Modelling the Development of Osteoarthritis in the Human Knee Joint

Adam Reeve

Supervised by
Dr. Piaras Kelly and
Dr. Pranesh Kumar

September 2009

THE UNIVERSITY OF AUCKLAND
FACULTY OF ENGINEERING
Department of Engineering Science

Abstract

Osteoarthritis is a joint disease which mainly affects the elderly, but can also be a problem for much younger people. Osteoarthritis affects the whole joint, including not only the cartilage but the surrounding bone. Osteoarthritis may develop through multiple pathways, but is believed to often initiate through abnormalities in bone remodelling. The aim of this project was to develop a computer model of the process of osteoarthritis development, based on changes in bone remodelling.

A bone remodelling simulation was developed using the Abaqus finite element modelling software and Python. In this simulation a model of a joint is loaded multiple times and at each step the internal density distribution of the bone is updated based on the stress and strain experienced within the bone. At the surface of the bone, individual elements are also added and removed, allowing the bone to grow into empty space.

This simulation was able to illustrate that changes in the process of bone remodelling can lead to stiffening of the subchondral bone and bony growths known as osteophytes, symptoms which are often seen in osteoarthritis patients and can lead to cartilage damage.

With improved understanding of how osteoarthritis develops in different patients, it will be possible to more accurately target the underlying causes of the disease and develop improved treatments for patients.

Acknowledgements

Firstly, I would like to thank my supervisors, Dr. Piaras Kelly and Dr. Pranesh Kumar for their guidance, support and encouragement throughout this project.

Thank you very much to Michael Byrne for help and advice on the use of HyperMesh for meshing my three dimensional tibia model.

Thank you also to Thomas McKay and Ben O'Brien for their advice on Abaqus and the use of Fortran subroutines in Abaqus, as well as their thoughts on how to model bone remodelling.

Finally, I would like to thank Dr. Iain Anderson for advice on how to model bone loading and also sharing other ideas on this project.

Contents

List of Figures

1 Introduction

1.1 Project Aims

The aim of this project is to create a computer model to illustrate that changes in bone remodelling can lead to the development of symptoms involved in osteoarthritis. Due to genetic variation within the population, different people have different levels of stimulus that result in bone growth or resorption. People who have a low threshold for bone growth are more likely to develop stiffer bone much earlier in life, which may lead to osteoarthritis.

The stiffening of subchondral bone is believed to be a major factor in cartilage damage (Radin and Rose, 1986). Using a finite element model of the hip joint, stiffer subchondral bone has been shown to increase the stress in the overlying hyaline cartilage, making it more likely to be damaged (Wei et al., 2005). This project aims to add to the work already done on modelling osteoarthritis by others, by incorporating bone growth models to show how subchondral stiffening and osteophytes can develop in osteoarthritis.

As part of this project a trip was taken to Tauranga hospital to see osteoarthritis patients and obtain a better understanding of how osteoarthritis presents both clinically and radiographically. Different patients had developed osteoarthritis through different paths. Most were elderly and the disease had developed with age, however the youngest patient was a 45 year old male who had developed osteoarthritis in the hip after damage to the joint as a young boy. Being overweight was also an important factor for some patients. Some patients had received knee replacements to treat their osteoarthritis and reported a much greater quality of life after the surgery, with improved mobility and freedom from pain.

1.2 The Human Knee Joint

The knee is the largest and most complex joint in the human body and is a type of synovial joint. Synovial joints contain a synovial cavity between the articulating bones which is surrounded by an articular capsule. Synovial joints also have layers of hyaline cartilage on the articulating surfaces of bones, known as articular cartilage. Articular cartilage has an

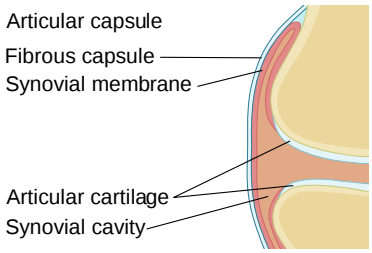


Figure 1.1. Diagram illustrating the features of a synovial joint.

extremely low coefficient of friction, and allows smooth movements of the joints in the body (Figure 1.1).

The knee joint actually consists of three joints within one synovial cavity (Figure 1.2). At the patellofemoral joint (not shown) the femur articulates with the patella. There are two tibiofemoral joints where the femoral condyles articulate with the tibia. These are modified hinge joints which allow flexion and extension of the knee in the sagittal plane and also permit some rotation.

A distinguishing feature of the knee is the incorporation of medial and lateral menisci. These are fibrocartilage pads which partly separate the joint cavity and disperse stress across the tibial surface. Like most joints, the knee also has a number of ligaments which are made of a fibrous tissue with high tensile strength, and help to prevent excessive movement that could lead to joint damage or dislocation.

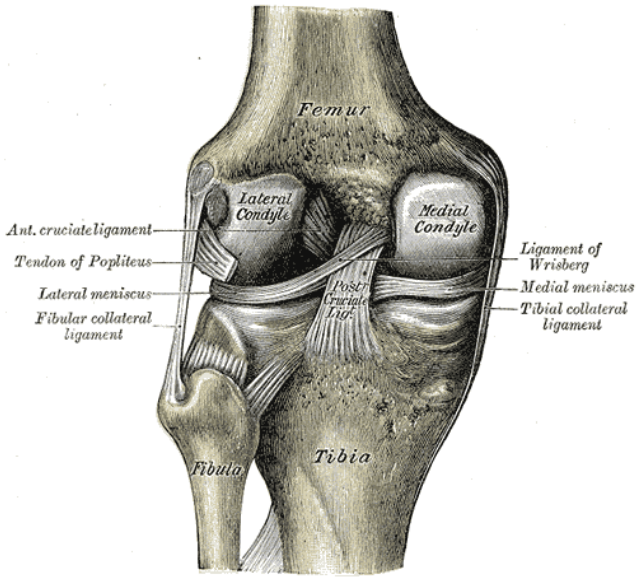


Figure 1.2. Posterior view of the left knee showing the femur, tibia, fibula, interior ligaments and menisci. Reproduced from Gray's Anatomy 1918, in the public domain.

At rest, the surfaces of the femur and tibia are separate. During walking and other load bearing activities the surfaces are compressed against each other to maximise the contact area and minimise stress transmitted through to the subchondral bone. Here the menisci and cartilage provide an important role in dispersing stress. The bone itself also has viscoelastic properties, and with larger applied loads deformation of the bone microstructure is important in absorbing stress.

1.3 Bone Remodelling

All bones in the body consist of two main types of bone, cortical and cancellous bone (Figure 1.3). Cortical bone is a very compact tissue with high stiffness and is found in the cortex, or outer layer of the bone. It is constructed from cylindrical units named osteons that contain a central haversian canal surrounded by concentric rings of extracellular matrix material. This matrix consists mostly of crystallised mineral salts, as well as collagen fibres and water. The most abundant mineral is calcium phosphate. There are small lacunae within the bone matrix which contain mature bone cells known as osteocytes. These osteocytes have finger-like projections within small channels called canaliculi that allow them to communicate with other cells in the bone matrix. The osteons are found to align with the directions of principal stress in the bone.

Cancellous bone is found within bones beneath the cortical bone, and has an irregular trabecular structure, illustrated in Figure 1.3. The space between the trabeculae is usually filled with bone marrow and the trabeculae are found to align with the directions of principal stress.

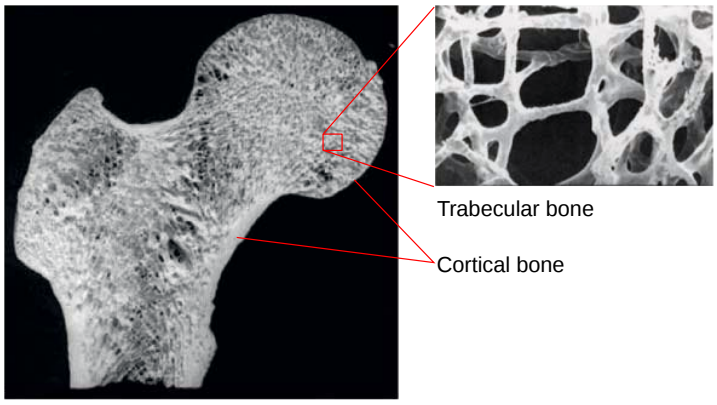


Figure 1.3. Cortical and cancellous bone regions in the femur, with an enlarged view showing the trabecular structure of the cancellous bone. Reproduced from Tsubota et al. (2009).

Bone undergoes a continuous process of remodelling throughout its lifetime. During this process, bone is resorbed by osteoclasts. Osteoblasts then deposit new bone material and as they build up new bone they become trapped within the bone matrix and become osteocytes. The net difference in the amount of bone resorbed and deposited may be positive or negative, resulting in an increase or decrease in the amount of bone present. Bone growth may be either interstitial or appositional. Interstitial growth refers to synthesis within the bone, and appositional growth is addition of bone at the surface (Guldberg et al., 2008).

Bone growth depends on the local strain field in the bone. This idea is often attributed to Julius Wolff, and known as Wolff's Law. It was originally based on observations that the trabecular bone structure is aligned with the principal stress directions (Guldberg et al., 2008). Bone cells involved in remodelling are able to sense strain, a process known as mechanotransduction. Forces transmitted from the bone surface cause mechanical deformation of cells and a change in interstitial fluid pressure. Dynamic fluid flows within the bone can also lead to shear stresses on the cell surface. These multiple mechanisms of force transduction can be sensed using stretch-activated ion channels in the plasma membrane.

Because of the complex interactions involved in mechanotransduction it is difficult to test exactly how cells sense transmitted stress and strain in-vivo, however various in-vitro experiments have been carried out to test intracellular responses to strain (Charras and Horton, 2002). It is believed that in general it is fluctuating fluid pressures that stimulate bone growth (Guldberg et al., 2008). Because of this, not only the loading magnitude but also the rate of loading is an important factor in stimulating bone growth. Experimental results have shown that an increase in loading frequency increases the rate of bone formation (Turner, 1998).

1.4 Osteoarthritis

Arthritis is a group of diseases that involve damage to the joints. Osteoarthritis is the most common form of arthritis, in which joint damage occurs due to mechanical degradation. Other forms of arthritis include rheumatoid arthritis, an autoimmune disease in which the immune system attacks the joint, and septic arthritis, an infection in the joint.

Osteoarthritis patients present with joint pain, stiffening of the joint and a decreased range of movement. There are four main radiographic signs of osteoarthritis. There is a decrease in joint space due to degradation of cartilage and thickening of the subchondral bone towards the joint space. Cartilage damage initiates with vertical tears at the interface with the underlying bone. There is stiffening of subchondral bone, seen as an increase in bone density on an x-ray image. Osteophytes are often visible and subchondral cysts can be present but are more rare. Osteophytes are small bone growths that form tangentially to the joint surface and can be several millimetres long. Subchondral cysts are fluid filled sacs

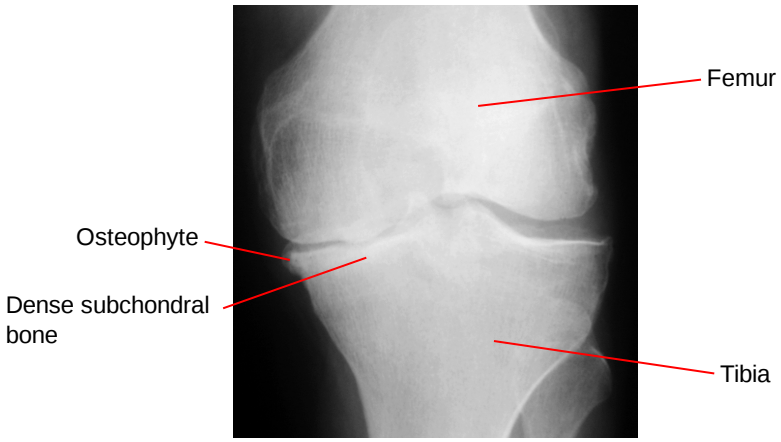


Figure 1.4. X-ray image of an osteoarthritic right knee from behind showing osteophyte formation and subchondral bone stiffening.

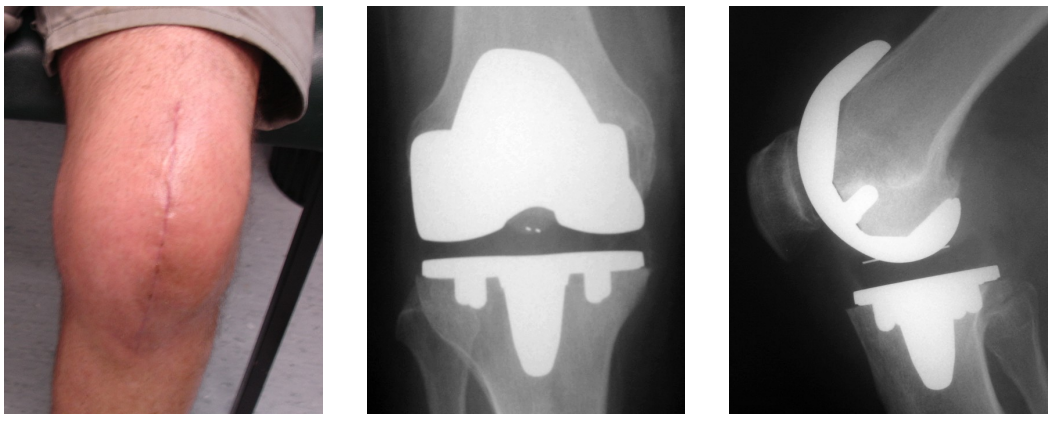
beneath the bone surface. When they are present there are usually only one or two, and they may range in size from a cubic millimetre to several cubic centimetres. Figure 1.4 shows an osteoarthritic knee with an osteophyte and subchondral stiffening.

Osteoarthritis has historically been viewed as a disease of the cartilage caused by breakdown of the cartilage with age. More recently there has been a shift in this view and an increased understanding that osteoarthritis involves the entire joint including the underlying bones (Karsdal et al., 2008). Osteoarthritis is a complex process with multiple pathways that lead to a similar result and multiple factors controlling its progression. Age is the most important factor in osteoarthritis development, and the probability of developing osteoarthritis increases significantly with age. Interestingly, many elderly people exhibit some radiographic symptoms of osteoarthritis such as stiffening of subchondral bone or osteophyte growth without exhibiting any clinical symptoms such as pain or a decreased range of movement. Part of the reason for the increased risk of osteoarthritis can be explained by changes in cartilage with age, which make it more susceptible to damage (Martin and Buckwalter, 2002).

Weight is an important factor in the development of osteoarthritis, as an increase in load carried by a joint increases the chance of joint damage. Studies of twins and siblings have shown that genetic factors play a large part in the development of osteoarthritis and affect the probability of developing osteoarthritis (Valdes and Spector, 2009). Biomechanical factors such as muscle strength and movement are also important in osteoarthritis development (Jackson et al., 2004). Osteoarthritis may also develop as a result of abnormal joint loading due to alignment problems or through trauma causing damage to the bone or cartilage.

The progression of osteoarthritis is still not thoroughly understood, and there are multiple theories for how symptoms develop. For example, subchondral cysts are often seen as "kissing lesions," where there are two cysts directly opposite each other across the joint space. These are widely believed to develop due to an influx of synovial fluid from the joint space into the subchondral bone, with a one-way-valve mechanism that prevents flow back into the joint space. Another theory exists however, that these cysts can be explained by stiffening of the subchondral bone. Increased bone growth causes an increase in interstitial pressure. In regions of bone with very high density, the interstitial pressure prevents blood flow to the region and leads to bone death, creating a cyst surrounded by a shell of high density bone. This explanation would also account for the observation that subchondral cysts often occur opposite each other, which is not explained by the theory that they develop through an influx of synovial fluid. Subchondral cysts often occur only in very late stages of osteoarthritis so it is likely that the stress levels required for this to take place require some degeneration of the overlying cartilage.

There is some evidence that with a change in loading, joints may be able to repair damage themselves. Damaged hyaline cartilage can be replaced by fibrocartilage which may eventually be able to develop into hyaline cartilage again (Radin and Burr, 1984). However, osteoarthritic knees usually require surgical treatment to remove pain and provide patients with a normal degree of movement. In some patients an osteotomy, or realignment of the bones may be an effective treatment, but usually an arthroplasty or joint replacement is required. Figure 1.5 shows a total knee replacement for treatment of osteoarthritis. This involves replacement of the femoral and tibial surfaces with metallic and plastic components. Unicompartmental knee replacements are another surgical treatment option, in which only the lateral or medial side of the knee is replaced.



(a) (b) (c)

Figure 1.5. Total knee replacement for treatment of osteoarthritis. (a) shows a knee from the outside after surgery, (b) shows an x-ray of the implant in the coronal plane, and (c) shows an x-ray in the sagittal plane.

2 Modelling Bone Growth and Osteoarthritis

2.1 The Finite Element Method

When modelling the mechanics of bones and joints there are a number of approaches that can be used. In many cases bones are assumed to be rigid bodies, due to the small amount they deform compared with soft tissues. In this project the stress and strain distribution within the bone are essential for determining how the bone remodels, so this approach is not practical. Instead the bone must be modelled as a deformable object. In nearly all applications where bones are modelled as deformable, the finite element method (FEM) is used.

The finite element method is a computational method for solving partial differential equations in complex geometric domains. It involves breaking up the problem domain into many smaller elements, with nodes defined at specific points on the elements. The solution can then be approximated over each element using low order polynomials. The finite element method is widely used for solving continuum mechanics problems in engineering, and has also been applied to biological problems including the mechanics of bones and joints.

2.2 Modelling Bone Remodelling with the FEM

There are many previous studies which have looked at the problem of modelling bone remodelling using the finite element method, and multiple approaches have been used. They fall into two main categories, apparent density remodelling and pixel or voxel based remodelling.

2.2.1 Apparent Density Remodelling

In the apparent density based approach to bone remodelling, a continuous density distribution is defined throughout the bone. The apparent density depends on the proportion of bone to empty space or bone-marrow within an element and can be thought of as the mass

of a small cube of cancellous bone, divided by the volume of the cube. This is different to the density of the cancellous bone itself, which is the mass of the bone divided by the area occupied by the bone only. The density may be defined as a relative density, where a value of zero means the element is completely empty space, and a value of one means the element is completely bone.

The bone remodelling process is modelled using a simple ordinary differential equation. The model is loaded multiple times and after each step the apparent density distribution is updated based on the stress and strain fields. The measure of stress or strain used to stimulate bone growth varies, commonly the strain energy per unit mass is used (Chen et al., 2007, Huiskes et al., 1992), or the average absolute value of the principal strains (Turner et al., 1997). An equilibrium strain level of the chosen strain measure is defined, and any strain above this value causes an increase in density. A stimulus below this level will cause a loss of bone, and a decrease in apparent density.

The apparent density remodelling approach using the strain energy per unit mass, given in differential equation form is given by Equation (2.1), where ρ is the apparent density, B is a constant controlling the rate of bone growth, SED is the strain energy density and k is the equilibrium level of strain energy per unit mass that will result in no net bone growth. The material parameters of the bone are defined to depend on the apparent density. A relationship between the Young's modulus and density can be obtained through mechanical testing of actual bones.

$$\frac{d\rho}{dt} = B\left(\frac{\text{SED}}{\rho} - k\right) \tag{2.1}$$

Generally a first order Euler's method is used to solve this equation, where the density at the next step is equal to the density at the last step, plus the right hand side of Equation (2.1) times the step size. Although the equation shows the rate of change of ρ with respect to time, there is no real time dependence in the simulation, as one loading step in the finite element simulation does not represent one load in real time, but rather an arbitrary number of similar loading cycles over an arbitrary amount of time. Therefore the loading frequency cannot be taken into account using this modelling approach. This also means that the step size used when solving the equation can be set to one, and the rate of bone growth is controlled by the parameter B instead. This parameter must be chosen to be small enough that there is no oscillatory behaviour in the solution.

In a simulation the apparent density distribution is assumed to be initially uniform , at approximately 20% of the density of solid trabecular bone. After multiple loading steps the density distribution adapts to the loading conditions applied, becoming denser in some regions and less dense in others (Figure 2.1). This method has shown to produce density distributions similar to the distributions seen in human bones (Jacobs et al., 1995).

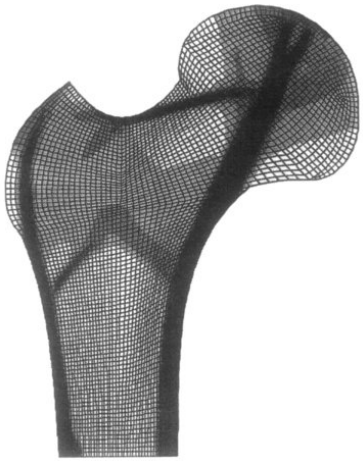


Figure 2.1. Density distribution in a femur after 500 remodelling steps using an apparent density remodelling method. Reproduced from Jacobs et al. (1995).

2.2.2 Pixel and Voxel Based Remodelling

The other approach to modelling bone remodelling is voxel based remodelling in three dimensions, or equivalently, pixel based remodelling in two dimensions. With this method all elements are either bone or empty, and no apparent density is used. Bone remodelling occurs by changing which elements are empty and which are solid bone.

Tsubota et al. (2002) developed a two dimensional pixel-based finite element model of bone remodelling in the proximal femur. Instead of having bone growth dependent on a function of stress or strain the authors instead assumed that bone growth was dependent on the nonuniformity of the stress and strain distribution. This supports the idea that fluid flows stimulate bone growth, as stress and strain gradients would cause fluid flow. The initial internal distribution of bone was generated by randomly filling the bone with a circular pattern, approximately the same size as cancellous bone trabeculae. The authors loaded their model multiple times, remodelling the bone at each step. They showed that the uniform strain criteria for bone remodelling accurately predicted the complex trabecular structure seen within the human femur (Figure 2.2).

Tsubota et al. (2009) extended the two-dimensional model into a three-dimensional voxel based model. The initial isotropic bone structure was generated by filling the bone with trabecular tori. This model was again able to accurately predict the trabecular structure of the femur and describe the orientation of bone along directions of principal stress.

Pixel and voxel based approaches are able to predict the trabecular structure of bone using the nonuniformity of strain as a stimulus for bone growth, however they require a very small element size. The two dimensional proximal femur model required 0.67 million

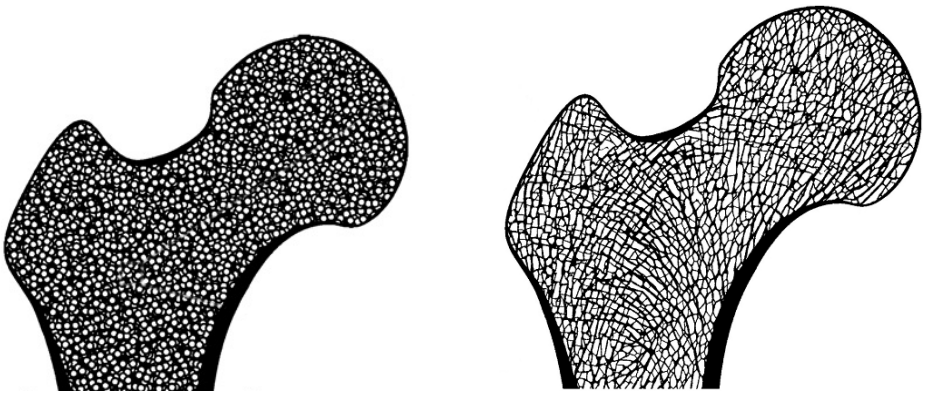


Figure 2.2. Change in microstructure within the proximal femur using a pixel based remodelling approach. (a) shows the initial distribution of bone in the femur and (b) shows the bone distribution after 16 loading steps. Reproduced from Tsubota et al. (2002).

elements and the three dimensional model had 93 million elements with a resolution of 87.5 μm. This approach is well suited for modelling the changes in bone at the trabecular level, but for large scale modelling of a whole bone or joint the bone microstructure is usually not important, and the apparent density information is all that is required to model the bone behaviour over time.

2.3 Modelling Osteoarthritis

Most research into bone remodelling is done in order to better understand the effect of bone prostheses, especially the process of prosthesis failure which is believed to often occur through bone remodelling (Turner et al., 2005). The metallic implants act as a stress shield on the surrounding bone, decreasing the stress experienced and decreasing the stimulus for bone growth. This causes bone resorption around the implant and an eventual loosening of the implant. A similar but opposite process is believed to occur in osteoarthritis, where increased bone growth causes the development of stiffened subchondral bone and osteophytes. Methods developed for modelling prosthesis failure are therefore also valid for modelling the development of osteoarthritis.

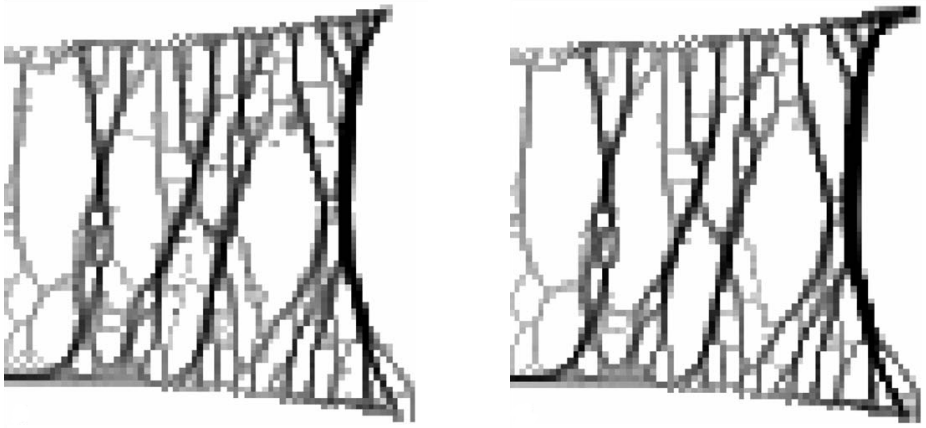
Studies of osteoarthritis using finite element models often look at the effect of different abnormalities of the joint on the joint environment (Wei et al., 2005, Shirazi and Shirazi-Adl, 2009), however there has been relatively little research done on modelling bone remodelling in osteoarthritis.

One study where the role of bone remodelling in osteoarthritis was looked at was carried out by He and Xinghua (2006). They modelled bone growth in a vertebra using a density

remodelling method, while also allowing the bone to expand into elements surrounding the bone. The strain energy density per unit mass was used as the stimulus for bone growth. The finite element model used consisted of a rectangular plate, larger than the vertebra. A section of elements within the plate were defined to be the initial vertebra shape, and 20 remodelling steps were carried out while only allowing the density within the vertebra to change.

After these steps, the loading conditions were changed to increase stress near the edge of the vertebral surface, and with subsequent loading steps the density in the empty elements adjacent to the bone was allowed to increase from zero. Using this method the authors were able to demonstrate the growth of an osteophyte at the edge of the vertebra (Figure 2.3). After each remodelling step the loading conditions were updated to ensure that the new bone elements that had been added to the edge received some load.

An important outcome of this study was that it demonstrated that osteophytes, a symptom commonly found in arthritis, could be explained by normal bone remodelling processes. In this case a change in the loading conditions initiated the development of the osteophyte.



(a) (b)

Figure 2.3. Growth of an osteophyte on a vertebra. (a) shows the initial density distribution within the vertebra, and (b) shows the distribution after remodelling with the growth of an osteophyte at the top right. Reproduced from He and Xinghua (2006).

3 Developing a Model in Abaqus

For this project the finite element software Abaqus was used.[1] Abaqus consists of the Abaqus/Standard and Abaqus/Explicit solvers for solving finite element problems, the Abaqus/CAE kernel which receives commands from the user and controls the solvers, and the Abaqus/CAE graphical user interface. Abaqus contains an interpreter for the Python programming language, and the user interface uses Python commands to communicate with the kernel. The user interface provides a 3D modelling environment and allows users to perform all steps of a finite element analysis from creating and meshing a geometry to analysing the output.

3.1 Interfacing with Abaqus

The Abaqus graphical user interface allows users to easily perform most normal engineering finite element analyses, however implementing a bone remodelling algorithm requires updating the material properties of a model during multiple loading steps. The tools in the user interface do not provide this functionality, so different methods of interfacing with Abaqus were investigated. Abaqus allows the use of user supplied Fortran subroutines to customise certain aspects of the application. A test "UMAT", or user material subroutine was written in Fortran with the idea that the stress and strain field from the previous loading step could be stored and then the UMAT subroutine could use this field to define the material properties for the next loading step. Incompatibilities between Abaqus and the Fortran compiler available meant that this approach wasn't successful.

Other possible methods investigated for interfacing with Abaqus include using a scripting language to write Abaqus input files and read output files, or using the built in Python interpreter for Abaqus. It was decided that using the Python interpreter would be the simplest way to interface with Abaqus and should provide all the functionality required. It is likely that this approach is more suitable than the use of Fortran subroutines investigated initially.

The first Python script written could open an Abaqus model database, run the simulation, read the output file then adjust which elements are specified as bone and which are empty

[1] Abaqus Version 6.8, SIMULIA, Providence, RI, USA.

to implement a pixel based bone remodelling method. The same Python script has been developed throughout this project and the final implementation is provided in Appendix A.

The python script consists of a single class definition, named *BoneGrowth*. In order to run a bone remodelling simulation the class is substantiated by passing a number of parameters including the name of the model database file, names of materials and other properties used by Abaqus, as well as parameters used in the bone remodelling equations. The class has a *runSimulation* method which can then be called to run the bone remodelling simulation.

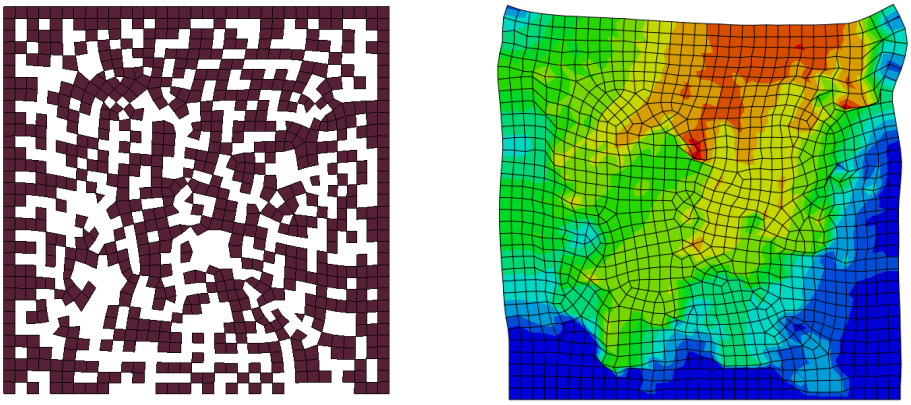
3.2 Pixel Based Remodelling

The aim of the project was to be able to show the growth of osteophytes in a joint, which would require adding and removing individual elements. Because of this a pixel based remodelling approach was initially implemented. Simple two dimensional blocks were constructed in Abaqus for testing the bone remodelling procedure. Two different materials were defined, a bone material with the stiffness of cortical bone (18.6 GPa), and an empty material with a relatively insignificant stiffness (0.01 MPa).

After each loading step, the Python script opens the output database created by Abaqus and uses the results to redefine the elements that make up the empty and bone sets. Initially the maximum of the absolute values of the principal stresses was used to determine which bone elements would be added and removed. If the strain in a bone element was below a certain threshold, that element was removed from the list of bone elements and added to the list of empty elements. If the strain in a bone element was above a certain threshold, any adjacent empty elements were removed from the list of empty elements and added to the bone elements.

In order to obtain a reasonable initial structure to begin the remodelling process, the interior of the bone was randomly filled with 25% bone elements, with the rest of the elements defined as empty. This is approximately the same proportion of bone seen in normal cancellous bone. Under loading, the random internal structure caused the bone to collapse, so the proportion of bone filled was increased to 50%. Even with a high proportion of bone in the initial structure it would collapse and deform significantly under load. This is because the structure generated by the random filling method is unstable and contains many large empty regions (Figure 3.1).

To avoid this problem, the method used by Tsubota et al. (2002) could have been implemented, where instead of randomly filling the elements in the bone interior, the bone is filled with a pattern of circular rings. Tsubota et al. used this method not only to provide a stable structure but also a structure which was close to the final trabeculae generated by the simulation. This procedure would be difficult to implement, and because a pixel based remodelling method requires higher resolution elements, it was decided that an apparent



(a) (b)

Figure 3.1. Bone block randomly filled with cancellous bone elements. (a) shows the bone elements, white areas are empty. The top, left and right sides have been specified as bone, and the internal has been randomly filled with 50% bone elements and 50% empty elements. (b) shows the deformed shape of the block under load, with displacement plotted. Blue through to red represents small to large displacements.

density based approach could be used inside the bone, and individual elements could be added and removed only at the bone surface to allow for the growth of osteophytes.

3.3 Apparent Density Remodelling

Abaqus allows mechanical properties, including the Young's modulus, to depend on temperature. This feature was used to allow the bone stiffness to change with the apparent density, which was specified in place of temperature data. In the Abaqus material property editor a table of values for the temperature and corresponding material properties can be entered. Abaqus linearly interpolates the values in the table to determine the material properties at a given temperature, or in this case an apparent density. The apparent density field was described using a discrete field, which is defined using a table of nodes or elements with the corresponding temperature at that position. This table can be accessed through the Python interface and updated after each loading step to change the apparent density throughout the bone.

Because bone cells sense strain through fluid movement within the extracellular matrix, the rate of loading and frequency of loading are important, however this cannot be incorporated in a static analysis. Static measurements of the stress and strain fields have been used in previous studies and should be able to provide a reasonable approximation. For this project initially, the use of the average absolute value of the principal strains was used, as in Turner et al. (1997). Using Equation (2.1) and the first order Euler method, the density

at each step was updated according to Equation (3.1), where ρ_{n+1} is the density at the next time step, ρ_n is the density at the current time step, B is a constant controlling the amount of bone growth at each step, E_1 and E_2 are the in-plane principal strains and k is the equilibrium strain level that will result in no growth.

$$\rho_{n+1} = \rho_n + B\left(\frac{|E_1| + |E_2|}{2} - k\right) \tag{3.1}$$

Initially the temperature field was defined at the element centroids. This made it simpler to update the density field as stress and strain invariants could easily be obtained through the Python interface at the centroid. This approach did not work however, as changing the value of the temperature field did not affect the stiffness. It was eventually determined that Abaqus requires the temperature field to be defined at the nodes, so this was implemented instead. Defining the bone apparent density at the nodes instead of the elements does have advantages. It has been shown that this helps prevent numerical instabilities and checkerboarding artifacts, which are discontinuities in apparent density between adjacent elements (Jacobs et al., 1995).

The remodelled density distribution within the knee produced after 30 remodelling steps is shown in Figure 3.2. The loading applied was a constant pressure through the cortical bone of the femur, with a symmetry boundary condition applied at the lower edge of the tibia. The cartilage was defined to be continuous across the joint, rather than modelling the contact between the two cartilage surfaces.

3.4 Working with the Python Interface for Abaqus

Most of the work in this project involved implementing bone remodelling methods in Abaqus. Both the algorithms used and the methods used to implement them in Abaqus changed as the project progressed. When a model is solved in Abaqus an output database is produced containing the solution data. After each loading step in the remodelling simulation this database is opened and read using the Python interface. The stress and strain fields are read from the database and the bone model is updated to represent bone changes due to remodelling.

When modelling apparent density changes in Abaqus the apparent density is defined using a temperature field, which must be defined at the model nodes. Because the stress and strain field are used to determine the apparent density, these fields are required at the nodes, however they can only be accessed at the element integration points or the element nodes using the Python interface. The element nodal values are simply extrapolated from the integration points, so at each node the field will be defined multiple times from the adjacent elements. In order to obtain field values at the global nodes these element nodal values

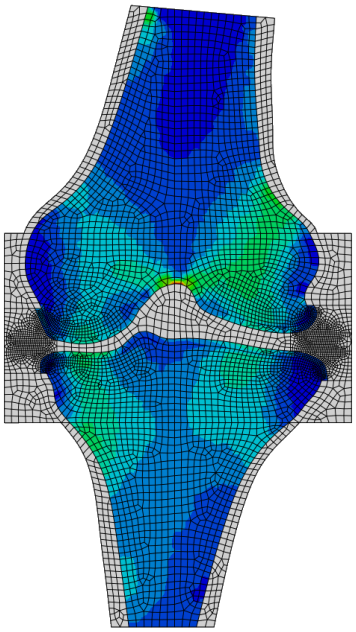


Figure 3.2. Density distribution within the knee in the coronal plane after 30 loading steps. The density ranges between 0.19 and 0.5 of the density of cancellous bone trabeculae. The grey regions are empty or cortical bone and the darker regions are where the mesh is dense to allow for surface remodelling.

were initially averaged, however this is a very slow process. Instead the final approach finds the first element nodal value which matches the required node. This makes the simulation run faster, but will lead to inaccuracies, especially with larger element sizes and steep stress and strain gradients.

When remodelling using the pixel or voxel based methods the strain in bone elements at the surface is examined, and if a measure of the strain is above some tolerance, the adjacent empty elements are converted to bone elements. This requires obtaining the labels of elements that are adjacent to the given element. Initially the adjacent elements were obtained by determining which other elements share more than one node with a given element. This information was generated then saved to a file. In subsequent simulations it could then be read in without having to be recalculated. An improved method was later developed by using interfaces built into Abaqus. Objects representing the edges of an element can be obtained, which in turn allow access to the other elements sharing this edge. This allowed the pixel base remodelling to be performed much faster.

3.5 Loading Conditions

One of the difficult aspects of this project was how to incorporate loading which would give a good approximation to the loading experienced by a human knee throughout its lifetime. For simplicity a statically loaded model was used.

The initial 2-dimensional knee joint model created consisted of a single finite element domain containing the distal femur and proximal tibia in the coronal plane (Figure 3.2). Separate materials were specified for the cartilage, cortical bone and cancellous bone regions and a load was applied through the cortex of the femoral shaft. This approach only allowed a single loading state to be modelled, as it was difficult to shift the position of the femur relative to the tibia to model different loading states.

To improve this, a second model was created in the sagittal plane consisting of only the tibia with a layer of cartilage on the surface (Figure 3.3). The sagittal plane was used as during knee flexion and extension, the femur moves over the tibia in this plane, so the loading changes from tension to compression in this plane. It is also observed that osteophytes grow more often in the plane the joint moves in.

There have been a number of studies into the load experienced by the knee joint. Taylor et al. (2004) investigated the load transmitted through the knee joint during walking and stair climbing using a musculoskeletal model and gait analysis. They found that during walking the maximum load through the knee is approximately 3.1 times a subject's body

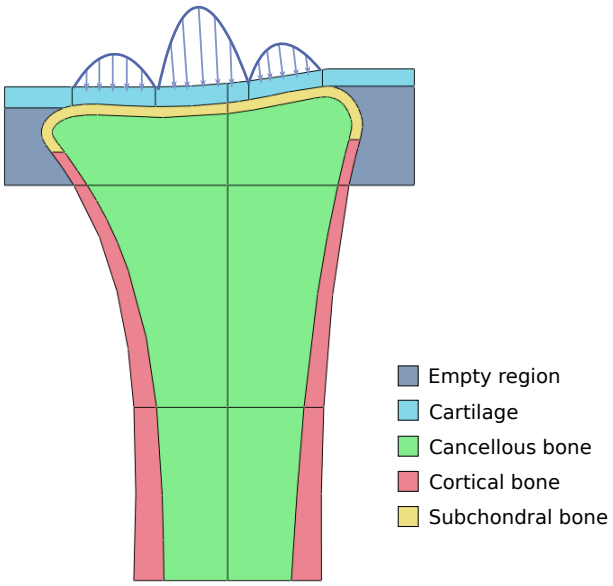


Figure 3.3. 2D model of the tibia in the sagittal plane, with different material sections labelled.

weight, although they did not gain any information on how this load is distributed across the joint surfaces. Ahmed and Burke (1983) used a pressure sensitive film and knee joints from cadavers to measure the pressure distribution on the tibia at different angles of flexion and Walker and Hajek (1972) investigated the contact areas between the tibia and femur at different angles of flexion using castings of the joint cavity in cadaveric joints. Zhao et al. (2007) investigated the difference in loading between the medial and lateral compartments of the tibia. They measured the loads from a single patient using a tibial prosthesis instrumented with uniaxial force transducers and found the split between medial and lateral load was approximately 55% – 45% respectively. For a more accurate measurement multiple patients would be required as this loading is likely to vary significantly in different patients due to differences in anatomy.

In order to try to represent the loading experienced by the tibia, a load of 3.1 times a body weight of 750 N was assumed, split 55% – 45% between the medial and lateral compartments. The load was all applied as a normal pressure, as the low coefficient of friction of the articular cartilage means there is very minimal shear stress on the joint surface. The two dimensional model used in the sagittal plane was defined to be through the medial compartment of the knee. The loading profile on the tibia was assumed to be parabolic, and three loading profiles were applied along the sagittal plane, each weighted by a constant (Figure 3.3). The centre load was weighted by 0.5, and the two outer loads were both weighted by 0.25. This loading approach is similar to that used by Tsubota et al. (2002) and Carter (1987).

3.6 Final Remodelling Approach

The first models which implemented density remodelling within the bone interior and pixel based remodelling at the surface were loosely based on geometry obtained from a computed tomography (CT) scan of a male human knee. These models consisted of cancellous bone which is remodelled using the apparent density approach, and cortical bone which is remodelled using the pixel based approach. The cortical bone had a constant stiffness throughout. This model was able to show density changes within the cancellous bone well, but the subchondral region was defined with a constant density so the model could not show stiffening of the subchondral bone.

The final bone remodelling method defined a fourth material in addition to the empty, cancellous and cortical materials (Figure 3.3). Approximately the top four millimetres below the bone surface was defined as subchondral bone. In this region the apparent bone density was updated at each step and the surface of the bone was also allowed to grow into the empty region. The subchondral bone region is not a different type of bone from cancellous or cortical bone but rather may vary between cortical and cancellous bone. The apparent density specified for any new bone elements added was the same as the initial

constant density applied before remodelling. The cortical bone region was assumed to be relatively inactive, so no change in the bone density was modelled here. Within the cancellous bone region the apparent bone density was updated at each step but no pixel based remodelling was used.

A region of cartilage was also defined above the subchondral bone, which also extended over the empty region. This was done because in osteoarthritis patients, osteophytes usually have a covering of cartilage. When there is empty space below the cartilage it will not have much effect, however if an osteophyte grows, the cartilage will help transfer some load through this new bone.

Instead of using the average absolute value of the principal strains as a stimulus for bone growth, the strain energy per unit mass was used instead as this more commonly accepted as a measure of the growth stimulus received by bone (Huiskes et al., 1992, Chen et al., 2007). The equilibrium value for the strain energy per unit mass within the subchondral region is able to be specified separately from within the cancellous bone region. This allows setting a lower equilibrium strain energy density in the subchondral region, so that stiffer cortical like bone will form in this region.

The cancellous and subchondral bone were assumed to be linearly elastic. The relationship between apparent density and stiffness was approximated by a quadratic function, which was determined experimentally by Rho et al. (1993). Equation (3.2) was used to relate the stiffness (in MPa) of the cancellous bone to the apparent density (in Kg m^{-3}).

$$E = -160 + 4\rho + 1.108 \times 10^{-3}\rho^2 \tag{3.2}$$

Within the subchondral bone region, the bone stiffness may be as high as that of the cortical bone, and the stiffness of cortical bone cannot be extrapolated from the stiffness of cancellous bone (Rho et al., 1993). In the subchondral bone region the equation was therefore shifted upwards by a constant value to give a maximum possible stiffness corresponding to that of cortical bone, at 18.6 GPa, compared to the 10.4 GPa stiffness of an individual trabecula in cancellous bone.

3.7 Cartilage Damage

In addition to bone changes, the damage of cartilage in osteoarthritis was also modelled. In a previous study by Peña et al. (2008), a finite element model of cartilage was developed and cartilage damage was modelled by decreasing the cartilage stiffness. The level of damage at each step was determined from the change in maximum shear stress. Other studies have suggested that the value of the maximum shear stress contributes to cartilage damage

(Wilson et al., 2006). Akizuki et al. (1986) showed that the stiffness of cartilage could decrease to as low as 2 MPa in patients with Osteoarthritis.

In osteoarthritis, cartilage damage is often seen as vertical splits where the cartilage joins with the underlying bone, and ultimately the cartilage can be completely worn away from the bone surface. This type of damage would depend on changing between a state of compression and tension in the cartilage as the bones of the joint articulate. In this project only a static analysis was used, so the maximum shear stress was used as the driving force for cartilage damage, and a very simple linear relationship was used between the cartilage stiffness and damage level, as in Peña et al. (2008).

The material properties for cartilage and cortical bone were taken from Wei et al. (2005), and the material properties for all sections within the bone are summarised in Table 3.1.

Table 3.1. Material parameters used in the simulation. All materials are assumed to be linearly elastic.

Region	Poisson's ratio	Young's Modulus (MPa)
Cartilage	0.47	$(0.1-15) \times \text{damage} + 15$
Empty	0.0	0.01
Cancellous bone	0.3	Equation (3.2)
Cortical bone	0.28	18600
Subchondral bone	0.3	Equation (3.2) + 8200

4 Osteoarthritis Simulation Results

To apply the bone remodelling algorithm to modelling osteoarthritis, two separate models were considered with different remodelling parameters. Two dimensional models of the tibia in the sagittal plane were used, with weighted parabolic loading profiles as described in Section 3.5. Plane strain conditions were assumed due to the large dimension of the bones out of the model plane, and all materials were assumed to be linearly elastic. The same initial state was used for both models, and was generated by remodelling the density of the bone over 20 steps without allowing the bone surface to change or cartilage damage to occur. The initial constant density distribution used was 25% of the maximum density and the equilibrium strain energy per unit mass was specified to be 5.0×10^{-5} Jm^{-3}. The apparent density is normalised so this has the same units as strain energy density.

Two simulations were then run using the initial desnsity distribution established, however different equilibrium strain levels and voxel remodelling levels were specified for the two models. The equilibrium strain energy per unit mass used was 5.0×10^{-5} Jm^{-3} for model A and 2.0×10^{-5} Jm^{-3} for model B. The strain energy per unit mass that would result in bone resorption in the subchondral bone was 1.0×10^{-8} Jm^{-3} for both models. For adding new bone elements a value of 3.0×10^{-5} Jm^{-3} was required for model A and 1.0×10^{-5} Jm^{-3} for model B. These values were chosen based on the strain energy levels observed in test simulations. The values for model A were chosen to represent a normal bone and the values for model B were chosen to represent an osteoarthritic bone that would grow osteophytes and show subchondral bone stiffening. A value of 800 was used for B, which controls the rate of bone growth.

At each step the apparent density was recorded at four nodes, which are shown in Figure 4.1. These nodes were chosen to represent a range of different positions. Two were directly beneath a loading point and one was between loading profiles, while another was deeper below the subchondral bone.

The change in density over the first twenty remodelling steps at these nodes is given in Figure 4.2. The first twenty steps were the same for both models and were used to establish an initial density distribution. Figure 4.2 shows that after these steps the density distribution had approached an equilibrium state, with little change in the density between steps. Two of the nodes show an initial increase in density followed by a gradual decrease. This is most likely due to interaction between the stiffness at different regions. As the

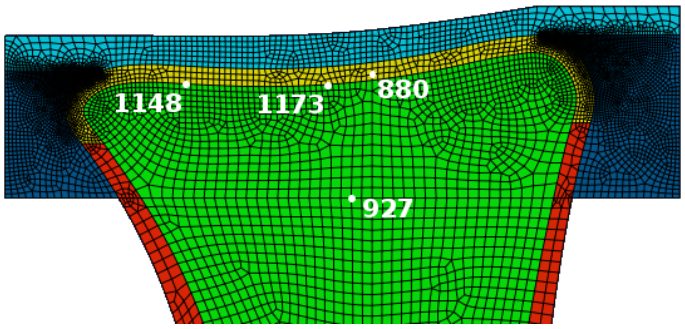


Figure 4.1. Position of nodes where the apparent density was recorded at each remodelling step.

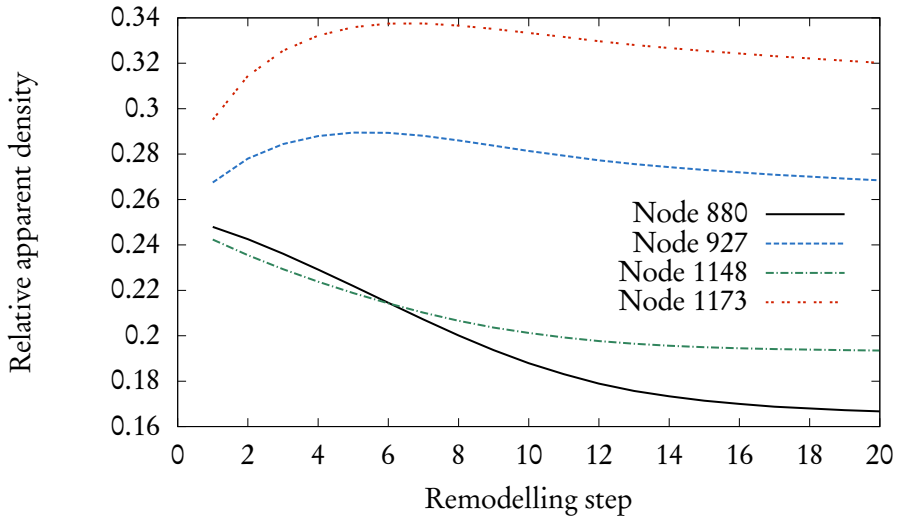
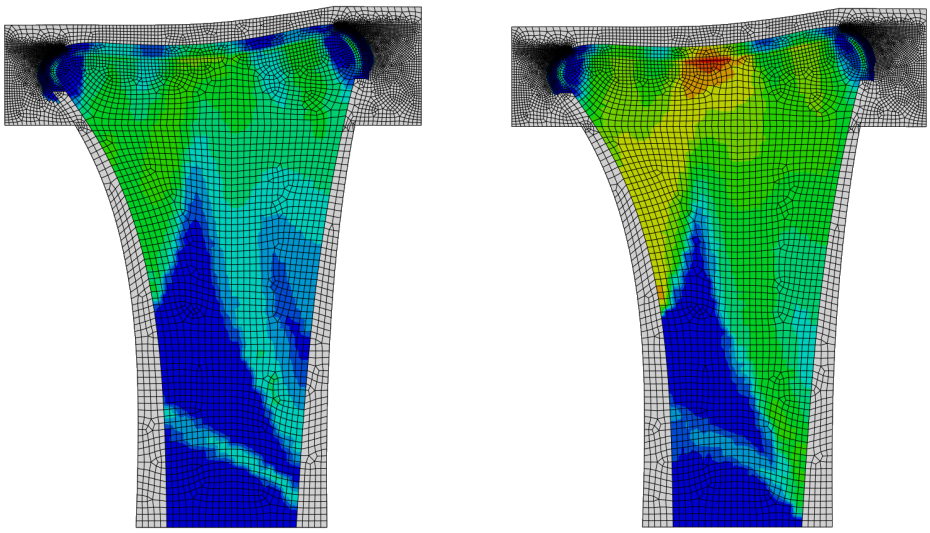


Figure 4.2. Change in density over time in the first 20 remodelling steps.

stiffness in one region is increasing, the stiffness in nearby regions will also be increasing or decreasing at different rates. As these other regions change stiffness this will affect the strain energy at the recorded region, and alter the density required to establish the equilibrium strain energy level.

4.1 Subchondral Stiffening

Figure 4.3 shows the remodelled density distributions for models A and B. The cancellous bone in model B is denser throughout the bone, especially in the subchondral bone beneath the central loading region. There is a discontinuity in the density distribution between the cancellous and subchondral bone, due to the change in material properties in these sections.



(a) (b)

Figure 4.3. Apparent density distributions for (a) model A and (b) model B after remodelling. The colour scale for each model is the same and ranges from a relative density of 0.0 (blue) to 0.65 (red).

Because both regions have the same equilibrium strain level, and the subchondral bone is stiffer at the same density, the subchondral bone will be less dense at the same position.

Figure 4.4 plots the density at node 1173 (Figure 4.1) at each remodelling step. Both models start with the initial density shown in Figure 4.2, however model B has a decreased equilibrium strain energy level. This causes an increase in bone growth, seen as an increase in apparent density, to try to reach the equilibrium strain level. Model A changes very little as the equilibrium strain level has not changed and the density distribution is very close to an equilibrium state. This plot clearly shows that the change in remodelling parameters for model B results in a large increase in apparent density of the subchondral bone. This in turn leads to an increase in bone stiffness in accordance with Equation (3.2).

Because cartilage has a high Poisson's ratio, it was expected that there may be larger tensile stresses directly beneath the cartilage, which would increase the strain energy density below the cartilage and lead to stiffer bone. The density plots do show an increase in the subchondral bone density but because of the change in material properties in this region it is difficult to compare the density with that of the underlying cancellous bone. There does not appear to be a significantly greater density directly below the bone surface however (Figure 4.5), so the effect of the large Poisson's ratio of the cartilage on the underlying bone is not great.

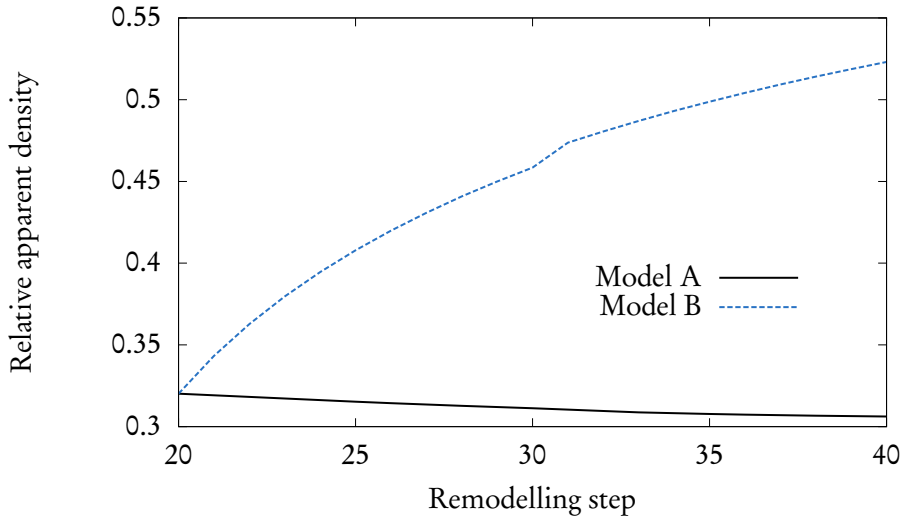


Figure 4.4. Change in density over time in model A and B at node 1173.

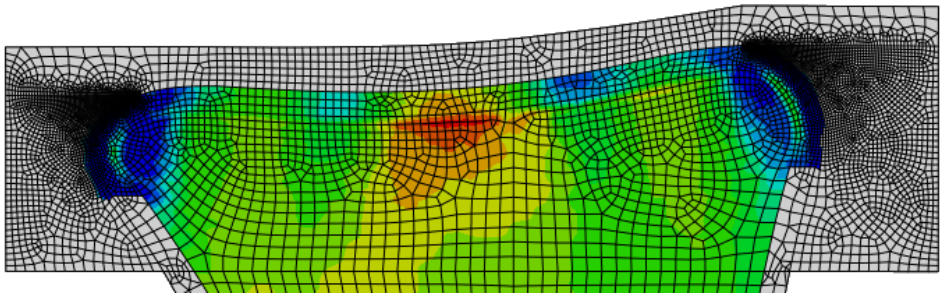


Figure 4.5. Bone density in the subchondral region of model B

4.2 Osteophyte Growth

Figure 4.6 shows the changes at the subchondral bone surface where osteophytes have developed in models A and B by adding new elements. The length of an osteophyte was defined as the horizontal length of all elements that were added to the initial geometry. In this figure it can be seen that the new bone grown has a greater density than the original bone. This is a limitation of the bone remodelling algorithm, in that all new bone elements added are given the same apparent density as the initial density of 0.25, and the original bone has had more time to remodel and decrease in density. An improved model may use the density of the surrounding existing bone to calculate the density of new bone elements. Table 4.1 gives the measured osteophyte lengths, which shows that model B has developed significantly larger osteophytes than model A. The increased osteophyte size is due to the decrease in strain energy per unit mass required for new bone growth. The models also

show a change in the surface of the bone below the top edge, with new elements added in this region also.

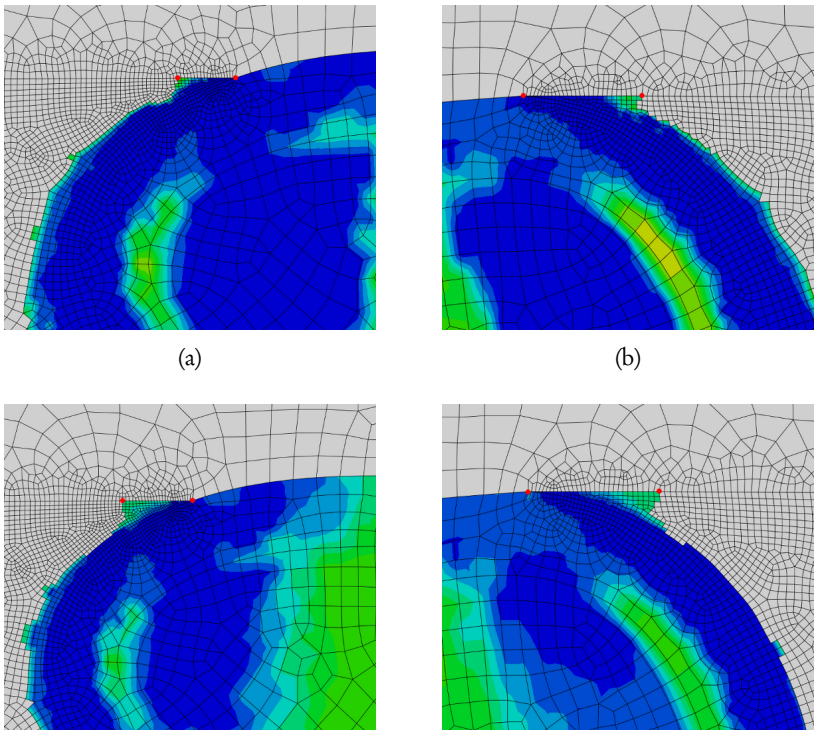


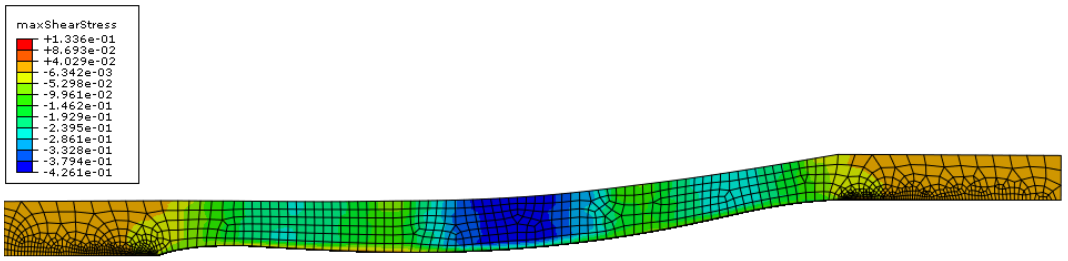
Figure 4.6. Osteophyte growth on models after twenty surface remodelling steps. The apparent density has been plotted within the bone and ranges from blue to yellow. Grey regions are empty. Red dots mark the positions where osteophyte lengths were measured. (a) shows the top left (anterior) of model A, (b) shows the top right (posterior) of model A, (c) shows the top left of model B, and (d) shows the top right of model B.

Table 4.1. Lengths of osteophytes measured from models A and B. The measured lengths are indicated in Figure 4.6.

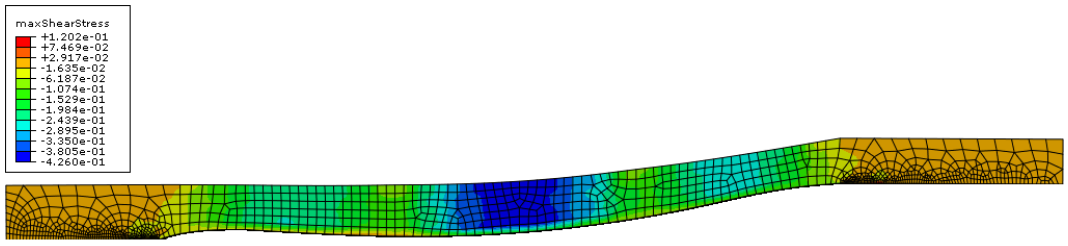
	Location	
	Posterior	Anterior
Model A	1.77 mm	3.56 mm
Model B	2.46 mm	3.89 mm

4.3 Cartilage Degradation

Figure 4.7 shows the maximum shear stress in the cartilage for models A and B in the final loading step. The stress distributions are relatively similar and the maximum shear stress over all of the cartilage is approximately the same, at 0.426 in the region where the central load is applied. The maximum shear strain and the Tresca stress were measured at four elements along the bottom of the cartilage and it was found that both measures decreased by approximately 1% at all four elements. The Tresca stress was used as this was the stress invariant which was found to increase with increasing subchondral bone stiffness by Wei et al. (2005). The increase in density of the subchondral bone in model B did not significantly change the stress distribution in the overlying cartilage in this model. Because there was not an increase in the maximum shear stress, the cartilage damage modelling method described in Section 3.7 did not come into effect and the cartilage remained at full strength.



(a)



(b)

Figure 4.7. Maximum shear stress in the cartilage in model A and B measured after 40 remodelling steps. The maximum shear stresses range from -4.26×10^{-1} to 1.34×10^{-1}.

4.4 Mesh Refinement

In addition to the two models A and B, a third model (model A_R) was also considered. This used a finer mesh size to ensure the remodelling method used was sufficiently accurate. This model used all the same parameters as model A, but had 19543 nodes compared to 9990 for models A and B. The meshes used for all models were finer near the top edges of the bone where individual elements would be added and removed.

Figure 4.8 shows the change in density over the first twenty remodelling steps used to establish an initial density distribution, as well as the following twenty steps for models A and the refined model, A_R. Because the models have different meshes applied, the node numbers used are different. The apparent densities have been recorded at the same position where possible, or at approximately the same position. Some of the difference between the two models can be attributed to the slightly different position of the nodes used, and overall there is relatively little difference between the densities. This indicates that the mesh used for models A and B is sufficiently refined to obtain a reasonably accurate result for the internal density distribution.

The osteophyte lengths for the refined model compared with model A are given in Table 4.2. The osteophyte lengths were significantly less with the refined model, showing that the element size affects the size of osteophytes grown. In model A_R the mean length of elements in the osteophyte region was 0.06 mm, compared with 0.13 mm for model A.

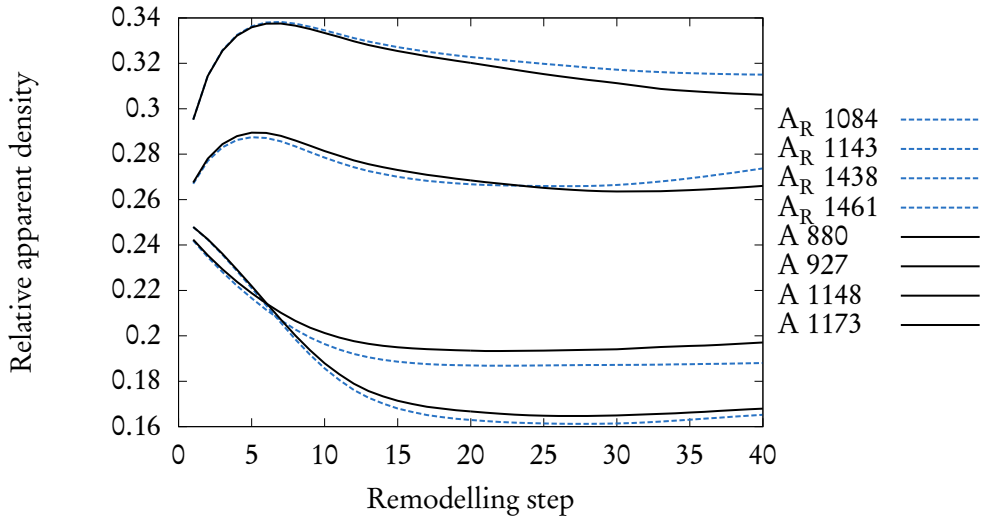


Figure 4.8. Change in density in the refined model (model A_R), compared with model A. Nodes used for the refined model were in the same position or approximately the same position as for model A.

Table 4.2. Lengths of osteophytes measured from models A and A_R.

	Location	
	Posterior	Anterior
Model A	1.77 mm	3.56 mm
Model A_R	1.19 mm	2.67 mm

The most obvious way in which element size affects the osteophyte length, is that at each remodelling step only a single layer of elements can be added, as new bone elements are added adjacent to existing bone. This means that for the same number of remodelling steps, the refined model will be able to grow less bone. In the final remodelling step of all three models, some bone elements were still being added and removed, so the model had not reached an equilibrium state.

5 Discussion and Future Directions

5.1 Conclusion

A bone remodelling algorithm was developed using the Abaqus finite element modelling software. Both pixel and apparent density based remodelling methods were investigated and the final remodelling method uses a combination of the two. The apparent density distribution within the bone is updated and bone elements are added and removed at the bone surface. A model of the proximal tibia was used, with weighted parabolic loading profiles on the tibial surface.

By varying bone remodelling parameters the model was able to demonstrate the development of osteoarthritis, and show that changes in bone remodelling can lead to changes in bone seen in osteoarthritis patients. The model showed both stiffening of the subchondral bone and the growth of osteophytes.

Decreasing the equilibrium strain energy parameter and the surface strain parameters in the simulation increased bone density and growth of osteophytes was observed. These changes in the model parameters correspond to changes in bone remodelling in-vivo, and represent an increased rate of bone growth. This causes a net increase in bone at lower strain levels. A difference in bone remodelling is most likely due to genetic variation between different people. This simulation therefore supports the theory that differences in bone remodelling between individuals are important in determining which people will develop osteoarthritis, and that genetic factors are very likely to be involved.

The model provides increased understanding of how osteoarthritis develops. With further knowledge on the process of osteoarthritis development and identification of genetic factors involved it may be possible to determine which patients are more likely to develop the disease. People that are at high risk of developing osteoarthritis can then be encouraged to decrease loading on their joints, for example by swimming or cycling instead of running for exercise. Patients that are more likely to develop osteoporosis on the other hand, a disease in which bone density is decreased, could be encouraged to load their joints with higher stresses to promote bone growth.

5.2 Discussion

The results obtained do not show an increase in maximum shear stress in the cartilage with an increased stiffness of the subchondral bone. Finite element simulations using simple blocks with layers of different materials showed that the position of the cartilage relative to the loading is important in whether cartilage will receive an increased or decreased stress. In cartilage directly below the loading position the von Mises and maximum shear stress decreased with increased stiffness of the underlying bone. In cartilage outside the region where loading is applied the stress increased with increased stiffness of the bone. Multiple parabolic loads from the femoral condyles are applied in the tibia model, so all of the cartilage is under compression. In reality the majority of cartilage will not be directly loaded at any one point in time, so it is likely that the reason an increase in cartilage stress was not seen is because all the cartilage was loaded.

In osteoarthritis there are also strong relationships between bone and the cartilage. Damage to cartilage further increases the stress in the underlying bone and accelerates changes in the subchondral bone, which will in turn further damage the overlying cartilage.

The results obtained with the refined model showed that a decrease in element size affected the resulting size of osteophytes grown. Decreased element size means that less bone can be grown in the same number of steps, so in order to make comparisons between models the same sized mesh must be used. This was the case for the two models investigated, so the change in osteophyte length grown is only due to the differences in bone remodelling parameters.

5.3 Future Directions

In this project only static loading conditions were used and parabolic loading profiles were assumed. To improve the simulation, loading conditions could be applied by modelling both the femur and tibia at different angles of flexion and using a contact mechanics analysis between the two articular surfaces. The action of the femoral condyles sliding over the tibia is also likely to be important, as this means the cartilage is changing between a state of compression and tension, which is likely to be important in the development of vertical splits in the cartilage. A temporal analysis which takes this movement into account could be used to more accurately model cartilage damage.

The simulation also does not take into account the loading frequency, which has shown to be important in bone remodelling. This could possibly be implemented in future models by weighting loads by a parameter related to the loading frequency.

Future efforts in modelling osteoarthritis may be able to incorporate temporal information

and correlate actual data from osteoarthritis patients with customised bone growth models. This would make it possible to estimate how the disease will progress over time in different patients. Improved understanding of how bone remodels in response to stress and strain will also increase the accuracy of bone remodelling simulations and make them more suitable for predicting bone remodelling in vivo.

With an increase in knowledge of the genetic factors involved, genetic information from individual patients may also be incorporated into bone remodelling simulations. This may include information on how chemical messengers are involved in communication between cartilage and subchondral bone, which is believed to be important for osteoarthritis development (Karsdal et al., 2008). In future, clinicians could use patient specific bone models that incorporate bone remodelling to determine how individual cases are likely to develop. This information could be used for planning treatment or determining if surgery is required.

A Bone Remodelling Python Script

```
"""
Load a bone model multiple times, simulating bone growith in cancellous
bone by changing the bone density, and in subchondral bone by changing
the density and adding and removing bone elements.

Author: Adam Reeve. CreationDate: 22/04/09
"""

from abaqus import *          # access session and mdb objects
from odbAccess import *       # get access to results
from abaqusConstants import *
import visualization
from operator import add

class BoneGrowth:
    def __init__(self,modelName,partName,instanceName,odbPath,jobName,stepName,
            corticalBoneSectName,cancellousBoneSectName,emptySectName,
                subchondralSectName,
            cartilageSectName,equilibriumStrain,subchondralEquilibriumStrain,
            surfaceStrainLevels,B,cartilageDamageStress,recordDensityNodes):
        """
        Initialise BoneGrowth object with model parameters
        """
        self.model = mdb.models[modelName]
        self.part = self.model.parts[partName]
        self.instanceName = instanceName
        self.odbPath = odbPath
        self.job = mdb.jobs[jobName]
        self.stepName = stepName
        self.corticalBoneSectName = corticalBoneSectName
        self.cancellousBoneSectName = cancellousBoneSectName
        self.emptySectName = emptySectName
        self.subchondralSectName = subchondralSectName
        self.cartilageSectName = cartilageSectName
        self.equilibriumStrain = equilibriumStrain
        self.subchondralEquilibriumStrain = subchondralEquilibriumStrain
        self.surfaceStrainLevels = surfaceStrainLevels
        self.B = B
        self.cartilageDamageStress = cartilageDamageStress
        self.discreteTempFieldName = "tempDF"
        self.cartilageFieldName = "cartilageDF"
        self.newBoneDensity = 0.25
        self.recordDensityNodes = recordDensityNodes
        self.outputFileName = self.job.name+"_density_record.txt"
    def __del__(self):
        pass

    def runSimulation(self,steps,delaySurface=0,delayCartilage=0,done=0):
        """
        Run bone remodelling simulation.
        Parameters:
            steps: Number of loading and remodelling steps
            delaySurface: Number of cancellous remodelling only steps before
```

```
                allowing remodelling of cortical bone surface
            delayCartilage: Steps before modelling cartilage damage
            done: Number of remodelling steps already performed previously.
                  Only used to change naming of saved output database
        Return value:
            Boolean specifying success or failure
            Saves output database as the job name with the loading iteration number
        """
        from shutil import copy
        # open file to record nodal densities to
        if len(self.recordDensityNodes) > 0:
            outfile = open(self.outputFileName,'w')
            for n in self.recordDensityNodes:
                outfile.write("%d " % n)
            outfile.write('\n')
            outfile.close()
        for i in range(steps):
            print "Loading step", i
            result = self.__loadModel()
            if not result:
                print "Error running job"
                return False
            else:
                output = session.openOdb(name="boneOutput",\
                    path=self.odbPath,readOnly=False)
                self.__cancellousRemodel(output, i >= delaySurface)
                fail = False
                if i >= delaySurface:
                    try:
                        self.__surfaceRemodel(output,self.subchondralSectName)
                    except RuntimeError:
                        fail = True
                if i>= delayCartilage:
                    self.__cartilageDamage(output)
                output.save(); output.close()
                # copy output file to new odb
                newOdbName = self.job.name+"_"+str(i+done)+".odb"
                copy(self.odbPath,newOdbName)
                print "\n\n"
                if fail:
                    return False
                else:
                    mdb.save()

        return True

    def __loadModel(self):
        """
        Run the job and wait for completion.
        Return value:
            Boolean specifying success or failure running job
        """
        import os
        import os.path
        if True in [key.find(os.path.basename(self.odbPath)) > 0 \
                for key in session.odbs.keys()]:
            session.odbs[self.odbPath].close()
        if os.path.exists(self.odbPath):
            os.remove(self.odbPath)
        self.job.submit()
        self.job.waitForCompletion()
        if (self.job.status != COMPLETED):
            return False
        else:
            return True
```

```
    def __cancellousRemodel(self,output,surfaceRemodelling):
        """
        Remodel cancellous and subchondral bone by defining a "temperature" field
        and using a temperature dependence on the cancellous bone material
        Parameters:
            output: Abaqus output file
        Return value:
            None
        """
        print "Apparent density remodelling"
        tempField = self.model.discreteFields[self.discreteTempFieldName]

        domainNodes = tempField.data[0].domain[:]
        densities = list(tempField.data[0].table[:])
        if len(domainNodes) < 1:
            raise RuntimeError, "No domain defined for discrete temperature field"

        # Only update density for nodes in the cancellous and subchondral bone sections
        cancellousNodes = [e.label for e in  self.__getAssignment(self.
            cancellousBoneSectName).getSet().nodes]
        subchondralNodes = [e.label for e in  self.__getAssignment(self.
            subchondralSectName).getSet().nodes]
        emptyEls = [e.label for e in  self.__getAssignment(self.emptySectName).getSet().
            elements]

        # get strain energy density
        sedField = output.steps[self.stepName].frames[1].fieldOutputs['SENER'].getSubset
            (position=ELEMENT_NODAL)

        # get SED only within the bone section
        elementLabels = [el.label for el in self.__getAssignment(self.
            cancellousBoneSectName).getSet().elements + \
            self.__getAssignment(self.subchondralSectName).getSet().elements]
        elementLabels.sort()
        odbElements = output.rootAssembly.instances[self.instanceName.upper()].\
            ElementSetFromElementLabels(name='InternalRemodellingSection',elementLabels=
                elementLabels)
        sedField = sedField.getSubset(region=odbElements)
        nodeLabels = [v.nodeLabel for v in sedField.values]

        # Remove duplicate entries from the domain nodes list, preserving order
        # these occur when splitting the temperature (density) domain into multiple
            surfaces
        domainNodesTemp = []
        densitiesTemp = []
        for (n,d) in zip(domainNodes,densities):
            if n not in domainNodesTemp:
                domainNodesTemp.append(n)
                densitiesTemp.append(d)
        domainNodes = domainNodesTemp
        densities = densitiesTemp
        del domainNodesTemp, densitiesTemp

        sed = [0.0]*len(domainNodes)
        for (i,n) in enumerate(domainNodes):
            # using getSubset to return the value at a node only uses the value from one
                 element, rather
            # than averaging the values from all elements sharing a node
            node = output.rootAssembly.instances[self.instanceName.upper()].
                getNodeFromLabel(n)
            sedAtNode = sedField.getSubset(region=node)
            # there will be nodes in domainNodes that are in the empty section, so will
                not be
            # in the strain energy density Field
            if len(sedAtNode.values) != 0:
                # most sedNodes only have lengths of 1, some have 2. (3500 vs 150)
```

```
                # not sure why some have lengths of 2. Ideally most would be 4.
                # Just use first value:
                sed[i] = sedAtNode.values[0].data

        # arrays for density field output and strain energy per unit mass
        # used to generate field outputs
        fieldDomain = []
        densityFieldData = []
        semFieldData = []

        # implement bone growth model
        for i in range(len(domainNodes)):
            inBone = False
            if domainNodes[i] in cancellousNodes:
                semFieldData.append([sed[i]/(densities[i]+0.01)]) # prevent division by
                    zero
                densities[i] = densities[i] + self.B*((sed[i]/(densities[i]+0.01)) -
                    self.equilibriumStrain)
                inBone = True
            elif domainNodes[i] in subchondralNodes:
                semFieldData.append([sed[i]/(densities[i]+0.01)])
                densities[i] = densities[i] + self.B*((sed[i]/(densities[i]+0.01)) -
                    self.subchondralEquilibriumStrain)
                inBone = True
            else:
                pass # density unchanged
            if densities[i] > 1.0:
                densities[i] = 1.0
            if densities[i] < 0.0:
                densities[i] = 0.0
            if inBone:
                fieldDomain.append(domainNodes[i])
                densityFieldData.append([densities[i]])

        # make a field output variable for the density and strain
        # energy density per unit mass so these can be visualised
        frame = output.steps[self.stepName].frames[1]
        instance = output.rootAssembly.instances[self.instanceName.upper()]
        densityField = frame.FieldOutput(name='density', description='updated density
            field for cancellous bone', type=SCALAR)
        densityField.addData(position=NODAL,labels=fieldDomain,data=densityFieldData,
            instance=instance)
        semField = frame.FieldOutput(name='SEM', description='strain energy density per
            unit mass', type=SCALAR)
        semField.addData(position=NODAL,labels=fieldDomain,data=semFieldData,instance=
            instance)
        output.save()

        # write density data to file at specified nodes:
        if len(self.recordDensityNodes) > 0:
            outfile = open(self.outputFileName,'a')
            for n in self.recordDensityNodes:
                outfile.write("%f " % frame.fieldOutputs['density'].getSubset(region=
                    instance.getNodeFromLabel(n)).values[0].data)
            outfile.write('\n')
            outfile.close()

        data = ((self.instanceName,1,domainNodes,densities),)
        tempField.setValues(data=data)

    def __surfaceRemodel(self,output,section):
        """
        Reassign bone section based on strain field from previous loading
        Parameters:
            output: Abaqus output database
            section: Model section to add and remove elements from.
```

```
            Removed elements are added to the empty section.
        Return value:
            None
        """
        print "Surface remodelling"
        boneSect = self.__getAssignment(section)
        emptySect = self.__getAssignment(self.emptySectName)

        boneElements = [e.label for e in boneSect.getSet().elements]
        emptyElements = [e.label for e in emptySect.getSet().elements]

        # get strain energy per unit mass, calculated in the density
        # remodelling function
        semField = output.steps[self.stepName].frames[1].\
            fieldOutputs['SEM'] # position=NODAL
        # average SE at nodes to get value at centroid
        sem = [0.0]*len(boneElements)
        for (i,n) in enumerate(boneElements):
            element = output.rootAssembly.instances[self.instanceName.upper()].
                getElementFromLabel(n)
            nodes = [output.rootAssembly.instances[self.instanceName.upper()].
                getNodeFromLabel(cn) for cn in element.connectivity]
            semAtNodes = [semField.getSubset(region=n) for n in nodes]
            sem[i] = average([n.values[0].data for n in semAtNodes if (len(n.values) >
                0)])
        semField = output.steps[self.stepName].frames[1].FieldOutput(\
            name='SEME', description='strain energy density per unit mass', type=SCALAR)
        semField.addData(position=CENTROID,labels=boneElements,data=[[s] for s in sem],\
            instance=output.rootAssembly.instances[self.instanceName.upper()])
        output.save()

        # modify list of boneElements and emptyElements by growing bone
        # in regions of high strain
        print "  Adding new bone"
        newBoneElements = self.__growNewBone(boneElements,sem,emptyElements)
        boneElements.extend(newBoneElements)
        for el in newBoneElements:
            emptyElements.remove(el)
        output.rootAssembly.instances[self.instanceName.upper()].\
            ElementSetFromElementLabels(name="newBoneElements",elementLabels=
                newBoneElements)

        # modify boneElemenst and emptyElements by killing off bone
        # where there is low strain
        print "  Killing bone"
        killedBoneElements = self.__killBone(boneElements,sem)
        [boneElements.remove(kbe) for kbe in killedBoneElements]
        emptyElements.extend(killedBoneElements)
        output.rootAssembly.instances[self.instanceName.upper()].\
            ElementSetFromElementLabels(name="killedBoneElements",elementLabels=
                killedBoneElements)

        # delete previous sets if they exist
        self.part.deleteSets(["boneElements","emptyElements"])
        # create new sets of elements
        newBoneRegion = self.part.\
            SetFromElementLabels(name="surfaceRemodelledElements",elementLabels=
                boneElements)
        newEmptyRegion = self.part.\
            SetFromElementLabels(name="emptyElements",elementLabels=emptyElements)
        # assign element sets to section assignments
        print "  Updating regions"
        boneSect.setValues(region=newBoneRegion)
        emptySect.setValues(region=newEmptyRegion)

        if len(boneElements) == 0:
```

```
            raise RuntimeError, "Removed all bone elements while remodelling surface"

    def __growNewBone(self,elements,strainMeasure,emptyElements):
        """
        Grow new bone in regions of high strain
        Return a list of elements that are currently empty
        that are to be turned into bone
        Parameters:
            elements: list of bone elements at the surface to add to
            strainMeasure: list of strain measure (SED) corresponding to given elements
            emptyElements: list of elements that are currently specified as empty
        Return value:
            List of element labels of new bone elements to add to set
        """

        # get elements where bone growth will occur
        growthElementLabels = [e for (e,strain) in zip(elements,strainMeasure) \
            if strain > self.surfaceStrainLevels[1]]

        newBoneElements = []
        # find empty elements around growth elements
        for ge in growthElementLabels:
            connectedEls = self.__getConnectedEls(ge)
            newBoneElements.extend([c for c in connectedEls if c in emptyElements \
                if c not in newBoneElements])
        return newBoneElements

    def __killBone(self,elements,strainMeasure):
        """
        Kill off bone in regions of low strain
        Return a list of elements that are currently bone
        that are to become empty
        Parameters:
            elements: list of bone elements at the surface to add to
            strainMeasure: list of strain measure (SED) corresponding to given elements
        Return value:
            List of element labels of bone elements to remove
        """

        killedBoneElements = [e for (e,strain) in zip(elements,strainMeasure) \
            if strain < self.surfaceStrainLevels[0]]

        return killedBoneElements

    def __cartilageDamage(self,output):
        """
        Simulate damage to cartilage by decreasing stiffness if
        stress in the cartilage is above a certain threshold. Use
        a temperature field similar to the method used to model
        cancellous bone.
        """
        print "Modelling cartilage damage"
        tempField = self.model.discreteFields[self.cartilageFieldName]

        domainNodes = tempField.data[0].domain[:]
        damage = list(tempField.data[0].table[:])
        if len(domainNodes) < 1:
            raise RuntimeError, "No domain defined for discrete temperature field in
                cartilage"

        # get invariants of stress field in cartilage
        stressField = output.steps[self.stepName].frames[1].\
            fieldOutputs['S'].getSubset(position=ELEMENT_NODAL)
        sf1 = stressField.getScalarField(MAX_PRINCIPAL)
        sf3 = stressField.getScalarField(MIN_PRINCIPAL)
```

```
        fieldDomain = []
        maxShearFieldData = []
        damageFieldData = []

        maxShearStress = [0.0]*len(domainNodes)
        for (i,n) in enumerate(domainNodes):
            node = output.rootAssembly.instances[self.instanceName.upper()].
                getNodeFromLabel(n)
            sn = [sf1.getSubset(region=node), sf3.getSubset(region=node)]
            if len(sn[0].values) != 0:
                maxShearStress[i] = (sn[0].values[0].data - abs(sn[1].values[0].data)) /
                    2.0
            if abs(maxShearStress[i]) > self.cartilageDamageStress:
                damage[i] = max(damage[i]+0.1, 0.0)
            damageFieldData.append([damage[i]])
            maxShearFieldData.append([maxShearStress[i]])

        frame = output.steps[self.stepName].frames[1]
        instance = output.rootAssembly.instances[self.instanceName.upper()]
        densityField = frame.FieldOutput(name='cartilageDamage', description='cartilage
            damage', type=SCALAR)
        densityField.addData(position=NODAL,labels=domainNodes,data=damageFieldData,
            instance=instance)
        maxShearField = frame.FieldOutput(name='maxShearStress', description='maximum
            shear stress', type=SCALAR)
        maxShearField.addData(position=NODAL,labels=domainNodes,data=maxShearFieldData,
            instance=instance)
        output.save()

        data = ((self.instanceName,1,domainNodes,damage),)
        tempField.setValues(data=data)

    def __getConnectedEls(self,el):
        """
        Parameters:
            el: element label to find connected elements for
        Return value:
            A list of element labels that share and edge with the given element
        """
        element = self.part.elements.getFromLabel(el)
        edges = element.getElemEdges()
        connectedEls = []
        for edge in edges:
            for ce in edge.getElements():
                if (ce.label != el) and ce.label not in connectedEls:
                    connectedEls.append(ce.label)
        return connectedEls

    def __getAssignment(self,sectionName):
        """
        Return a section from a section assignment
        Parameters:
            sectionName: section to find
        Return value:
            the section assignement object
        """
        section = [sa for sa in self.part.sectionAssignments \
            if sa.sectionName == sectionName]
        if len(section) == 0:
            raise RuntimeError, "Section not found"
        return section[0]

    def initiateCancellous(self,density):
        """
        Initiate a temperature discrete field
        with a constant value
```

```
            Parameters:
                density: Density proportion of bone to fill with initially
            Return value:
                None
            """
            domainNodes = [n.label for n in \
                self.__getAssignment(self.cancellousBoneSectName).getSet().nodes]
            domainNodes.extend([n.label for n in \
                self.__getAssignment(self.subchondralSectName).getSet().nodes])
            domainNodes.extend([n.label for n in \
                self.__getAssignment(self.emptySectName).getSet().nodes])
            densities = [density] * len(domainNodes)

            data = [[self.instanceName,1,domainNodes,densities],]

            temp = self.model.discreteFields[self.discreteTempFieldName]
            temp.setValues(data=data)

        def initiateCartilage(self):
            """
            Initiate a temperature discrete field
            with a constant value for cartilage damage
            Return value:
                None
            """
            domainNodes = [n.label for n in \
                self.__getAssignment(self.cartilageSectName).getSet().nodes]
            damages = [0.0] * len(domainNodes)
            data = [[self.instanceName,1,domainNodes,damages],]
            temp = self.model.discreteFields[self.cartilageFieldName]
            temp.setValues(data=data)

    def average(x):
        """
        return the mean value of a list
        """
        return reduce(add,x) / float(len(x))

    if __name__ == "__main__":
        pass
```

References

Ahmed, A. M. and Burke, D. L. (1983) In-vitro of measurement of static pressure distribution in synovial joints—part I: Tibial surface of the knee. *J Biomech Eng* 105(3): 216–225.

Akizuki, S., Mow, V. C., Muller, F., Pita, J. C., Howell, D. S. and Manicourt, D. H. (1986) Tensile properties of human knee joint cartilage: I. Influence of ionic conditions, weight bearing, and fibrillation on the tensile modulus. *J Orthop Res* 4(4): 379–392.

Carter, D. R. (1987) Mechanical loading history and skeletal biology. *J Biomech* 20(11-12): 1095–1109. F. Gaynor Evans Anniversary Issue on Bone Biomechanics.

Charras, G. T. and Horton, M. A. (2002) Single cell mechanotransduction and its modulation analyzed by atomic force microscope indentation. *Biophys J* 82(6): 2970–2981.

Chen, G., Pettet, G., Pearcy, M. and McElwain, D. (2007) Comparison of two numerical approaches for bone remodelling. *Med Eng Phys* 29(1): 134–139.

Guldberg, R. E., Gemmiti, C. S., Kolambkar, Y. and Porter, B. (2008) Physical stress as a factor in tissue growth and remodeling. In *Principles of Regenerative Medicine*, pp. 512–535. Academic Press, San Diego.

He, G. and Xinghua, Z. (2006) The numerical simulation of osteophyte formation on the edge of the vertebral body using quantitative bone remodeling theory. *Joint Bone Spine* 73(1): 95–101.

Huiskes, R., Weinans, H. and Van Rietbergen, B. (1992) The relationship between stress shielding and bone resorption around total hip stems and the effects of flexible materials. *Clin Orthop Relat Res* 274: 124–134.

Jackson, B., Wluka, A., Teichtahl, A., Morris, M. and Cicuttini, F. (2004) Reviewing knee osteoarthritis – a biomechanical perspective. *J Sci Med Sport* 7(3): 347–357.

Jacobs, C. R., Levenston, M. E., Beaupré, G. S., Simo, J. C. and Carter, D. R. (1995) Numerical instabilities in bone remodeling simulations: The advantages of a node-based finite element approach. *J Biomech* 28(4): 449–459.

Karsdal, M., Leeming, D., Dam, E., Henriksen, K., Alexandersen, P., Pastoureau, P., Altman, R. and Christiansen, C. (2008) Should subchondral bone turnover be targeted when treating osteoarthritis? *Osteoarthritis Cartilage* 16(6): 638–646.

Martin, J. A. and Buckwalter, J. A. (2002) Aging, articular cartilage chondrocyte senescence and osteoarthritis. *Biogerontology* 3(5): 257–264.

Peña, E., Calvo, B., Martínez, M. A. and Doblaré, M. (2008) Computer simulation of damage on distal femoral articular cartilage after meniscectomies. *Comput Biol Med* 38(1): 69–81.

Radin, E. L. and Burr, D. B. (1984) Hypothesis: Joints can heal. *Semin Arthritis Rheum* 13(3): 293–302.

Radin, E. L. and Rose, R. M. (1986) Role of subchondral bone in the initiation and progression of cartilage damage. *Clin Orthop Relat Res* 213: 34–40.

Rho, J. Y., Ashman, R. B. and Turner, C. H. (1993) Young's modulus of trabecular and cortical bone material: Ultrasonic and microtensile measurements. *J Biomech* 26(2): 111–119.

Shirazi, R. and Shirazi-Adl, A. (2009) Computational biomechanics of articular cartilage of human knee joint: Effect of osteochondral defects. *J Biomech* In Press, Corrected Proof.

Taylor, W. R., Heller, M. O., Bergmann, G. and Duda, G. N. (2004) Tibio-femoral loading during human gait and stair climbing. *J Orthop Res* 22(3): 625–632.

Tsubota, K., Adachi, T. and Tomita, Y. (2002) Functional adaptation of cancellous bone in human proximal femur predicted by trabecular surface remodeling simulation toward uniform stress state. *J Biomech* 35(12): 1541–1551.

Tsubota, K., Suzuki, Y., Yamada, T., Hojo, M., Makinouchi, A. and Adachi, T. (2009) Computer simulation of trabecular remodeling in human proximal femur using large-scale voxel FE models: Approach to understanding Wolff's law. *J Biomech* 42(8): 1088–1094.

Turner, A., Gillies, R., Sekel, R., Morris, P., Bruce, W. and Walsh, W. (2005) Computational bone remodelling simulations and comparisons with DEXA results. *J Orthop Res* 23(4): 705–712.

Turner, C. H. (1998) Three rules for bone adaptation to mechanical stimuli. *Bone* 23(5): 399–407.

Turner, C. H., Anne, V. and Pidaparti, R. M. V. (1997) A uniform strain criterion for trabecular bone adaptation: Do continuum-level strain gradients drive adaptation? *J Biomech* 30(6): 555–563.

Valdes, A. M. and Spector, T. D. (2009) The contribution of genes to osteoarthritis. *Med Clin N Am* 93(1): 45–66.

Walker, P. S. and Hajek, J. V. (1972) The load-bearing area in the knee joint. *J Biomech* 5: 581–589.

Wei, H.-W., Sun, S.-S., Jao, S.-H. E., Yeh, C.-R. and Cheng, C.-K. (2005) The influence of mechanical properties of subchondral plate, femoral head and neck on dynamic stress distribution of the articular cartilage. *Med Eng Phys* 27(4): 295–304.

Wilson, W., van Burken, C., van Donkelaar, C. C., Buma, P., van Rietbergen, B. and Huiskes, R. (2006) Causes of mechanically induced collagen damage in articular cartilage. *J Orthop Res* 24(2): 220–228.

Zhao, D., Banks, S. A., D'Lima, D. D., Jr., C. W. C. and Fregly, B. J. (2007) In vivo medial and lateral tibial loads during dynamic and high flexion activities. *J Orthop Res* 25: 593–602.