we measured the motion curve of the test point, the kinetic energy and the volume gain. We will present results for the twist test case as the bend test case shows the same observations.

We have omitted showing detailed plots of all experiments due to space considerations. Figure 5.13 shows that SI-FEM is unstable for large time steps. Figure 5.14 suggests that the I-FEM method has not yet converged and even smaller time step sizes would be needed. When we examine the convergence rates from Section 5.7.4 it seems more plausible that the problem lies in very poor numerical solutions. It raises the question of whether a line–search method really is best suited for this method?

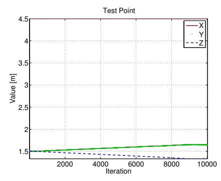
We observed that the L-FEM method is similar to the SI-FEM method. The L-FEM method does not appear to be unstable for larger time step sizes and only show slightly different “displaced” kinetic energy curve. Both methods seem to converge to the same result.



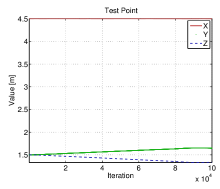
(a) $\Delta t = 10^{-2}$ s, $l_{\text{min}} = 0.8$ m



(b) $\Delta t = 10^{-3}$ s, $l_{\text{min}} = 0.8$ m

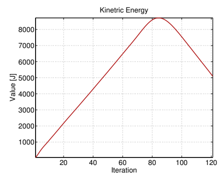


(c) $\Delta t = 10^{-4}$ s, $l_{\text{min}} = 0.8$ m

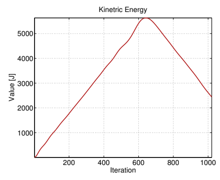


(d) $\Delta t = 10^{-5}$ s, $l_{\text{min}} = 0.8$ m

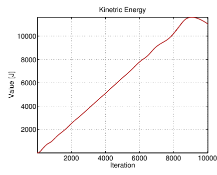
Figure 5.13: Motion curve of the test point $(4.5, 1.5, 1.5)$ using the SI-FEM method for the twist test case under varying time step sizes. Observe the simulation blow up for $\Delta t = 10^{-2}$ seconds.



(a) $\Delta t = 10^{-2}$ s, $l_{\text{min}} = 0.8$ m



(b) $\Delta t = 10^{-3}$ s, $l_{\text{min}} = 0.8$ m



(c) $\Delta t = 10^{-4}$ s, $l_{\text{min}} = 0.8$ m

Figure 5.14: The kinetic energy when using the I-FEM method for the twist test case under varying time step sizes. Notice that I-FEM appears to not have converged.

We noticed that the FVM method looks similar to the L-FEM and SI-FEM methods. When looking at the volume gain we see that FVM is an order of magnitude larger than L-FEM and SI-FEM. Looking at the motion curves and energy plots one can not tell the L-FEM and FVM methods apart by visual inspection. This is in agreement with theory stating the FVM is algebraic equivalent to the L-FEM case [Erleben(2011a)].

5.7.7 Adaptive Time Stepping

To determine what the proper time step size should be for our simulation methods we used an adaptive time step method. This will try to adapt the time step size during simulation to the best suitable value. The idea is simply to take a coarse simulation step with time step size Δt and a fine simulation step using two time steps with $\frac{\Delta t}{2}$. If the nodal positional difference between coarse and fine simulation steps is larger than a given threshold $10^[-4]$ then the time step size is reduced to half its size $\Delta t \leftarrow \frac{\Delta t}{2}$ otherwise the simulation step is taken using the results of the fine simulation. If there have been no time step reductions for a fixed number of times, four in our simulations, then we try to double the time step size. One could add an extra test against volume inversion and use this as a criteria for time step reduction or use some guard against a too large a rate of volume change. In our tests we only used the positional difference test.

Based on the previous experiments we observe that the I-FEM is not computational efficient and suffers from lack of numerical robustness. Further, L-FEM is really no different than FVM . Thus, we have tested COR, SI-FEM and FVM on the eight test cases described below. For each simulation we let it run for 3 seconds of simulated time. To further test the differences between the competing simulation methods we created a few more test cases besides the twist and bend test cases.

In our first variation we linear increase the traction in the twist and bend test cases during the first simulated second to five times that of the constant traction used in the standard twist and bend test cases. We refer to these two test cases as *incremental bend* and *incremental twist* or just I-bend and I-twist for short. In our second variation we used our fixed position boundary conditions instead of the traction to generate the bending and twisting motions. Here we apply a time varying Dirichlet boundary conditions to the free end of the beam. During the first simulated second we twist the free end one time around the x axis and we bend the free end $\frac{\pi}{2}$ radians around the y axis. We refer to these test cases as *boundary condition* bend and twist or BC-bend and BC-twist for short. Finally we extend our portfolio of test cases by a *squeeze* and *stretch* test case. Here we use time varying boundary conditions on the end caps that during the first simulated second either linear compress the beam to half size along the x axis or linear elongate the beam to twice its length along the x axis.

The timing results are shown in Table 5.3. The adaptive time step size method for the FVM results in average time-step size per test case that varied from 0.0034 to 0.0001 seconds and a standard deviation that varied from

0.0001 to 0.0015 seconds. For the SI-FEM we killed *squeeze* after frame 55, where wall clock time per simulation step took more than 10 hours to compute. In the other test cases the average time-step size varied from 0.0001 to 0.0017 seconds and a standard deviation that varied from 0.0001 to 0.0008 seconds. For the COR method the average time step size varied from 0.0001 to 0.0009 with a standard deviation that varied from 0.0002 to 0.0005. From these average numbers we conclude that the order of time step size for all methods are the same. For the wanted accuracy a time step size of $\Delta t \approx 10^{-4}$ seconds to 10^{-5} seconds should be used. Looking more closely we observe that FVM can take larger time steps than SI-FEM and that COR

FVM	Avg	Max	Std
Bend	0.0018	0.0040	0.0009
I-Bend	0.0006	0.0080	0.0011
BC-Bend	0.0002	0.0010	0.0001
Twist	0.0017	0.0040	0.0008
I-Twist	0.0034	0.0080	0.0015
BC-Twist	0.0001	0.0010	0.0001
Squeeze	0.0001	0.0010	0.0001
Stretch	0.0002	0.0010	0.0001
SI-FEM	Avg	Max	Std
Bend	0.0017	0.0020	0.0007
I-Bend	0.0016	0.0040	0.0008
BC-Bend	0.0001	0.0010	0.0001
Twist	0.0010	0.0020	0.0005
I-Twist	0.0015	0.0040	0.0008
BC-Twist	0.0001	0.0010	0.0001
Squeeze	NA	NA	NA
Stretch	0.0002	0.0010	0.0001
COR	Avg	Max	Std
Bend	0.0003	0.0040	0.0005
I-Bend	0.0001	0.0080	0.0003
BC-Bend	0.0003	0.0020	0.0002
Twist	0.0005	0.0020	0.0003
I-Twist	0.0002	0.0080	0.0003
BC-Twist	0.0002	0.0020	0.0002
Squeeze	0.0003	0.0010	0.0002
Stretch	0.0009	0.0020	0.0005

Table 5.3: Variation of average (Avg) and maximum (Max) time step sizes and standard deviation (Std) across different test cases for different methods. All numbers are given in seconds. Observe that all methods used a time step size of the same order and has a low standard deviation. NA means not available.

We also measured the wall clock time of each adaptive time step. FVM varied from 2.63 seconds to 2.73 seconds with standard deviation in the range of

0.0061 seconds to 0.1827 seconds. SI-FEM used on average from 3.3 seconds to 3.7 seconds with a standard deviation from 0.14 seconds to 0.37 seconds. Finally COR uses on average from 20.22 seconds to 22.29 seconds with a standard deviation from 0.42 seconds to 24.89 seconds. This indicates that FVM and SI-FEM are comparable in performance characteristics whereas COR is more expensive and has less predictive performance.

Finally we verified which test cases resulted in element inversion. Our results are summarized in Table 5.4. From the coarse grain summary it appears that SI-FEM is slightly better to avoid element inversion.

	FVM	SI-FEM	COR
Bend	+	+	+
I-Bend	-	+ †	-
BC-Bend	-	-	-
Twist	+	+	+
I-Twist	+	+	-
BC-Twist	-	-	-
Stretch	-	-	+
Squeeze	-	NA	-

Table 5.4: Volume consistency testing. The table summarizes test cases that kept a positive volume (+) and those that resulted in element inversions (-). Notice that SI-FEM is slightly better than the others († postive but decreasing). NA means not available.

In Figure 5.15 we show a still frame after 1.5 seconds of simulated time for the twist test cases. Figure 5.16 shows for the bend cases and Figure 5.17 the stretch and squeeze test cases. In all simulations we have observed that the I-Bend test case has different energy plots between the FVM and SI-FEM method. In all other test cases we observe similar shaped energy plots between the FVM and SI-FEM methods. The COR method does have similar shapes as SI-FEM but time appears to be slowed down. We omitted showing energy plots due to space considerations. When we examine the actual motion produced by the three methods. It quickly shows that COR method produces strange behavior when we have extreme deformations like in the BC-Bend test case and BC-Twist cases. SI-FEM and FVM produce very similar motions. However, in the case of BC-Bend SI-FEM shows in our view a better result than FVM. In our opinion COR seems bad at dealing with the stretch case whereas SI-FEM and FVM looks more plausible. All methods show problems with the extreme squeeze test case. However, in our view COR seems to be the worst.

5.8 Discussion and Further Work

We have proposed several implementation specific alternatives and discussed variations on numerical methods. The results are ready to implement algo-

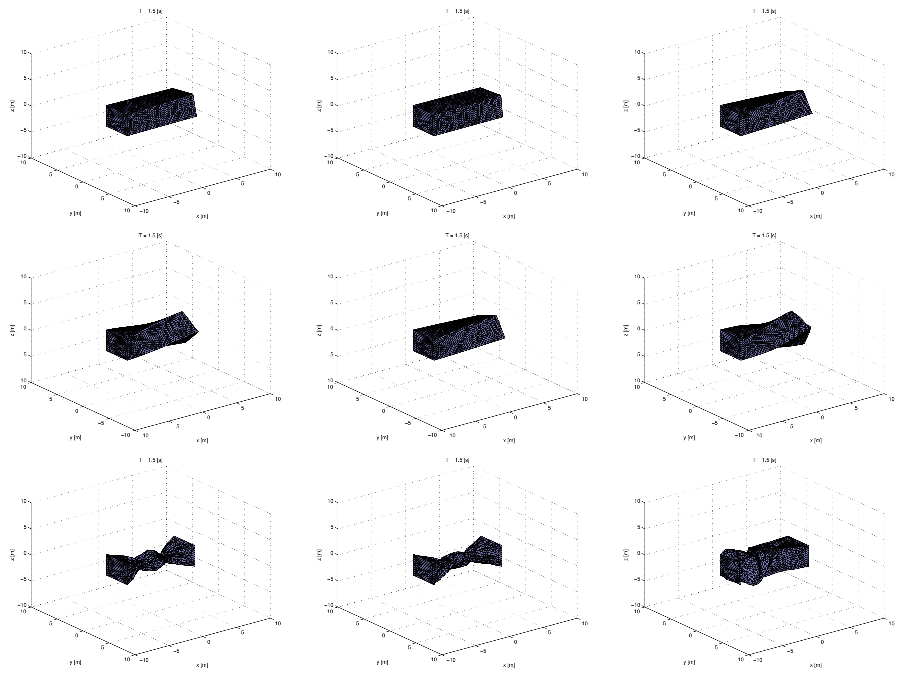


Figure 5.15: Top row shows the Twist test case, second row is the I-Twist test case and the last row is the BC-Twist test case. Columns are FVM, SI-FEM and COR methods respectively.

rithms which we have backed up with an open source Matlab implementation [Erleben(2011e)]. In our opinion we have conducted a large scale parameter study and performed several convergence experiments and performance measurement. Our testing includes a total of approximately 1000 computing hours the equivalent of five days run time on our 8 CPU node cluster.

We have omitted a parameter study of PCG and GMRES methods mostly because in our experience the default settings from Matlab seem to work well for us in past projects. It could be interesting to investigate if tuning these parameters could provide better results. We did not use preconditioner for GMRES. As we explained in Section 5.2.1 one can apply a Shur system reduction which allows usage of PCG and makes preconditioning straightforward. Even if we gained better accuracy of the Newton system solution we have little hope this will change the overall convergence behavior as we have initially tried with direct methods without seeing any change in convergence behavior. The PCG method has one major advantage over GMRES as it does not need restarts nor extra internal storage. So from a performance viewpoint one should use the Shur reduction approach.

From our convergence rate measurements COR and SI-FEM showed the expected behavior. I-FEM does not appear to be robust. In cases where we varied

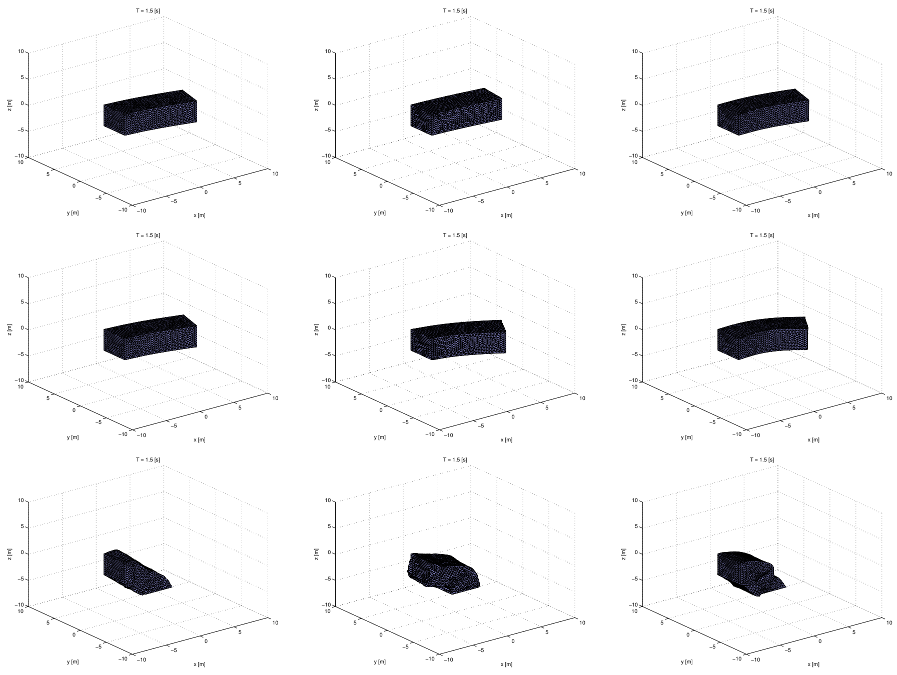


Figure 5.16: Top row shows the Bend test case, second row is the I-Bend test case and the last row is the BC-Bend test case. Columns are FVM, SI-FEM and COR methods respectively.

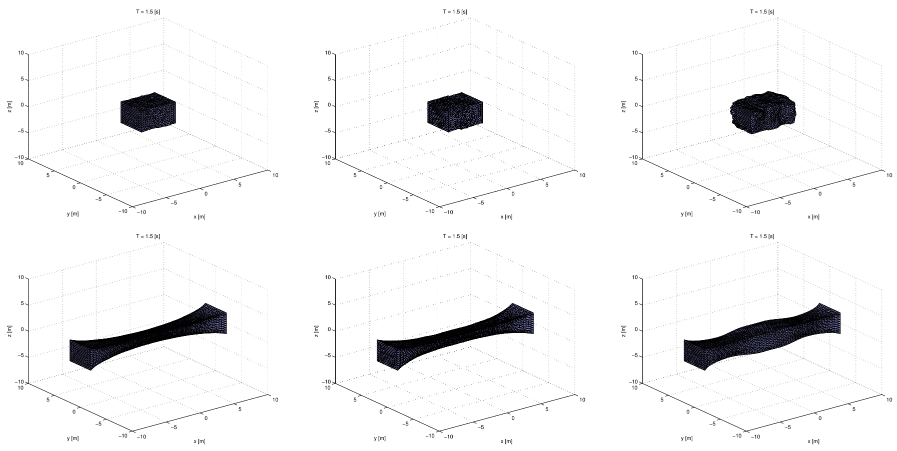


Figure 5.17: Top row is the squeeze test case and the bottom row is the stretch test case. Columns are FVM, SI-FEM, and COR methods respectively.

time step size and minimum mesh element size I-FEM showed erratic behavior and even divergence. Our parameter studies suggested to us that neither warm starting nor incremental loading seem to be worth the effort when solving the I-FEM. We speculate that this may be due to the non-linear nature combined with the high dimensionality of the problem. Maybe a different numerical approach would be more robust and efficient? We leave this for future work.

Our convergence experiments showed that FVM behaves like SI-FEM and L-FEM. It is computationally cheap compared to those as it does not require a PCG method. Besides FVM does not suffer from the time integration instability that SI-FEM showed in our experiments. Performance study suggest that COR is far too expensive than FVM and I-FEM is even more expensive or suffers from robustness issues. Our adaptive time step studies did show that SI-FEM deliver more visual pleasing results than FVM and can avoid element inversion is some cases where FVM fails.

In our view for animation purpose the FVM method seems to be the winner as the most robust and efficient method that can compute physical realistic motions and allows for a great number of material models. Although COR is a competitive method in the literature it is hard-wired to the small strain elasticity model and our experiments suggest that COR is a poor competitor. FVM does suffer from one major drawback as it is tailored for linear elements assuming a constant deformation gradient over each element. If this assumption is broken then FVM can not be used. In our view the L-FEM method seems to be the best alternative in this case. If one can afford a slightly larger computational cost then SI-FEM method is to be prefered over FVM and L-FEM.

Having given this recommendation on methods we should mention that there are subjects we have not touched upon. For instance we have omitted incompressibility, plasticity, and contact forces. We have restricted our derivations to linear tetrahedral elements as these are common in computer graphics. Isoparametric elements can be advantageous for numerical integration of the weak form equations. We have omitted this subject altogether as this is rarely seen in computer graphics. Besides the linear elements often results in closed form solutions or simple numerical approximations so the isoparametric elements seem in our view to be overkill. In any case we leave all these subjects for future work.

Chapter 6

Gluing FEM objects and Contact Forces

Blah bla.

6.1 Introduction

When modeling human tissue with deformable models using finite element method (FEM) then each tissue in the body is represented by a single FEM object. For now we can think of a FEM object as a set of algebraic equations defined on a computational mesh like a tetrahedra mesh. As an example the Femur bone is represented by one computational mesh and the Tibiea would be represented by another computational mesh. Ligaments attached to both bones help keep the knee joint together and these would be represented by a third FEM object. At the end caps of the bones one finds a thin layer of cartilage which for all purpose can be modelled as being rigidly attached to the bone surface. Inbetween the two cartilage layers one finds the Miniscus which are allowed to move freely between the two cartilage surfaces. The synovial fluid fills up all cavaties in the knee and is kept in place by a membrane. In our models we will ignore the fluid effects.

From our motivating example we realize that our simulation method need to be able to deal with two FEM objects that are in static contact. We call this "gluing". The dynamic contact is the other type of contact we must deal with. This involves determing the contact surfaces which change shape during motion and computing the contact forces that prevent penetration at the contact surfaces and causes friction at the contact surface.

6.2 Gluing Object Together

Before describing how to glue together two objects we will describe the typical assembly process. Imagine an object with the nodal index set $\mathcal{I} = \{1, 2, \ldots\}$ and that we have a partitioning of into two subsets $\mathcal{A} \subset \mathcal{I}$ and $\mathcal{B} \subset \mathcal{I}$ such that $\mathcal{A} \cap \mathcal{B} \neq \emptyset$ and $\mathcal{A} \cup \mathcal{B} = \mathcal{I}$. The assembly process of any global finite element matrix $\mathbf{A}$ from a set of element matrices $\mathbf{A}^e$ can be written as

$$\mathbf{A}_{ij} = \sum_{e \in \mathcal{E}_{ij}} \mathbf{A}^e_{\mathbb{IJ}} \tag{6.1}$$

where $i, j \in \mathcal{I}$ are global nodal indices. The element index set $\mathcal{E}_{ij}$ is defined as the subset of all elements that have both i and j in their index set and $\mathbb{I}$ and $\mathbb{J}$ are the corresponding local indices of element e which corresponds to the global indices i and j. Applying the partitioning to the assembly process results in

$$\mathbf{A}^{\mathcal{A}}_{ij} = \sum_{e \in \mathcal{E}^{\mathcal{A}}_{ij}} \mathbf{A}^e_{\mathbb{IJ}} \tag{6.2}$$

and

$$\mathbf{A}^{\mathcal{B}}_{ij} = \sum_{e \in \mathcal{E}^{\mathcal{B}}_{ij}} \mathbf{A}^e_{\mathbb{IJ}}. \tag{6.3}$$

That is we simply restricted $\mathcal{E}_{ij}$ to the partitions $\mathcal{A}$ and $\mathcal{B}$. Now the global assembly can be constructed from the partitioned assemblies

$$\mathbf{A}_{ij} = \begin{cases} \mathbf{A}^{\mathcal{A}}_{ij} + \mathbf{A}^{\mathcal{B}}_{ij} & \text{if } i, j \in \mathcal{A} \cap \mathcal{B}, \\ \mathbf{A}^{\mathcal{A}}_{ij} & \text{if } i, j \notin \mathcal{B}, \\ \mathbf{A}^{\mathcal{B}}_{ij} & \text{if } i, j \notin \mathcal{A}. \end{cases} \tag{6.4}$$

Now we have the basical building blocks to describe how two mesh objects can be glued together. Assume we have two objects with node index sets $\mathcal{X}$ and $\mathcal{Y}$. Then after having performed the finite element discretization and assembly processes we have one set of algebraic equations for each object

$$\begin{bmatrix} \mathbf{A}^{\mathcal{X}} & \mathbf{0} \\ -\Delta t \mathbf{I}^{\mathcal{X}} & \mathbf{I}^{\mathcal{X}} \end{bmatrix} \begin{bmatrix} \vec{v}^{\mathcal{X}} \\ \vec{x}^{\mathcal{X}} \end{bmatrix} = \begin{bmatrix} \vec{b}^{\mathcal{X}}_v \\ \vec{b}^{\mathcal{X}}_x \end{bmatrix}, \tag{6.5a}$$

$$\begin{bmatrix} \mathbf{A}^{\mathcal{Y}} & \mathbf{0} \\ -\Delta t \mathbf{I}^{\mathcal{Y}} & \mathbf{I}^{\mathcal{Y}} \end{bmatrix} \begin{bmatrix} \vec{v}^{\mathcal{Y}} \\ \vec{x}^{\mathcal{Y}} \end{bmatrix} = \begin{bmatrix} \vec{b}^{\mathcal{Y}}_v \\ \vec{b}^{\mathcal{Y}}_x \end{bmatrix}. \tag{6.5b}$$

Gluing the two objects together means that we have a subset $\mathcal{N} \subset \mathcal{X}$ and $\mathcal{M} \subset \mathcal{Y}$ such that we have a one-to-one mapping between $\mathcal{N}$ and $\mathcal{M}$. That means after gluing with have $\mathcal{N} = \mathcal{M}$ and $(\mathcal{X} \setminus \mathcal{N}) \cap (\mathcal{Y} \setminus \mathcal{M}) = \emptyset$ and that $\mathcal{N} = \mathcal{X} \cup \mathcal{Y}$. We may now observe that gluing objects together is similar to the partitioned assembly process if one relabel the nodes in $\mathcal{M}$ with the nodes from $\mathcal{N}$ before making the global assembly. Thus we have

$$\mathbf{A}_{ij} = \begin{cases} \mathbf{A}^{\mathcal{X}}_{ij} + \mathbf{A}^{\mathcal{Y}}_{ij} & \text{if } i,j \in \mathcal{N}, \\ \mathbf{A}^{\mathcal{X}}_{ij} & \text{if } i,j \notin \mathcal{Y} \setminus \mathcal{M}, \\ \mathbf{A}^{\mathcal{Y}}_{ij} & \text{if } i,j \notin \mathcal{X} \setminus \mathcal{N}. \end{cases} \tag{6.6}$$

Let us introduce the shorthand notation $\mathcal{U} = \mathcal{X} \setminus \mathcal{N}$ and $\mathcal{V} = \mathcal{Y} \setminus \mathcal{M}$. We may now write all equations into one system of equations,

$$\begin{bmatrix} \mathbf{A}^{\mathcal{X}}_{\mathcal{U}\mathcal{U}} & \mathbf{0} & \mathbf{A}^{\mathcal{X}}_{\mathcal{U}\mathcal{N}} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}^{\mathcal{Y}}_{\mathcal{V}\mathcal{V}} & \mathbf{A}^{\mathcal{Y}}_{\mathcal{V}\mathcal{N}} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{A}^{\mathcal{X}}_{\mathcal{N}\mathcal{U}} & \mathbf{A}^{\mathcal{Y}}_{\mathcal{N}\mathcal{V}} & \mathbf{A}^{\mathcal{X}}_{\mathcal{N}\mathcal{N}} + \mathbf{A}^{\mathcal{Y}}_{\mathcal{N}\mathcal{N}} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ -\Delta t \mathbf{I}^{\mathcal{X}}_{\mathcal{U}\mathcal{U}} & \mathbf{0} & \mathbf{0} & \mathbf{I}^{\mathcal{X}}_{\mathcal{U}\mathcal{U}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\Delta t \mathbf{I}^{\mathcal{Y}}_{\mathcal{V}\mathcal{V}} & \mathbf{0} & \mathbf{0} & \mathbf{I}^{\mathcal{Y}}_{\mathcal{V}\mathcal{V}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & -\Delta t \mathbf{I}^{\mathcal{X}}_{\mathcal{N}\mathcal{N}} & \mathbf{0} & \mathbf{0} & \mathbf{I}^{\mathcal{X}}_{\mathcal{N}\mathcal{N}} \end{bmatrix} \begin{bmatrix} \vec{v}^{\mathcal{X}}_{\mathcal{U}} \\ \vec{v}^{\mathcal{X}}_{\mathcal{V}} \\ \vec{v}^{\mathcal{X}}_{\mathcal{N}} \\ \vec{x}^{\mathcal{X}}_{\mathcal{U}} \\ \vec{x}^{\mathcal{X}}_{\mathcal{V}} \\ \vec{x}^{\mathcal{X}}_{\mathcal{N}} \end{bmatrix} = \begin{bmatrix} \vec{b}^{\mathcal{X}}_{v,\mathcal{U}} \\ \vec{b}^{\mathcal{X}}_{v,\mathcal{V}} \\ \vec{b}^{\mathcal{X}}_{v,\mathcal{N}} + \vec{b}^{\mathcal{Y}}_{v,\mathcal{M}} \\ \vec{b}^{\mathcal{X}}_{x,\mathcal{U}} \\ \vec{b}^{\mathcal{X}}_{x,\mathcal{V}} \\ \vec{b}^{\mathcal{X}}_{x,\mathcal{N}} + \vec{b}^{\mathcal{Y}}_{x,\mathcal{M}} \end{bmatrix}. \tag{6.7}$$

On a side note one should observe that gluing could just as easily have been done as a pre-processing step on the computational meshes merging them into one mesh before performing the assembly process as usual.

6.2.1 A more algebraic approach

Let $\vec{x}, \vec{z} \in \mathbb{R}^N$ and $\vec{y}, \vec{w} \in \mathbb{R}^M$. We are now given two symmetric linear systems

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \begin{bmatrix} \vec{x} \\ \vec{y} \end{bmatrix} = \begin{bmatrix} \vec{a} \\ \vec{b} \end{bmatrix} \tag{6.8}$$

and

$$\begin{bmatrix} \mathbf{E} & \mathbf{F} \\ \mathbf{G} & \mathbf{H} \end{bmatrix} \begin{bmatrix} \vec{z} \\ \vec{w} \end{bmatrix} = \begin{bmatrix} \vec{c} \\ \vec{d} \end{bmatrix} \tag{6.9}$$

We may assume that $\mathbf{A}$, $\mathbf{D}$, $\mathbf{E}$ and $\mathbf{H}$ are non singular. Rewritting into one system we have

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} & \mathbf{0} & \mathbf{0} \\ \mathbf{C} & \mathbf{D} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{E} & \mathbf{F} \\ \mathbf{0} & \mathbf{0} & \mathbf{G} & \mathbf{H} \end{bmatrix} \begin{bmatrix} \vec{x} \\ \vec{y} \\ \vec{z} \\ \vec{w} \end{bmatrix} = \begin{bmatrix} \vec{a} \\ \vec{b} \\ \vec{c} \\ \vec{d} \end{bmatrix} \tag{6.10}$$

Exploiting the gluing constraint that $\vec{y} = \vec{w}$ we rewrite the system into

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} & \mathbf{0} \\ \mathbf{C} & \mathbf{D} & \mathbf{0} \\ \mathbf{0} & \mathbf{F} & \mathbf{E} \\ \mathbf{0} & \mathbf{H} & \mathbf{G} \end{bmatrix} \begin{bmatrix} \vec{x} \\ \vec{y} \\ \vec{z} \end{bmatrix} = \begin{bmatrix} \vec{a} \\ \vec{b} \\ \vec{c} \\ \vec{d} \end{bmatrix} \tag{6.11}$$

This is an over-constrainted system having M too many constraints. As one can add any together any rows of a linear system without changing the solution we add the fourth row to the second row and then we drop the fourth row,

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} & \mathbf{0} \\ \mathbf{C} & \mathbf{D}+\mathbf{H} & \mathbf{G} \\ \mathbf{0} & \mathbf{F} & \mathbf{E} \end{bmatrix} \begin{bmatrix} \vec{x} \\ \vec{y} \\ \vec{z} \end{bmatrix} = \begin{bmatrix} \vec{a} \\ \vec{b}+\vec{d} \\ \vec{c} \end{bmatrix} \tag{6.12}$$

Let us just examine that we have not changed the problem by this rewrite. We wish to know if we have a solution for

$$\mathbf{C}\vec{x} + (\mathbf{D}+\mathbf{H})\vec{y} + \mathbf{G}\vec{z} = \vec{b} + \vec{d} \tag{6.13}$$

Do we then also have a solution for the equations

$$\mathbf{C}\vec{x} + \mathbf{D}\vec{y} = \vec{b} \tag{6.14a}$$

$$\mathbf{H}\vec{y} + \mathbf{G}\vec{z} = \vec{d} \tag{6.14b}$$

To prove this we will reorganize the terms in equation (6.13) to yield

$$\left(\mathbf{C}\vec{x} + \mathbf{D}\vec{y} - \vec{b}\right) = -\left(\mathbf{H}\vec{y} + \mathbf{G}\vec{z} - \vec{d}\right) \tag{6.15}$$

In general $\mathbf{D}$ is not equal to $\mathbf{H}$ but both are non singular so the equation can only hold if left and right hand sides are both equal to zero. Thus, we find that any solution for our combined equation (6.13) will also be a solution for the two original equations (6.14).

6.3 Signorini Problem

From physics we know that at the contact surface we must have

$$-\sigma(\vec{u}^A) = \sigma(\vec{u}^B) \tag{6.16a}$$

where σ is the Cauchy stress tensor and $\vec{u}$ is the displacement field of respectively body A and body B. For frictionless contact we have for any point on the contact surface Γ_C

$$\sigma_n(\vec{u}^B) \leq 0 \tag{6.17a}$$

$$\vec{\sigma}_T(\vec{u}^A) = \vec{\sigma}_T(\vec{u}^B) = 0 \tag{6.17b}$$

$$g \geq \vec{n} \cdot \left(\vec{u}^A + \vec{u}^B\right) \tag{6.17c}$$

$$\left(\vec{n} \cdot \left(\vec{u}^A + \vec{u}^B\right) - g\right) \sigma_n(\vec{u}^B) = 0 \tag{6.17d}$$

where

$$\sigma_n = \sum_i \sum_j \vec{n}_i \sigma_{ij} \vec{n}_j = \vec{n} \cdot \vec{t} \tag{6.18a}$$

$$\vec{\sigma}_T = \sigma\vec{n} - \sigma_n\vec{n} = \vec{t} - \vec{t}\left(\vec{n}\vec{n}^T\right) \tag{6.18b}$$

and g is the initial gap (gap function). A fixed surface may exist Γ_D where

$$\vec{u} = \vec{0} \tag{6.19}$$

and a surface Γ_N under external load

$$\sigma \vec{n} = \vec{t} \tag{6.20}$$

For frictional contacts adding Coulombs law of friction means

$$\vec{u}_T = \vec{0} \Rightarrow \parallel \sigma_T \parallel \leq \mu \parallel \sigma_n \parallel \tag{6.21a}$$

$$\vec{u}_T \neq \vec{0} \Rightarrow \sigma_T \leq -\mu \parallel \sigma_n \parallel \frac{\vec{u}_T}{\parallel \vec{u}_T \parallel} \tag{6.21b}$$

Kenny Says: Oh dear, here we used displacements and not velocities – also they are not complementary to forces but rather stress tensors.

6.4 Handling of Dynamic Contact

We have two objects labelled A and B Let us abstractly introduce the contact region as a point sampling of contact points $\mathcal{K}$. Each contact point has one spatial position associated with it as well as a unit normal direction and a measure of penetration or gap.

$$\vec{p}_i, \vec{n}_i, g_i \forall i \in \mathcal{K} \tag{6.22}$$

The normal vector is by convention always pointing in the direction from A to B.

The computational meshes of A and B may not have nodal positions $\vec{x}$ collocated with the contact points.

For every contact point we may find the embedding surface triangle from one object. Thus

$$\vec{p}_i^A = \sum_{a \in \mathcal{T}^A} N_a \vec{x}_a \tag{6.23}$$

where $\mathcal{T}^A$ is the surface triangle from object A that contains $\vec{p}_i$ and N_a is the local shape function of node a. Of course we have $\vec{p}_i^A = \vec{p}_i^B$. Similariy we may find the velocity of the contact point with respect to object A

$$\dot{\vec{p}}_i^A = \sum_{a \in \mathcal{T}^A} N_a \vec{v}_a \tag{6.24}$$

and the contact traction at the contact point would be

$$\vec{t}_i^A = \sum_{a \in \mathcal{T}^A} N_a \vec{t}_a \tag{6.25}$$

6.4.1 What the F...?

From FEM discretization we have the nodal velocities (velocity update rule)

$$\mathbf{A}\vec{v} = \vec{b} + \Delta t \mathbf{L}\vec{t} \tag{6.26}$$

The **L**-matrix depends on the shape functions and the specific element type used. One can think of it as a load distribution matrix which converts nodal traction into nodal forces.

From kinematic relations (FEM approximation) we have the contact point velocity

$$\vec{u} = \mathbf{L}\vec{v} \tag{6.27}$$

Putting it together we get

$$\vec{u} = \mathbf{L}\mathbf{A}^{-1}\left(\vec{b} + \Delta t \mathbf{L}\vec{t}\right) = \Delta t \mathbf{L}\mathbf{A}^{-1}\mathbf{L}\vec{t} + \mathbf{L}\mathbf{A}^{-1}\vec{b} \tag{6.28}$$

Non penetartion constraint and non-sticking normal force

$$g_i \geq 0 \quad \perp \quad \vec{n}_i^T \vec{t}_i \geq 0 \tag{6.29}$$

where g_i is the gap funciton of the ith contact point, $\vec{n}_i$ is the contact normal of the ith contact point and $\vec{t}_i$ is the traction force at the ith contact point.

From time discretization we have the position update rule

$$\vec{x}^t + 1 = \vec{x}^t + \Delta t \vec{v} \tag{6.30}$$

How do we define the gap function so we can replace it by a Taylor approximation?

$$g_i = g_i^0 + \Delta t \nabla_{\vec{x}} g_i \vec{v} \geq 0 \quad \perp \quad \vec{n}_i^T \vec{t}_i \geq 0 \tag{6.31}$$

if

$$g \approx \vec{n}^T(\vec{p}_i^A - \vec{p}_i^B) \tag{6.32}$$

then

$$\nabla_x g_i = (\nabla_x \vec{p}_i^A - \nabla_x \vec{p}_i^B)\vec{n} \tag{6.33}$$

and by FEM approximation

$$\nabla_x p_i = \sum_k \nabla_x N_a x_a = \mathbf{J}\vec{x} \tag{6.34}$$

so

$$g_i^0 + \vec{n}_i^T \nabla_x P_i \vec{v} \geq 0 \quad \perp \quad \vec{n}_i^T \vec{t}_i \geq 0 \tag{6.35}$$

Chapter 7

Previous Work in Computer Graphics

Crack development and propagation was studied in [O'Brien and Hodgins(1999)] where stress tensors are used to form cracks during deformation. A finite element method is used with linear shape functions on tetrahedral elements. The formulation is based on the Green strain tensor and the Cauchy stress tensor split into elastic and viscous terms further a lumped mass matrix is used. During simulation principal stresses are examined and used to split the stress tensor into tensile and compressive components. These components are used to form a separation tensor from which a crack plane can be determined. There is not a lot of details on the finite element method, instead the book by Cook et. al. [Cook et al.(2007)Cook, Malkus, Plesha, and Witt] referenced. [O'Brien et al.(2002)O'Brien, Bargteil, and Hodgins] extends [O'Brien and Hodgins(1999)] to handle ductile fracture.

In [Fisher and Lin(2001)] penetration depth estimation is done for elastic objects represented by tetrahedral meshes. Initially a distance field is generated on the tetrahedral mesh using a fast marching level set method. During simulation nodal positions are looked up in the deformed distance field using linear shape functions and the distance values are used as penetration depth estimations. Contact forces are computed using a penalty based method and deformations are computed using a finite element method.

In [Hirota et al.(2001)Hirota, Fisher, State, Lee, and Fuchs] quasi static simulation is done which is based on the first Piola–Kirchhoff stress tensor and the finite element method. A penalty force method for computing normal forces between elastic objects is presented. They use a precomputed material depth which is interpolated during motion to provide a continuous gap function. Boundary triangles and tetrahedrons are then intersected and the intersection is triangulated and contact force integrals are evaluated and assembled over each triangle.

Distance values are used in both [Fisher and Lin(2001)] and [Hirota et al.(2001)Hirota, Fisher, State, Lee, and Fu to compute penalty forces. In the former work distance fields are updated during simulation and distance values are used in a spring like penalty force method. In the latter work distance values are precomputed but not updated during simulation. The penalty forces are obtained by integrating the intersection volumes. The two works are similar but not compared to each other.

In [Debunne et al.(2001)Debunne, Desbrun, Cani, and Barr] real time deformation is considered where space and time adaptiveness are exploited. The finite element method is used based on the Green strain tensor, the Cauchy stress tensor and linear elasticity. The authors add a level of detail hierarchy of tetrahedral meshes using linear shape functions. The paper cite [O'Brien and Hodgins(1999)] for details on the the force formula.

In [Müller et al.(2001)Müller, McMillan, Dorsey, and Jagnow] the focus is on real time deformation and fracturing. They apply a finite element method using tetrahedral meshes and linear shape functions. The model is very similar to [O'Brien and Hodgins(1999)]. Only a static analysis is done. To deal with the dynamics a hybrid simulation is applied. Here free floating parts are treated as rigid bodies and only as elastic objects during collisions. It seems that all work based on [O'Brien and Hodgins(1999)] assumes that the Cauchy stress tensor is assumed to be work conjugate to the Green strain tensor which is not correct.

In [Müller et al.(2002)Müller, Dorsey, McMillan, Jagnow, and Cutler] the first version of the stiffness warping method is introduced – the corotational linear finite element method. In this paper a per vertex rotation matrix is used for the warping which causes ghost forces. The rotations are found by a least square error approach, where three outgoing edges are selected and the best rotation of material edges into spatial edges are found.

[Hirota(2002)] developed a finite element method for static equilibrium simulation using second Piola–Kirchhoff stress tensor and Green strain tensor. Several different materials are used too. The work follows Bonet and Wood [Bonet and Wood(2000)] in spirit. The essential contribution is a continuous gap function for computing contact forces.

In [Teran et al.(2003)Teran, Blemker, Hing, and Fedkiw] a finite volume method (FVM) is derived. Here it is assumed that one has constant strain per tetrahedral element. This corresponds to using linear shape functions. Elastic forces are treated explicitly in the time integration rather than using a full implicit method. This corresponds to the semi implicit Euler time stepping scheme we know from fixed time stepping methods in rigid body dynamics. The paper actually does not address boundary conditions and explains mainly how to deal with control volumes not on the boundary of the computational mesh. The paper shows that this finite volume method is numerically the same as a finite element method using linear shape functions on tetrahedral elements.

In [Hauth et al.(2003)Hauth, Groß, and Straßer] the focus is interactive solid dynamics using a viscoelastic model and parameters from measurements in a finite element method setting. The Cauchy stress tensor is used. The Green strain is used as the work conjugate when writing the elastic energy. This is in contradiction with continuum mechanics [Lai et al.(2009)Lai, Rubin, and Krempl].

In [Irving et al.(2004)Irving, Teran, and Fedkiw] the finite volume/element method from [Teran et al.(2003)Teran, Blemker, Hing, and Fedkiw] is used. The work deals with handling of inversion and degenerated tetrahedral elements. These problems are detected by looking at the sign of the determinant of the deformation gradient. The approach works by diagonalizing the deformation gradient $\mathbf{F} = \mathbf{U}\hat{\mathbf{F}}\mathbf{V}^T$. This is done in three steps. First the eigenvalue decomposition is done of $\mathbf{C} = \mathbf{F}^T\mathbf{F} = \mathbf{V}\hat{\mathbf{F}}^2\mathbf{V}^2$. Second if $\mathbf{V}$ is a reflection i.e. $\det(\mathbf{V}) = -1$ the sign of one of the columns is flipped. Third, one computes $\mathbf{U} = \mathbf{F}\mathbf{V}\hat{\mathbf{F}}^{-1}$ and examines if $\det(\mathbf{U}) = -1$ then one negates the minimum element of $\hat{\mathbf{F}}$ and the corresponding column of $\mathbf{U}$. After having done the diagonalization the constitutive laws are restated and even remodeled in terms of diagonalized stresses and deformation gradients. This comes with one disadvantage as one can not simply use constitutive laws known from continuum mechanics, one has to restate and reformulate them in this new diagonalized setting. Elastic forces are treated explicitly whereas damping forces are treated implicitly.

In [Teran et al.(2005a)Teran, Sifakis, Blemker, Ng-Thow-Hing, Lau, and Fedkiw] the work [Teran et al.(2003)Teran, Blemker, Hing, and Fedkiw] and [Irving et al.(2004)Irving, Teran, and Fedkiw] are applied to modeling muscles as well as numerous other level set method techniques. [Irving et al.(2006)Irving, Teran, and Fedkiw] is an extended version of [Irving et al.(2004)Irving, Teran, and Fedkiw]. Here arbitrary Lagrangian Eulerian (ALE) scheme is introduced using an isoparametric representation. The result is a more general formulation that covers hexahedral elements.

In [Müller and Gross(2004)] the vertex-based stiffness warping method from [Müller et al.(2002)Müller, Dorsey, McMillan, Jagnow, and Cutler] is improved to an element-based method. The method is based on using polar decomposition for extracting the rotational part of the deformation gradient.

A point-based approach for simulating large deformations similar to smoothed particule hydrodynamics (SPH) is presented in [Müller et al.(2004)Müller, Keiser, Nealen, Pauly, Gross, and Alexa]. The model is based on Cauchy stress and Green strain tensors. These are taken to be work conjugate which contradicts definitions from continuum mechanics [Lai et al.(2009)Lai, Rubin, and Krempl].

[Barbič and James(2005)] presents a model dimension reduction technique for Saint Venant–Kirchhoff materials leading to efficient sub-space integration. Rayleigh damping is used.

[Choi and Ko(2005)] introduced the modal warping method which extends modal analysis to handle large rotational deformations in a manner similar to stiffness warping. A vertex based rotation is used in modal space. The work is based on the observation that for small displacements it is possible to keep track of the rotation of a material point by calculating the curl of the displacement field. Rayleigh damping is employed.

In [Wu and Heng(2005)] the context is surgical simulation. The authors use a finite element method based on linear elasticity for small displacements, Cauchy stress and strain tensors. A hybrid model is developed dividing the tissue into an operational region and a non-operational region.

A non-linear quasi static finite element method is developed in [Teran et al.(2005b)Teran, Sifakis, Irving, and Fedk based on the work [Teran et al.(2003)Teran, Blemker, Hing, and Fedkiw] and [Irving et al.(2004)Irving, Teran, and Fedkiw]. They discuss linearization of elastic forces in the context of diagonalization and further ensures positive definiteness of the resulting stiffness matrix.

In [Duriez et al.(2006)Duriez, Dubois, Kheddar, and Andriot] a linear complementarity problem (LCP) formulation is derived for the contact problem between deformable models simulated using a linear finite element method. Further a Gauss–Seidel like algorithm is presented for computing contact force solutions.

In [Galoppo et al.(2007)Galoppo, Otaduy, Tekin, Gross, and Lin] a layered dynamics approach is used to model skin deformations for skeleton based characters. The actual skin deformation is computed using a linear finite element method.

In [Irving et al.(2007)Irving, Schroeder, and Fedkiw] volume conservation is treated by adding incompressibility constraint to the finite volume/element method from [Teran et al.(2003)Teran, Blemker, Hing, and Fedkiw]. A pressure field is solved by requiring the velocity field to be divergence free. Locking is avoided by using composite tetrahedral elements. A finite volume method discretization is done of the continuity equation using median vertex centered control volumes and applying Dirichlet boundary conditions setting the pressure to zero outside the volume.

In [Otaduy et al.(2007)Otaduy, Germann, Redon, and Gross] adaptive multigrid is applied to the corotational linear elasticity finite element method [Müller and Gross(2004)] with implicit integration.

In [Otaduy and Gross(2007)] constraint-based contact forces for haptic is considered. Here the corotational linear elastic finite element method is applied.

The corotational linear elasticity finite element method is applied in [Saupin et al.(2007)Saupin, Duri
where a mesh hierarchy is created as the basis for a multigrid solver.

In [Spillmann et al.(2007)Spillmann, Becker, and Teschner], they use both linear and non-linear finite element methods to demonstrate a scheme for handling collision of deformable objects. Not much detail is given in regards to specifics of the finite element methods.

[Bargteil et al.(2007)Bargteil, Wojtan, Hodgins, and Turk] introduced an enhanced volume preserving plasticity model for the Lagrangian finite elements method presented by [Irving et al.(2004)Irving, Teran, and Fedkiw]. Further re-meshing is used to deal with ill conditioning from large plastic flows. A multiplicative plasticity model is used.

Inversion handling for corotational linear finite element method is treated in [Schmedding and Teschner(2008)]. The rotation returned by polar decomposition is improper for an inverted element and is therefore corrected.

In [Georgii and Westermann(2008)] a quaternion-based approach is presented for computing element rotations for the corotational linear finite element method. The authors present an efficient sparse block matrix product for using multigrid methods.

A discontinuous Galerkin finite element method based on corotated linear elasticity were presented in [Kaufmann et al.(2009)Kaufmann, Martin, Botsch, and Gross].

Recent work explores example-based elastic materials [Martin et al.(2011)Martin, Thomaszewski, Grinspun, and G
Here a discrete Green strain field descriptor is used to represent a deformation example. The work present a method for interpolating motion between these discrete Green strain descriptors which is used to define a concept of a manifold and a projection onto this manifold. From these building blocks resulting elastic forces can be found.

Simulation of coupled rigid and deformable objects were dealt with in [Miguel and Otaduy(2011)] by applying a partitioning to the linear complementarity problem (LCP) formulation of the contact forces. The partitioning allows for a reformulation of the LCP taking advantage of separating contact sets acting only on rigid bodies.

Corotational elasticity is used for interactive character skinning in [McAdams et al.(2011)McAdams, Zhu, Selle, En
The authors apply a hexahedral grid and a modified Newton method is outlined that takes care in ensuring positive definiteness of the tangent stiffness matrix. The framework is further accelerated by applying multigrid methods where a matrix-free approach is used taking advantage of the hexahedral grid in the coarsening.

A new method for meshless models for complex deformable solids was presented in [Faure et al.(2011)Faure, Gilles, Bousquet, and Pai]. Here a small number of reference frames are used with associated material-aware shape functions which allow for heterogenous material properties. Several different kind of materials are examined and interactive animation rates are achieved.

Mixed implicit and explicit time integration of deformable models were studied in [Fierz et al.(2011)Fierz, Spillmann, and Harders]. The authors perform an element-wise analysis of the stiffness matrix to identity which elements should be implicitly time integrated and which should not resulting in a so called IMEX solver.

In [Kim and James(2011)] non-linear finite element method for deformable models is used for character skinning. The authors apply a multi-domain subspace deformation method. The main contribution is a "fast sandwich transform" technique that solves the coupling between subdomains.

An Eulerian approach to solid simulation is presented in [Levin et al.(2011)Levin, Litven, Jones, Sueda, and Pai]. Here a single solid object lives in its own Eulerian grid. The idea is simply to advect the material coordinate field on the grid using an upwind scheme while the simulation runs. The authors apply Green strain and follow the method of [Teran et al.(2003)Teran, Blemker, Hing, and Fedkiw] for their discretization. Contact is solved using Gauss' principle of least constraint no friction is present although the authors speculate that staggering can be used to add friction.

The practice and experience paper by Sin et. al. [Sin et al.(2013)Sin, Schroeder, and Barbič] surveys several methods used in the field of computer graphics and presents numerical results from the middleware library VEGA.

Bibliography

[Baraff and Witkin(1998)] D. Baraff and A. Witkin. Large steps in cloth simulation. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 43–54, New York, NY, USA, 1998. ACM.

[Barbič and James(2005)] J. Barbič and D. L. James. Real-time subspace integration for st. venant-kirchhoff deformable models. *ACM Trans. Graph.*, 24:982–990, July 2005.

[Bargteil et al.(2007)Bargteil, Wojtan, Hodgins, and Turk] A. W. Bargteil, C. Wojtan, J. K. Hodgins, and G. Turk. A finite element method for animating large viscoplastic flow. *ACM Trans. Graph.*, 26, July 2007.

[Bergou et al.(2008)Bergou, Wardetzky, Robinson, Audoly, and Grinspun] M. Bergou, M. Wardetzky, S. Robinson, B. Audoly, and E. Grinspun. Discrete elastic rods. *ACM Trans. Graph.*, 27(3):1–12, 2008.

[Bonet and Wood(2000)] J. Bonet and R. D. Wood. *Nonlinear continuum mechanics for finite element analysis*. Cambridge University Press, 2000.

[Botsch et al.(2006)Botsch, Pauly, Gross, and Kobbelt] M. Botsch, M. Pauly, M. Gross, and L. Kobbelt. Primo: coupled prisms for intuitive surface modeling. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, SGP '06, pages 11–20, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[Bridson et al.(2002)Bridson, Fedkiw, and Anderson] R. Bridson, R. Fedkiw, and J. Anderson. Robust treatment of collisions, contact and friction for cloth animation. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 594–603, New York, NY, USA, 2002. ACM.

[Chadwick(1999)] P. Chadwick. *Continuum mechanics: concise theory and problems*. Courier Dover Publications, 1999.

[Choi and Ko(2005)] M. G. Choi and H.-S. Ko. Modal warping: Real-time simulation of large rotational deformation and manipulation. *IEEE Transactions on Visualization and Computer Graphics*, 11:91–101, January 2005.

[Cook et al.(2007)Cook, Malkus, Plesha, and Witt] R. D. Cook, D. S. Malkus, M. E. Plesha, and R. J. Witt. *Concepts and Applications of Finite Element Analysis*. John Wiley & Sons, 2007.

[Debunne et al.(2001)Debunne, Desbrun, Cani, and Barr] G. Debunne, M. Desbrun, M.-P. Cani, and A. H. Barr. Dynamic real-time deformations using space & time adaptive sampling. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, pages 31–36, New York, NY, USA, 2001. ACM.

[Duriez et al.(2006)Duriez, Dubois, Kheddar, and Andriot] C. Duriez, F. Dubois, A. Kheddar, and C. Andriot. Realistic haptic rendering of interacting deformable objects in virtual environments. *IEEE Transactions on Visualization and Computer Graphics*, 12:36–47, January 2006.

[Erleben(2011a)] K. Erleben. Hyper elasticity for small and large deformations based on a lagrangian formulation. Unpublished, 2011a.

[Erleben(2011b)] K. Erleben. The tangent stiffness matrix for for the implicit nonlinear finite element method. Unpublished, 2011b.

[Erleben(2011c)] K. Erleben. Preliminaries on tensor notation and continuum mechanics. Unpublished, 2011c.

[Erleben(2011d)] K. Erleben. Implementation tricks and tips for finite element modeling of hyper elastic materials. Unpublished, 2011d.

[Erleben(2011e)] K. Erleben. hyper-sim. Published online at GitHub https://github.com/erleben/hyper-sim, November 2011e. Open source project for simulation methods for hyper elastic materials in physics-based animation.

[Faure et al.(2011)Faure, Gilles, Bousquet, and Pai] F. Faure, B. Gilles, G. Bousquet, and D. K. Pai. Sparse meshless models of complex deformable solids. *ACM Trans. Graph.*, 30:73:1–73:10, August 2011. doi: http://doi.acm.org/10.1145/2010324.1964968. URL `http://doi.acm.org/10.1145/2010324.1964968`.

[Fierz et al.(2011)Fierz, Spillmann, and Harders] B. Fierz, J. Spillmann, and M. Harders. Element-wise mixed implicit-explicit integration for stable dynamic simulation of deformable objects. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '11, pages 257–266, New York, NY, USA, 2011. ACM. doi: http://doi.acm.org/10.1145/2019406.2019440. URL `http://doi.acm.org/10.1145/2019406.2019440`.

[Fisher and Lin(2001)] S. Fisher and M. C. Lin. Deformed distance fields for simulation of non-penetrating flexible bodies. In *Proceedings of the Eurographic workshop on Computer animation and simulation*, pages 99–111, New York, NY, USA, 2001. Springer-Verlag New York, Inc.

[Galoppo et al.(2007)Galoppo, Otaduy, Tekin, Gross, and Lin] N. Galoppo, M. A. Otaduy, S. Tekin, M. H. Gross, and M. C. Lin. Soft articulated characters with fast contact handling. *Comput. Graph. Forum*, 26(3): 243–253, 2007.

[Galoppo et al.(2009)Galoppo, Otaduy, Moss, Sewall, Curtis, and Lin] N. Galoppo, M. A. Otaduy, W. Moss, J. Sewall, S. Curtis, and M. C. Lin. Controlling deformable material with dynamic morph targets. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games*, I3D '09, pages 39–47, New York, NY, USA, 2009. ACM.

[Georgii and Westermann(2008)] J. Georgii and R. Westermann. Corotated finite elements made fast and stable. In *Proceedings of the 5th Workshop On Virtual Reality Interaction and Physical Simulation*, pages 11–19, 2008.

[Grinspun(2006)] E. Grinspun. A discrete model of thin shells. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Courses*, pages 14–19, New York, NY, USA, 2006. ACM.

[Hauth et al.(2003)Hauth, Groß, and Straßer] M. Hauth, J. Groß, and W. Straßer. Interactive physically based solid dynamics. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '03, pages 17–27, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[Hirota(2002)] G. Hirota. *An improved finite element contact model for anatomical simulations*. PhD thesis, The University of North Carolina at Chapel Hill, 2002.

[Hirota et al.(2001)Hirota, Fisher, State, Lee, and Fuchs] G. Hirota, S. Fisher, A. State, C. Lee, and H. Fuchs. An implicit finite element method for elastic solids in contact. In *Proceedings of the Computer Animation*, pages 136–146, 2001.

[Irving et al.(2004)Irving, Teran, and Fedkiw] G. Irving, J. Teran, and R. Fedkiw. Invertible finite elements for robust simulation of large deformation. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '04, pages 131–140, Aire-la-Ville, Switzerland, Switzerland, 2004. Eurographics Association.

[Irving et al.(2006)Irving, Teran, and Fedkiw] G. Irving, J. Teran, and R. Fedkiw. Tetrahedral and hexahedral invertible finite elements. *Graph. Models*, 68:66–89, March 2006.

[Irving et al.(2007)Irving, Schroeder, and Fedkiw] G. Irving, C. Schroeder, and R. Fedkiw. Volume conserving finite element simulations of deformable models. *ACM Trans. Graph.*, 26, July 2007.

[Joe(1991)] B. Joe. GEOMPACK - a software package for the generation of meshes using geometric algorithms. *Advances in Engineering Software*, 13: 325–331, 1991.

[Kaufmann et al.(2009)Kaufmann, Martin, Botsch, and Gross] P. Kaufmann, S. Martin, M. Botsch, and M. Gross. Flexible simulation of deformable models using discontinuous galerkin fem. *Graph. Models*, 71:153–167, July 2009.

[Kim and James(2011)] T. Kim and D. L. James. Physics-based character skinning using multi-domain subspace deformations. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '11, pages 63–72, New York, NY, USA, 2011. ACM. doi: http://doi.acm.org/10.1145/2019406.2019415. URL `http://doi.acm.org/10.1145/2019406.2019415`.

[Lai et al.(2009)Lai, Rubin, and Krempl] W. M. Lai, D. Rubin, and E. Krempl. *Introduction to Continuum Mechanics*. Butterworth-Heinemann, fourth edition edition, 2009.

[Lanczos(1986)] C. Lanczos. *The variational principles of mechanics*. Dover Publications, 4 edition edition, March 1986.

[Levin et al.(2011)Levin, Litven, Jones, Sueda, and Pai] D. I. W. Levin, J. Litven, G. L. Jones, S. Sueda, and D. K. Pai. Eulerian solid simulation with contact. *ACM Trans. Graph.*, 30:36:1–36:10, August 2011. doi: http://doi.acm.org/10.1145/2010324.1964931. URL `http://doi.acm.org/10.1145/2010324.1964931`.

[Martin et al.(2011)Martin, Thomaszewski, Grinspun, and Gross] S. Martin, B. Thomaszewski, E. Grinspun, and M. Gross. Example-based elastic materials. *ACM Trans. Graph.*, 30:72:1–72:8, August 2011. doi: http://doi.acm.org/10.1145/2010324.1964967. URL `http://doi.acm.org/10.1145/2010324.1964967`.

[McAdams et al.(2011)McAdams, Zhu, Selle, Empey, Tamstorf, Teran, and Sifakis] A. McAdams, Y. Zhu, A. Selle, M. Empey, R. Tamstorf, J. Teran, and E. Sifakis. Efficient elasticity for character skinning with contact and collisions. *ACM Trans. Graph.*, 30:37:1–37:12, August 2011. doi: http://doi.acm.org/10.1145/2010324.1964932. URL `http://doi.acm.org/10.1145/2010324.1964932`.

[Miguel and Otaduy(2011)] E. Miguel and M. A. Otaduy. Efficient simulation of contact between rigid and deformable objects. In *ECCOMAS - Multibody Dynamics*, 2011. URL `http://www.gmrv.es/Publications/2011/MO11`.

[Müller and Gross(2004)] M. Müller and M. Gross. Interactive virtual materials. In *Proceedings of Graphics Interface 2004*, GI '04, pages 239–246, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2004. Canadian Human-Computer Communications Society.

[Müller et al.(2001)Müller, McMillan, Dorsey, and Jagnow] M. Müller, L. McMillan, J. Dorsey, and R. Jagnow. Real-time simulation of deformation and fracture of stiff materials. In *Proceedings of the Eurographic workshop on Computer animation and simulation*, pages 113–124, New York, NY, USA, 2001. Springer-Verlag New York, Inc.

[Müller et al.(2002)Müller, Dorsey, McMillan, Jagnow, and Cutler] M. Müller, J. Dorsey, L. McMillan, R. Jagnow, and B. Cutler. Stable real-time deformations. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '02, pages 49–54, New York, NY, USA, 2002. ACM.

[Müller et al.(2004)Müller, Keiser, Nealen, Pauly, Gross, and Alexa] M. Müller, R. Keiser, A. Nealen, M. Pauly, M. Gross, and M. Alexa. Point based animation of elastic, plastic and melting objects. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '04, pages 141–151, Aire-la-Ville, Switzerland, Switzerland, 2004. Eurographics Association.

[Müller et al.(2005)Müller, Heidelberger, Teschner, and Gross] M. Müller, B. Heidelberger, M. Teschner, and M. Gross. Meshless deformations based on shape matching. *ACM Trans. Graph.*, 24:471–478, July 2005.

[Müller et al.(2007)Müller, Heidelberger, Hennix, and Ratcliff] M. Müller, B. Heidelberger, M. Hennix, and J. Ratcliff. Position based dynamics. *J. Vis. Comun. Image Represent.*, 18:109–118, April 2007.

[Nealen et al.(2006)Nealen, Müller, Keiser, Boxerman, and Carlson] A. Nealen, M. Müller, R. Keiser, E. Boxerman, and M. Carlson. Physically based deformable models in computer graphics. *Computer Graphics Forum*, 25(4):809–836, Dec. 2006.

[Newmark(1959)] N. M. Newmark. A method of computation for structural dynamics. *ASCE Journal of the Engineering Mechanics Division*, pages 67–94, 1959.

[Nocedal and Wright(1999)] J. Nocedal and S. J. Wright. *Numerical optimization*. Springer Series in Operations Research. Springer-Verlag, New York, 1999. ISBN 0-387-98793-2.

[O'Brien and Hodgins(1999)] J. F. O'Brien and J. K. Hodgins. Graphical modeling and animation of brittle fracture. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '99, pages 137–146, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[O'Brien et al.(2002)O'Brien, Bargteil, and Hodgins] J. F. O'Brien, A. W. Bargteil, and J. K. Hodgins. Graphical modeling and animation of ductile fracture. *ACM Trans. Graph.*, 21:291–294, July 2002.

[Otaduy and Gross(2007)] M. A. Otaduy and M. Gross. Transparent rendering of tool contact with compliant environments. In *Proceedings of the Second Joint EuroHaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, pages 225–230, Washington, DC, USA, 2007. IEEE Computer Society.

[Otaduy et al.(2007)Otaduy, Germann, Redon, and Gross] M. A. Otaduy, D. Germann, S. Redon, and M. Gross. Adaptive deformations with fast tight bounds. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '07, pages 181–190, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.

[Persson and Strang(2004)] P.-O. Persson and G. Strang. A simple mesh generator in matlab. *SIAM Review*, 46(2):329–345, June 2004.

[Quek and Liu(2003)] S. S. Quek and G. R. Liu. *Finite Element Method: A Practical Course: A Practical Course*. Butterworth-Heinemann, May 2003.

[Reddy(2008)] J. N. Reddy. *An introduction to continuum mechanics:with applications*. Cambridge University Press, 2008.

[Saad(2003)] Y. Saad. *Iterative methods for sparse linear systems*. Society for Industrial Mathematics, 2003.

[Sarah F. F. Gibson(1997)] B. M. Sarah F. F. Gibson. A survey of deformable modeling in computer graphics. Technical Report TR97-19, Mitsubishi Electric Research Laboratories, December 1997.

[Saupin et al.(2007)Saupin, Duriez, and Grisoni] G. Saupin, C. Duriez, and L. Grisoni. Embedded multigrid approach for real-time volumetric deformation. In *Proceedings of the 3rd international conference on Advances in visual computing - Volume Part I*, ISVC'07, pages 149–159, Berlin, Heidelberg, 2007. Springer-Verlag.

[Schmedding and Teschner(2008)] R. Schmedding and M. Teschner. Inversion handling for stable deformable modeling. *The Visual Computer (CGI 2008 special issue)*, 24(7–9):625–633, 2008.

[Selle et al.(2008)Selle, Lentine, and Fedkiw] A. Selle, M. Lentine, and R. Fedkiw. A mass spring model for hair simulation. *ACM Trans. Graph.*, 27(3): 1–11, 2008.

[Shewchuk(2002)] J. R. Shewchuk. What is a good linear element? interpolation, conditioning, and quality measures. In *Eleventh International Meshing Roundtable*, pages 115–126, September 2002.

[Sin et al.(2013)Sin, Schroeder, and Barbič] F. S. Sin, D. Schroeder, and J. Barbič. Vega: Nonlinear fem deformable object simulator. *Computer Graphics Forum*, 31(1), 2013.

[Spencer(2004)] A. J. M. Spencer. *Continuum mechanics*. Courier Dover Publications, 2004.

[Spillmann et al.(2007)Spillmann, Becker, and Teschner] J. Spillmann, M. Becker, and M. Teschner. Non-iterative computation of contact forces for deformable objects. *Journal of WSCG*, 15(1–3), 2007.

[Teran et al.(2003)Teran, Blemker, Hing, and Fedkiw] J. Teran, S. Blemker, V. N. T. Hing, and R. Fedkiw. Finite volume methods for the simulation of skeletal muscle. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '03, pages 68–74, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[Teran et al.(2005a)Teran, Sifakis, Blemker, Ng-Thow-Hing, Lau, and Fedkiw] J. Teran, E. Sifakis, S. S. Blemker, V. Ng-Thow-Hing, C. Lau, and R. Fedkiw. Creating and simulating skeletal muscle from the visible human data set. *IEEE Transactions on Visualization and Computer Graphics*, 11:317–328, May 2005a.

[Teran et al.(2005b)Teran, Sifakis, Irving, and Fedkiw] J. Teran, E. Sifakis, G. Irving, and R. Fedkiw. Robust quasistatic finite elements and flesh simulation. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '05, pages 181–190, New York, NY, USA, 2005b. ACM.

[Terzopoulos et al.(1987)Terzopoulos, Platt, Barr, and Fleischer] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. *SIGGRAPH Comput. Graph.*, 21:205–214, August 1987.

[Teschner et al.(2004)Teschner, Heidelberger, Muller, and Gross] M. Teschner, B. Heidelberger, M. Muller, and M. Gross. A versatile and robust model for geometrically complex deformable solids. In *CGI '04: Proceedings of the Computer Graphics International*, pages 312–319, Washington, DC, USA, 2004. IEEE Computer Society.

[Wicke et al.(2010)Wicke, Ritchie, Klingner, Burke, Shewchuk, and O'Brien] M. Wicke, D. Ritchie, B. M. Klingner, S. Burke, J. R. Shewchuk, and J. F. O'Brien. Dynamic local remeshing for elastoplastic simulation. *ACM Trans. Graph.*, 29:49:1–49:11, July 2010.

[Wriggers(2002)] P. Wriggers. *Computational Contact Mechanics*. John Wiley & Sons, Ltd., 2002.

[Wu and Heng(2005)] W. Wu and P.-A. Heng. An improved scheme of an interactive finite element model for 3D soft-tissue cutting and deformation. *The Visual Computer*, 21(8-10):707–716, Sept. 2005.

[Zhu et al.(2010)Zhu, Sifakis, Teran, and Brandt] Y. Zhu, E. Sifakis, J. Teran, and A. Brandt. An efficient multigrid method for the simulation of high-resolution elastic solids. *ACM Trans. Graph.*, 29:16:1–16:18, April 2010.

[Zienkiewicz and Taylor(2000)] O. C. Zienkiewicz and R. L. Taylor. *The Finite Element Method Volume 1: The Basis*. Butterworth-Heinemann, fifth edition edition, 2000.

Appendix A

Appendix

A.1 Supplementary Details and Preliminaries

This appendix seeks to provide detailed arguments and refreshing of essential mathematical knowledge that we felt did not fit well into the main text.

A.1.1 Mathematical Preliminaries

The Gauss divergence theorem can be written as

$$\int_v \nabla \cdot \vec{a} dv = \int_{\partial v} \vec{a} \cdot \vec{n} ds, \tag{A.1}$$

where $\vec{a}$ is any vector field defined over the closed volume v and ∂v is the surface of the volume with outward unit normal $\vec{n}$ defined everywhere. A second order tensor is a linear mapping that maps a vector into a vector. We write $\vec{c} = \mathbf{A}\vec{b}$ where $\mathbf{A}$ is a second order tensor applied to the vector $\vec{b}$. One may define a double contraction operator between two second order tensors $\mathbf{A}$ and $\mathbf{B}$. The double contraction is defined as

$$\mathbf{A} : \mathbf{B} = \text{tr}\left(\mathbf{A}^T \mathbf{B}\right) = \sum_{ij} \mathbf{A}_{ij} \mathbf{B}_{ij}. \tag{A.2}$$

This is a very convenient notation for writing compact formulas. Observe that $\mathbf{A} : \mathbf{B}^T = \mathbf{A}^T : \mathbf{B}$. We wish to define the divergence of a second order tensor. From the definition of divergence of a vector we have

$$\nabla \cdot \vec{c} = \sum_i \partial_i \vec{c}_i. \tag{A.3}$$

We substitute $\vec{c} = \mathbf{A}\vec{b}$. This gives

$$\nabla \cdot (\mathbf{A}\vec{b}) = \sum_i \partial_i (\mathbf{A}\vec{b})_i \tag{A.4a}$$

$$= \sum_i \partial_i \left(\sum_j \mathbf{A}_{ij} \vec{b}_j \right) \tag{A.4b}$$

$$= \underbrace{\left(\sum_i \sum_j (\partial_i \mathbf{A}_{ij}) \hat{e}_j \right)}_{\nabla \cdot \mathbf{A}} \cdot \vec{b} + \mathbf{A} : (\nabla \vec{b})^T \tag{A.4c}$$

$$= (\nabla \cdot \mathbf{A}) \cdot \vec{b} + \mathbf{A} : (\nabla \vec{b})^T. \tag{A.4d}$$

If $\mathbf{A}$ is symmetric this simplifies to

$$\nabla \cdot (\mathbf{A}\vec{b}) = (\nabla \cdot \mathbf{A}) \cdot \vec{b} + \mathbf{A} : (\nabla \vec{b}). \tag{A.5}$$

A.1.2 Dealing with Volume Integrals

The Cauchy equation is usually defined in spatial coordinates. Thus, integration over the spatial volume and multiplication of the admissible test function yields,

$$\int_v \left(\rho \ddot{\vec{x}} - \nabla \cdot \sigma - \vec{b} \right) \cdot \vec{w} dv = 0, \tag{A.6}$$

where $\rho(\vec{x})$, $\sigma(\vec{x})$, $\vec{w}(\vec{x})$, and $\vec{b}(\vec{x})$ are functions of spatial coordinates and not material coordinates. Using $dv = jdV$ we make a change of variables, $\vec{x} = \vec{\phi}(\vec{X}, t)$,

$$\int_V \left(\rho \ddot{\vec{x}} - \nabla \cdot \sigma - \vec{b} \right) \cdot \vec{w} j dV = 0 \tag{A.7}$$

where $\rho(\vec{\phi}(\vec{X}, t))$, $\sigma(\vec{\phi}(\vec{X}, t))$, $\vec{w}(\vec{\phi}(\vec{X}, t))$ and $\vec{b}(\vec{\phi}(\vec{X}, t))$ now are functions of material coordinates. Finally, we divide both the left and right hand side of the equation with the determinant of the deformation gradient. This yields the volume integrals we use for deriving the corotational linear finite element method,

$$\int_V \left(\rho \ddot{\vec{x}} - \nabla \cdot \sigma - \vec{b} \right) \cdot \vec{w} dV = 0. \tag{A.8}$$

A.1.3 The Small Strain Tensor

One may find two versions of the small strain tensor (Cauchy strain) tensor,

$$\varepsilon_0 = \frac{1}{2}(\nabla_0 \vec{u} + \nabla_0 \vec{u}^T), \tag{A.9a}$$

$$\varepsilon = \frac{1}{2}(\nabla \vec{u} + \nabla \vec{u}^T). \tag{A.9b}$$

The first is defined in terms of material derivatives and the second defined in terms of spatial derivatives. We have added a subscript to the former to distinguish between them. The displacement field $\vec{u}$ and the deformation gradient $\mathbf{F} = \nabla_0 \vec{x} = \mathbf{I} + \nabla_0 \vec{u}(\vec{X})$ are functions of $\vec{X}$ and not $\vec{x}$. One may obtain both a spatial and material version of the Cauchy strain tensor by linearization of the Euler strain tensor

$$\mathbf{e} = \frac{1}{2}(\mathbf{I} - (\mathbf{F}\mathbf{F}^T)^{-1}) \tag{A.10}$$

or the Green strain tensor

$$\mathbf{E} = \frac{1}{2}(\mathbf{F}^T\mathbf{F} - \mathbf{I}). \tag{A.11}$$

Thus, $\varepsilon_0 \approx \mathbf{e}$ and $\varepsilon \approx \mathbf{E}$. In the small strain regime one may for all practical matters consider $\nabla_0 \vec{u} \approx \nabla \vec{u}$, so using that assumption one obviously have $\varepsilon_0 = \varepsilon$ and it really does not matter whether one uses one over the other.

A.2 The Assembly Process

A computational mesh consists of elements which each is defined by a set of nodes (or quadrature points). One way to store this information is as a two-dimensional matrix $\mathbf{T}$. Let the total number of elements be given by E and the total number of nodes of each element by C then $\mathbf{T} \in \mathbb{N}^{E \times C}$. Here row e stores the nodal indices of the e^{th} element. Observe that the column indices are local node indices for each row element whereas the row stores the global nodal indices.

We wish to assemble the global matrix $\mathbf{K} \in \mathbb{R}^{3V \times 3V}$ where V is the total number of nodes in the computational mesh. Initially we compute the E local matrices $\mathbf{K}^e \in \mathbb{R}^{3C \times 3C}$. These local matrices are conveniently stored in a linear array indexed by the element index e. Further, the local matrices can be computed in an embarrassingly naive parallel manner. In the following all matrices are considered as block matrices. All blocks are of dimension 3×3. Thus, indexing is done on blocks and not on entries.

To assemble the global $\mathbf{K}$ matrix we simply loop over the elements, for each element we lookup the corresponding local matrix and the global nodal indices before mapping these into the proper positions in the global $\mathbf{K}$ matrix. This can be written in pseudo code as follows

```
01 : **Algorithm matrix-assembly**$(\ldots)$
02 :   **for** $e = 1$ **to** $E$ **do**
03 :     **for** $i = 1$ **to** $C$ **do**
04 :       **for** $j = 1$ **to** $C$ **do**
05 :         $I \leftarrow \mathbf{T}_{e,i}$
06 :         $J \leftarrow \mathbf{T}_{e,j}$
07 :         $\mathbf{K}_{I,J} \leftarrow \mathbf{K}_{I,J} + \mathbf{K}^e_{i,j}$
08 :       **next** $j$
09 :     **next** $i$
10 :   **next** $e$
11 : **End algorithm**
```

Lines 05-06 are the local to global node indexing. Observe that Line 07 is the cause for a race condition in a parallel implementation. The pseudo code can easily be modified to assembly of vectors and we leave this as an exercise for the reader.

One may solve the race condition by rewriting the algorithm to use a gathering approach rather than the current scattering approach. The trick lies in rewriting the element loop into a node loop. For this to work we need an auxiliary data structure. For each node N we define the element incidence list $\mathcal{I}_N$ where each element in the list is a tuple of an element index e and the local index n of N as seen from element e. The resulting pseudo code is

```
01 : **Algorithm matrix-assembly-2**$(\ldots)$
02 :   **for** $N = 1$ **to** $V$ **do**
03 :     **for each** $(e, n) \in \mathcal{I}_N$ **do**
04 :       **for** $j = 1$ **to** $C$ **do**
05 :         $J \leftarrow \mathbf{T}_{e,j}$
06 :         $\mathbf{K}_{N,J} \leftarrow \mathbf{K}_{N,J} + \mathbf{K}^e_{n,j}$
07 :       **next** $j$
08 :     **next** $(e, n)$
09 :   **next** $N$
10 : **End algorithm**
```

The outer loop in line 02 is embarrassingly naive parallel. This algorithm version of the assembly process can be reworked into a matrix-free algorithm for computing the matrix-vector product of $\mathbf{K}$ and some given vector. The change needed is limited to line 06 and we leave it as an exercise for the reader.

A.3 Power Conjugacy

Here we will investigate work and power conjugacy. Conceptually we will think of the admissible test function used in the finite element analysis as a virtual velocity. Thus, $\vec{w} = \frac{\partial \vec{\omega}}{\partial t}$ where $\vec{\omega}$ could be thought of as the corresponding virtual displacement. This allows us to think physically about the weak form reformulation as virtual instantaneous power. It allows for an interpretation of the weak form as the principle of virtual power (D' Lamberts Principle q[Lanczos(1986)]). Our derivation in the following is not limited to virtual velocities. If true velocities or displacements were used the same derivation steps would hold. We keep the $\vec{w}$ notation to make the connection to the finite element method equations more apparent.

Now the elastic stress term is manipulated as follows

$$P_\sigma = \int_v \sigma : \nabla \vec{w} dv, \tag{A.12a}$$

$$= \int_v \sigma : \frac{1}{2} \left(\nabla \vec{w} + \vec{w}^T \right) dv, \tag{A.12b}$$

$$= \int_v \sigma : \mathbf{D}_w dv. \tag{A.12c}$$

In the first line we used the symmetry of the Cauchy stress tensor and in the second line we used the definition of the symmetric rate deformation tensor. The subscript is used to make it explicit that it is the virtual velocity we are working with and not the true spatial velocity. Thus, we have shown that σ and $\mathbf{D}$ are power conjugates. Next we will restate the elastic term using the Piola–Kirchhoff stress tensors. Recall that $\mathbf{F}_{\vec{\omega}} = \mathbf{I} + \nabla_0 \vec{\omega}$. Taking the time derivative leads to

$$\dot{\mathbf{F}}_{\vec{\omega}} = \nabla_0 \vec{w} \tag{A.13}$$

and by the chain rule we find that

$$\nabla_0 \vec{w} = \frac{\partial \vec{w}}{\partial \vec{x}} \frac{\partial \left(\vec{X} + \vec{\omega} \right)}{\partial \vec{X}} = \nabla \vec{w} \mathbf{F}_{\vec{\omega}}, \tag{A.14}$$

where we used $\vec{x}_\omega = \vec{X} + \vec{\omega}$. Thus, we find

$$\nabla \vec{w} = \dot{\mathbf{F}}_{\vec{\omega}} \mathbf{F}_{\vec{\omega}}^{-1}. \tag{A.15}$$

The spatial integral can now be rewritten as

$$\int_v \sigma : \nabla \vec{w} dv = \int_v \sigma : \dot{\mathbf{F}}_{\vec{\omega}} \mathbf{F}_{\vec{\omega}}^{-1} dv, \tag{A.16a}$$

$$= \int_V j \sigma \mathbf{F}_{\vec{\omega}}^{-T} : \dot{\mathbf{F}}_{\vec{\omega}} dV, \tag{A.16b}$$

$$= \int_V \mathbf{P} : \dot{\mathbf{F}}_{\vec{\omega}} dV. \tag{A.16c}$$

Leading to the conclusion that the power conjugate of the first Piola–Kirchhoff stress tensor is the time derivative of the virtual deformation gradient. By definition we can write the virtual Green strain as

$$\mathbf{E}_{\vec{\omega}} = \frac{1}{2} \left(\mathbf{F}_{\vec{\omega}}^T \mathbf{F}_{\vec{\omega}} - \mathbf{I} \right) \tag{A.17}$$

and taking the time derivative of the virtual Green strain tensor gives

$$\dot{\mathbf{E}}_{\vec{\omega}} = \frac{1}{2}\left(\dot{\mathbf{F}}_{\vec{\omega}}^T \mathbf{F}_{\vec{\omega}} + \mathbf{F}_{\vec{\omega}}^T \dot{\mathbf{F}}_{\vec{\omega}}\right). \tag{A.18}$$

Using the found relation $\nabla \vec{w} = \dot{\mathbf{F}}_{\vec{\omega}} \mathbf{F}_{\vec{\omega}}^{-1}$ and the definition of the virtual symmetric rate tensor we find

$$\mathbf{D}_w = \mathbf{F}_{\vec{\omega}}^{-T} \dot{\mathbf{E}}_{\vec{\omega}} \mathbf{F}_{\vec{\omega}}^{-1}. \tag{A.19}$$

Using this yields

$$\int_v \sigma : \mathbf{D}_w dv = \int_v \sigma : \mathbf{F}_{\vec{\omega}}^{-T} \dot{\mathbf{E}}_{\vec{\omega}} \mathbf{F}_{\vec{\omega}}^{-1} dv, \tag{A.20a}$$

$$= \int_V j \mathbf{F}_{\vec{\omega}}^{-1} \sigma \mathbf{F}_{\vec{\omega}}^{-T} : \dot{\mathbf{E}}_{\vec{\omega}} dV, \tag{A.20b}$$

$$= \int_V \mathbf{S} : \dot{\mathbf{E}}_{\vec{\omega}} dV. \tag{A.20c}$$

Showing that the second Piola–Kirchhoff stress tensor is power conjugate with the time derivative of the virtual Green strain tensor.

One can show work conjugacy terms using an almost identical approach assuming $\vec{w}$ to be a virtual displacement. This would lead to the relations

$$W_\sigma = \int_v \sigma : \varepsilon_{\vec{w}} dv = \int_V \mathbf{P} : \mathbf{F}_{\vec{w}} dV = \int_V \mathbf{S} : \mathbf{E}_{\vec{w}} dV. \tag{A.21}$$

A.4 Bonet and Wood for the Second Piola–Kirchhoff Stress Tensor

We will use the second Piola–Kirchhoff weak form for the elastic term see Appendix A.3 for details on deriving it. We wish to compute

$$D(\mathbf{S} : \dot{\mathbf{E}}_{\vec{\omega}})[\vec{u}] = D\mathbf{S}[\vec{u}] : \dot{\mathbf{E}}_{\vec{\omega}} + D\dot{\mathbf{E}}_{\vec{\omega}}[\vec{u}] : \mathbf{S}. \tag{A.22}$$

Recall that $\mathbf{S} = 2\frac{\partial \Psi}{\partial \mathbf{C}} = \frac{\partial \Psi}{\partial \mathbf{E}}$ so by the chain rule we have

$$D\mathbf{S}_{\mathbb{I},\mathbb{J}}[\vec{u}] = \sum_{\mathbb{K}=1}^{3} \sum_{\mathbb{M}=1}^{3} 4 \frac{\partial^2 \Psi}{\partial \mathbf{C}_{\mathbb{I},\mathbb{J}} \partial \mathbf{C}_{\mathbb{K},\mathbb{M}}} D\mathbf{E}_{\mathbb{K},\mathbb{M}}[\vec{u}]. \tag{A.23a}$$

We define the fourth order elasticity tensor $\mathcal{C}$ as

$$\mathcal{C}_{\mathbb{I},\mathbb{J},\mathbb{K},\mathbb{M}} = \mathcal{C}_{\mathbb{K},\mathbb{M},\mathbb{I},\mathbb{J}} \equiv 4 \frac{\partial^2 \Psi}{\partial \mathbf{C}_{\mathbb{I},\mathbb{J}} \partial \mathbf{C}_{\mathbb{K},\mathbb{M}}}. \tag{A.24}$$

Using our definition we write

$$D\mathbf{S}[\vec{u}] = \mathcal{C} : D\mathbf{E}[\vec{u}]. \tag{A.25}$$

Using this we find

$$D(\mathbf{S} : \dot{\mathbf{E}}_{\vec{\omega}})[\vec{u}] = \dot{\mathbf{E}}_{\vec{\omega}} : \mathcal{C} : D\mathbf{E}[\vec{u}] + D\dot{\mathbf{E}}_{\vec{\omega}}[\vec{u}] : \mathbf{S}. \tag{A.26}$$

We have

$$D\mathbf{F}[\vec{u}] = \nabla_0 \vec{u} = \nabla \vec{u}\mathbf{F}, \tag{A.27}$$

so using the product rule on the definition of the Green strain tensor leads to

$$\begin{aligned} D\mathbf{E}[\vec{u}] &= \frac{1}{2}\left((D\mathbf{F}[\vec{u}])^T \mathbf{F} + \mathbf{F}^T D\mathbf{F}[\vec{u}]\right), && \text{(A.28a)} \\ &= \frac{1}{2}\left(\mathbf{F}^T \nabla \vec{u}^T \mathbf{F} + \mathbf{F}^T \nabla \vec{u} \mathbf{F}\right), && \text{(A.28b)} \\ &= \mathbf{F}^T \left(\frac{1}{2}\left(\nabla \vec{u} + \nabla \vec{u}^T\right)\right) \mathbf{F}, && \text{(A.28c)} \\ &= \mathbf{F}^T \varepsilon \mathbf{F}. && \text{(A.28d)} \end{aligned}$$

It follows that $\dot{\mathbf{E}}_{\vec{\omega}} = D\mathbf{E}[\dot{\vec{\omega}}] = D\mathbf{E}[\vec{w}]$ so

$$D(\mathbf{S} : \dot{\mathbf{E}}_{\vec{\omega}})[\vec{u}] = D\mathbf{E}[\vec{w}] : \mathcal{C} : D\mathbf{E}[\vec{u}] + D\dot{\mathbf{E}}_{\vec{\omega}}[\vec{u}] : \mathbf{S}. \tag{A.29}$$

From the definition of the Green strain tensor we find the time derivative

$$\dot{\mathbf{E}}_{\vec{\omega}} = \frac{1}{2}\left(\dot{\mathbf{F}}_{\omega}^T \mathbf{F}_{\omega} + \mathbf{F}_{\omega}^T \dot{\mathbf{F}}_{\omega}\right). \tag{A.30}$$

The directional derivative is then

$$\begin{aligned} D\dot{\mathbf{E}}_{\vec{\omega}}[\vec{u}] = \frac{1}{2}\Big(\left(D\dot{\mathbf{F}}_{\omega}[\vec{u}]\right)^T \mathbf{F}_{\omega} + (D\mathbf{F}_{\omega}[\vec{u}])^T \dot{\mathbf{F}}_{\omega} \\ + \dot{\mathbf{F}}_{\omega}^T D\mathbf{F}_{\omega}[\vec{u}] + \mathbf{F}_{\omega}^T D\dot{\mathbf{F}}_{\omega}[\vec{u}]\Big). \end{aligned} \tag{A.31}$$

Since $D\mathbf{F}_{\vec{\omega}}[\vec{u}] = \nabla_0 \vec{u}$, $\dot{\mathbf{F}}_{\vec{\omega}} = \nabla_0 \vec{w}$ and $D\dot{\mathbf{F}}_{\vec{\omega}}[\vec{u}] = 0$ we have

$$D\dot{\mathbf{E}}_{\vec{\omega}}[\vec{u}] = \frac{1}{2}\left(\nabla_0 \vec{w}^T \nabla_0 \vec{u} + \nabla_0 \vec{u}^T \nabla_0 \vec{w}\right). \tag{A.32}$$

Substitution of this into our derivation and using the symmetry of $\mathbf{S}$ results in

$$D(\mathbf{S} : \dot{\mathbf{E}}_{\vec{\omega}})[\vec{u}] = D\mathbf{E}[\vec{w}] : \mathcal{C} : D\mathbf{E}[\vec{u}] + \mathbf{S} : \nabla_0 \vec{u}^T \nabla_0 \vec{w} \tag{A.33}$$

which becomes

$$D(\mathbf{S} : \dot{\mathbf{E}}_{\vec{\omega}})[\vec{u}] = \mathbf{F}^T \varepsilon_{\vec{w}} \mathbf{F} : \mathcal{C} : \mathbf{F}^T \varepsilon \mathbf{F} + \mathbf{S} : \nabla_0 \vec{u}^T \nabla_0 \vec{w}. \tag{A.34}$$

Now in principle all one need to do is to make the approximations and rewrite the equations into the form $\sum_{a,b} \vec{w}_a \mathbf{K}_{a,b} \vec{u}_b$ as where done in Section 4.4.