Delft University of Technology

Department of Precision and Microsystems Engineering

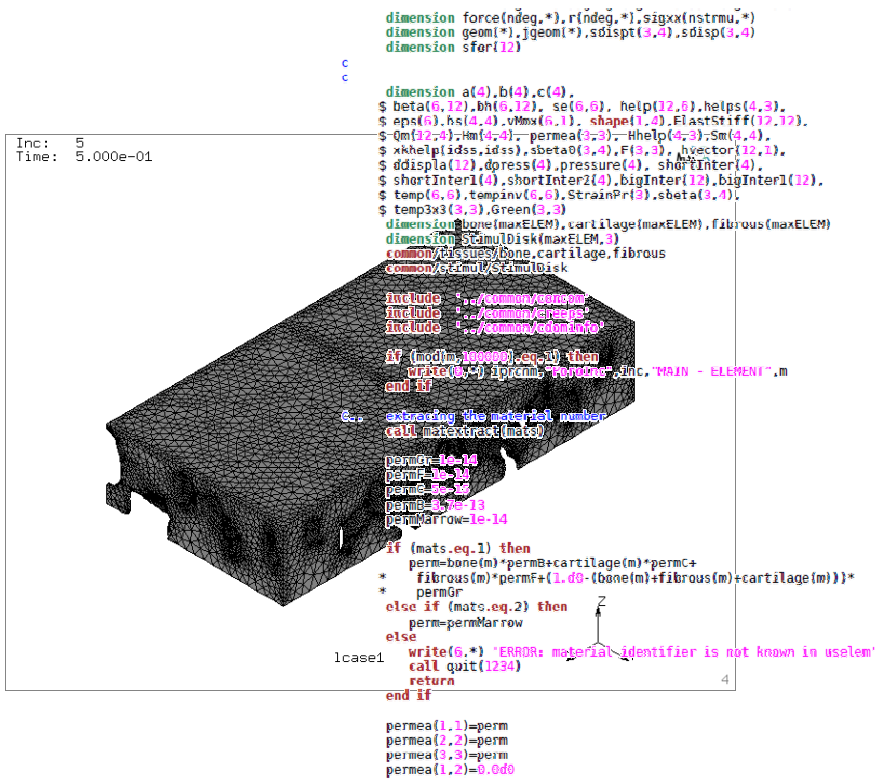
A feasibility study on the acceleration and upscaling of bone ingrowth simulation

Name: Yoeng Sin Khoe

Report no: EM 09.014
Coach:
Professor: Prof. Dr. Ir. Fred van Keulen
Specialisation: Tissue Biomechanics & Implants
Type of report: MSc Thesis
Date: June 30th, 2009

A feasibility study on the acceleration and upscaling of bone ingrowth simulation

AN INVESTIGATION INTO THE FEASIBILITY OF APPLYING A HOMOGENIZATION SCHEME TO BONE INGROWTH MODEL AS WELL OTHER OPTIONS OF ACCELERATING THE BONE INGROWTH MODEL DEVELOPED BY A. ANDREYKIV.



NAME: YOENG SIN KHOE

EMAIL: YS.KHOE@STUDENT.TUDELFT.COM

STUDENT NUMBER: 1062425

COURSE CODE: BMA0340

DATE: JUNE 30TH, 2009

Preface

This thesis marks the end of my MSc track at the faculty of Biomedical Engineering. Within the faculty I have followed the track 'Tissue Biomechanics and Implants'.

During the course of this my graduation work I have learned a lot. This investigation has taught me a lot on some of the essentials behind soft-tissue biomechanics, finite element codes and Fortran programming. But what I will probably will remember most is the value of planning, or rather the pitfall of the lack of planning.

I would like to take this opportunity to thank prof. dr. ir. Fred van Keulen and dr. ir. Andriy Andreykiv for their support and comments during my graduation period.

Finally I would like to thank my girlfriend, my family and my friends for their everlasting friendship support and critical views that I highly value, but do not always follow.

Yoeng Sin Khoe
Rotterdam, June 30th, 2009

Abstract

Fixation of uncemented implants is known to be more problematic than cemented implants as the proces relies on the growth of bone into the porous implant surface. This bone growth determines the fixation of the implant and has been the subject of many investigations [1-6]. A literature survey [7] revealed that these models are either a highly detailed simulation of a small section of the interface between bone and implant or a more generalized simulation of a complete implant. This thesis is aimed at bridging the gap between the two.

An investigation of the bone ingrowth model by Andreykiv [1] revealed possibilities of accelerating this simulation. Furthermore a feasibility study was performed to apply computational homogenization in order to upscale the results of the detailed microscopic model to a higher level.

Investigation of the bone ingrowth model showed that it cannot be simplified. All elements of the model are essential in predicting the tissue growth within the system. A 65% gain in speed of the simulation was obtained by optimizing the code of the subroutines (written for MSC Marc). Furthermore the feasibility study on computational homogenization shows that the implementation in a commercial code leads to extremely long computing times.

In light of these results it is concluded that homogenization is not the method to bridge the gap between the detailed microscopic simulations and the simulations of the complete implants. Recommendations are given on the continued research to use the results of the detailed simulation on a higher level.

Table of Contents

1 Introduction

Diseases such as arthritis or osteoporosis damage the joints. Such diseases and their effects are becoming more and more common due to the aging of the population[1][2]. This increases the need for medical treatments and surgical procedures to aid and prevent people who suffer from the effects of these diseases. One option for advanced stages of arthritis is a total joint replacement, for example a shoulder replacement is depicted in Figure 1.

Figure 1 Example of a shoulder replacement[3]

Such an implant must be attached to the existing bone-structure. This fixation can be achieved through either a cemented or an ingrowth implant. The cemented implant employs a cement to glue the implant to the bone. An ingrowth implants is press-fitted after which natural bone growth will fixate the implant to the bone

While a cemented implant may provide better initial fixation, it is more prone to damage at a later stage due to high stresses in the cement layer [8]. An ingrowth type implant will, given adequate growth conditions, provide a better fixation. Bone will grow into the implant, interlock with the implant and maintain the connection through the natural process of bone remodeling.

In order to gain a deeper understanding of how bone ingrowth develops between implant and bone, models [1, 2, 4-6] have been developed that models a section of this interface. One of these models was developed by Andreykiv [1] in which a the bone growth in a complex porous coating (see Figure 2) was studied.

[1] Rijksinstituut voor Volksgezondheid en Milieu (RIVM) – (http://www.rivm.nl/vtv/object_document/o1794n18373.html)

[2] Rijksinstituut voor Volksgezondheid en Milieu (RIVM) – (http://www.rivm.nl/vtv/object_document/o1716n18370.html)

[3] Source: Zimmer, http://www.zimmer.com

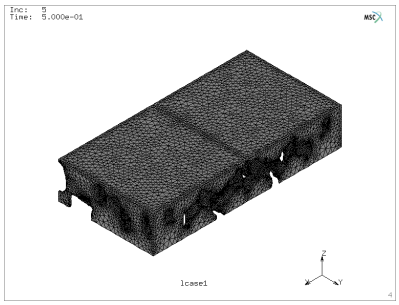


Figure 2 **FEM geometry for the bone ingrowth model developed by Andreykiv [1]**

The knowledge that is gained from FEM models such as these can be applied to increase the knowledge of implant fixation and could be used to design better prosthesis' or develop better surgical procedures that ensure better implant fixation. It has already been shown that improved implant design leads to less complications [9].

Although adequate simulations on the development of bone around implants exist [7], the implants models are highly simplified and rather much more detailed analyses are preferred, such as those developed by Andreykiv [1] or Liu [2]. The level of detail in the model presented above is very high and implementing such a model for a full scale implant analysis will result in impractical simulation duration. The model of this section of the interface requires about 3 days on 3 dual core processors. A full scale detailed implant analysis would probably take months of computing time.

It would be preferable if the knowledge gained on this level could be used in simulations on the larger scale. This thesis shall investigate ways to reduce the time needed for this simulation. Furthermore, an investigation will be made into the possibility of applying a homogenization scheme that allows the results of these detailed microstructural simulations to be used on a higher level (i.e. macrostructural simulations). 3 methods of reducing the computation time are investigated; 1) homogenization, 2) model reduction and 3) code optimization.

Chapter 2 of this thesis will develop the necessary theory. It covers some basic mechanics as well as the theory of the computational homogenization scheme. Chapter 3 describes the bone ingrowth model developed by Andreykiv. With the knowledge of the bone ingrowth model, Chapter 4 identifies possible routes for accelerating the simulations and defines the feasibility study for the computational homogenization scheme. Chapter 5 gives the results of model optimizations. Chapter 6 gives the results of code optimizations and Chapter 7 discusses the implementation and results of the computational homogenization algorithm. Chapter 8 summarizes the results and formulates the conclusion. Finally Chapter 9 gives recommendations for further research.

2 Theory

This chapter is a summary of the mathematics, mechanics required for understanding the mechanics of bone ingrowth and the method of homogenization. It serves as a reference for the methods that are used in this thesis.

2.1 Notation

The following notation for mathematical quantities is used in this chapter.

A or a	Capital or lowercase letter	Scalar
$\vec{a}$	Lowercase letter with an arrow	Vector
$\boldsymbol{A}$	Bold capital	Matrix
${}^{n}\boldsymbol{A}$	Bold capital with a number in upper left side	nth-order tensor
C_{ijkl}	Scalar notation for tensors.	Scalar
e_i	Basis vectors of a coordinate system	Scalar
$\blacksquare_m$	Quantities on the microscopic level	<subscript>
$\blacksquare_M$	Quantities on the macroscopic level	<subscript>

2.2 Tensor algebra

The purpose of tensor calculus is to separate calculations from their reference coordinate system. Such a method allows the calculations to be performed in any arbitrary coordinate system and eases the problems that arise when calculations span two (or more) non-corresponding coordinate systems.

The separation of the coordinate system is achieved through the notion of covariant and contravariant tensors. In this thesis, all calculations are done in a (rectilinear) Cartesian space. Therefore the notion of covariant and contravariant tensors becomes unimportant as the two become identical [10].

Further simplifications that are inherent to a Cartesian space allow the interpretation of tensors as an extension to matrix/vector algebra. Where matrix/vector algebra is limited to 2 dimensional arrays, tensor algebra allows calculations with n-dimensional arrays.

Using the simplified view on tensor calculus, the following operations can be described.

Conjugation for 2 2nd order tensors	$A^{c}_{ij} = A_{ji}$
Dyadic Product between 2 1-dimensional tensors	$\boldsymbol{C} = a \otimes b = \vec{a}\vec{b},$ $C_{ij} = a_i b_j e_i e_j$
Inner product for 1st order tensors	$C = \vec{a} \cdot \vec{b}$

	$C = a_i b_i$
Double Inner product for 2nd order tensor	$C = \boldsymbol{A}:\boldsymbol{A}$
	$C = A_{ij}B_{ji}$
Double Inner product for 4th & 2nd order tensor	$\boldsymbol{C} = {}^{\mathbf{4}}\boldsymbol{A}:\boldsymbol{B}$
	$C_{ij} = A_{ijkl}B_{lk}e_i e_j$
Double Inner product for 4th order tensor	$\boldsymbol{C} = {}^{\mathbf{4}}\boldsymbol{A}:{}^{\mathbf{4}}\boldsymbol{B}$
	$C_{ijkl} = A_{ijmn}B_{nmkl}e_i e_j e_k e_l$
Quadruple Inner product for 4th order tensor	$C = {}^{\mathbf{4}}\boldsymbol{A}\vdots{}^{\mathbf{4}}\boldsymbol{B}$
	$C = A_{ijkl}B_{lkji}$

2.3 Stress & Strain in Continuum Mechanics

In continuum mechanics the deformation of a body under a load is studied. In this concept, 2 states of a body can be defined, namely the undeformed and the deformed body.

In order to characterize this deformation an infinitesimal piece of material $d\vec{X}$ is tracked as it deforms into $d\vec{x}$. This deformation, expressed in terms of the undeformed coordinate system is expressed by

$$d\vec{x} = \frac{\partial \vec{x}}{\partial \vec{X}} d\vec{X} = \boldsymbol{F} d\vec{X}.$$

This deformation and the deformation tensor $\boldsymbol{F}$ are schematically shown in Figure 3**.**

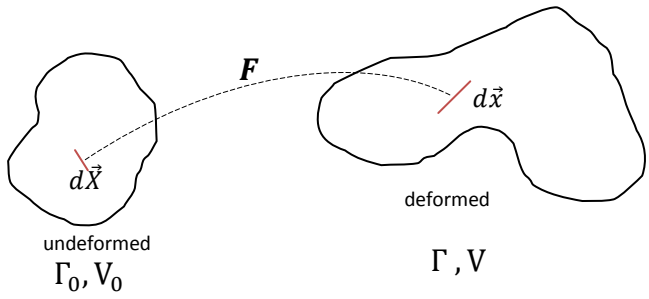


Figure 3 Definition of states and deformation measure

The following sections develop the concepts of strain and stress that are applicable to this thesis.

2.3.1 Strain Measures

Strain quantifies the deformation of a body. The deformation that a material point (with corresponding position vector) undergoes can be quantified as the difference between the magnitudes of the position vectors

$$d\vec{x}\cdot d\vec{x} - d\vec{X}\cdot d\vec{X} = d\vec{X}\cdot(\boldsymbol{F}^T\boldsymbol{F} - \boldsymbol{I})\cdot\ d\vec{X} = d\vec{x}\cdot(\boldsymbol{I} - (\boldsymbol{F}^{-1})^T\boldsymbol{F}^{-1})\cdot d\vec{x}$$

It can be seen that deformations can quantified with reference to the deformed or the undeformed state.

When deformations are given with respect to the deformed configuration, the Green strain tensor ($\boldsymbol{E}$) is used, which is defined as

$$\boldsymbol{E} = \frac{1}{2}(\boldsymbol{F}^T\boldsymbol{F} - \boldsymbol{I}) = \frac{1}{2}(\boldsymbol{C} - \boldsymbol{I}),$$

Where $\boldsymbol{C}$ is named the Right Cauchy Green Deformation tensor.

When deformations are given with respect to the undeformed configuration, the Almansi strain tensor ($\boldsymbol{e}$) is used, which is defined as

$$\boldsymbol{e} = \frac{1}{2}(\boldsymbol{I} - (\boldsymbol{F}^{-1})^T\boldsymbol{F}^{-1}) = \frac{1}{2}(\boldsymbol{I} - \boldsymbol{b}^{-1}),$$

Where $\boldsymbol{b}$ is named the Left Cauchy's deformation tensor or the Finger tensor.

This thesis will take the Lagrangian approach in order to define the problems.

2.3.2 Stress Measures

In small deformation mechanics, the Cauchy stress tensor is most widely used. The tensor is completely defined in the deformed configuration.

When large deformations are present, the deformed area is generally not known and a new stress measure is defined. The 1st Piola-Kirchhoff stress tensor relates the forces in the deformed configuration to the area in the undeformed configuration. This can be achieved for example by mapping the Cauchy stresses to the undeformed configuration

$$\boldsymbol{P} = \boldsymbol{\sigma} J \boldsymbol{F}^{-1},$$

Where $J = \det(\boldsymbol{F})$. The 1st Piola-Kirchhoff stress tensor is not symmetric, because the deformation tensor is generally not symmetric.

The introduction of the 2nd Piola-Kirchoff stress tensor allows the stresses to be decoupled from the reference coordinates, resulting in a generalized interpretation of the forces. This second mapping is achieved by

$$\boldsymbol{S} = \mathrm{J}\boldsymbol{F}^{-1}\boldsymbol{\sigma}\boldsymbol{F}^{-T}.$$

Examining the equations for the stress tensors, we can easily relate the 1st and 2nd PK tensor by a simple mapping using the deformation tensor

$$\boldsymbol{S} = \boldsymbol{F}^{-1}\boldsymbol{P} \iff \boldsymbol{P} = \boldsymbol{F}\boldsymbol{S}.$$

The decoupling of the stresses and the reference coordinate system provides a power tool in which an arbitrary deformation with its corresponding stress state can easily be described. In additions the 2nd PK tensor is symmetric, which in numerical analyses results a more efficient use of processor power and memory.

The physical interpretation of the 1st PK tensor eases the implementation of such a stress measure.

2.3.3 Work conjugated couples

The product of the stress and strain should result in the work done. Therefore stress and strain measures are so called conjugate couples. 3 Common examples are:

Cauchy stress tensor ($\boldsymbol{\sigma}$)	Engineering strain or natural strain($\boldsymbol{\varepsilon}$)
1st Piola-Kirchoff stress tensor ($\boldsymbol{P}$)	Deformation Tensor ($\boldsymbol{F}$)
2nd Piola-Kirchoff stress tensor ($\boldsymbol{T}$)	Green Strain Tensor ($\boldsymbol{E}$)

The selection of stress and strain measures is arbitrary, but in order to be able to correctly determine the work a work conjugated couple must be selected. In light of computational homogenization (developed in Chapter 2.4) the 1st Piola-Kirchhoff stress tensor and the deformation tensor are selected. It is argued [11] that the combination of 1st Piola-Kirchoff and deformation tensor is most suitable for reasons of ease of implementation. The 1st Piola-Kirchoff stress van be determined using the forces in the deformed configuration in combination with known values of the undeformed setting.

2.4 Computational Homogenization

Homogenization is a method that takes inhomogeneities in a model and allows them to be presented as a homogeneous system. Materials handled in a mechanical analysis are usually considered homogeneous. Take for instance the simply supported beam, a classical textbook example with which student are handed the principles of forces and deflections. The simply supported beam is assumed to be constructed of a homogeneous material like steel. However the reality is quite different, steel is a material consisting of multiple constituents that interact at grain boundaries. Yet through experience and testing we have found that the characteristics of this heterogeneous material can be captured in coefficients that predict the response of the global structure in specific conditions.

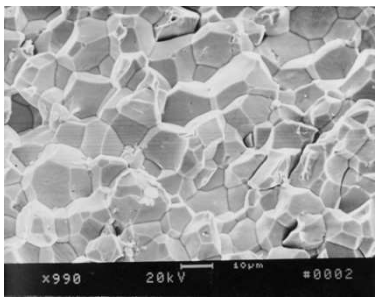


Figure 4 close up image of steel, showing the inhomogeneous nature of steel[4]

Although this detailed microstructure actually determines the properties of the material, on a much larger scale, the material appears to be homogeneous with certain 'effective

[4] Source: http://www.affiliatedinc.net/lab/case.html

properties'. These effective properties can be obtained be either testing the material extensively, or by computing the effective properties based on the knowledge of the structure on microscopic level. Homogenization is the term used to describe the later process of computing effective properties at a macroscopic scale using information that is available on the microscopic scale.

In the literature survey [7] a comparison of various upscaling methods was made from which it was concluded that computational homogenization was the choice of homogenization method for the bone-ingrowth model of Andreykiv [1].

The following sections will go into the theoretical background of the computational homogenization scheme. In accordance with Figure 3, coordinates in the undeformed configuration are denoted by $\vec{X}$ and coordinates in the deformed configuration are denoted by $\vec{x}$. The volume and the boundaries with subscript 0 denoted the respective quantities in the undeformed configuration.

Furthermore a distinction is made between a macroscopic model and a microscopic model. This setting is shown in Figure 5. Quantities in the macroscopic model have subscript M, whilst quantities at the microscopic level carry subscript m.

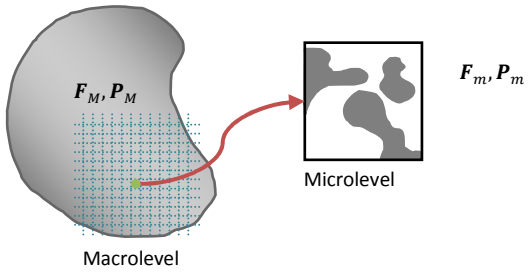


Figure 5 Definition of the macroscopic and microscopic level

The first step in the computational homogenization scheme is the selection of the parameters that are to be upscaled. The requirement is that these parameters together form a work conjugated couple, so that the Hill-Mandel condition can be satisfied (see 2.4.3). Although all work conjugated couples are valid choices, the implementation is aided with the choice for the combination of the deformation tensor and the 1st Piola-Kirchhoff stress tensor. It will be shown that this selection allows for an eased upscaling of the stress tensor (see 2.4.3).

2.4.1 Relating the Microscopic and Macroscopic Deformation Tensor

A non-linear deformation ($\Delta\vec{x}$) in the system imposed on a macroscopic level can be described by the linear deformation (as a function of the macroscopic deformation tensor $\boldsymbol{F}_M$)and additional higher order deformations, which will be named micro fluctuations ($\vec{w}$). This relation is given by

$$\Delta\vec{x} = \boldsymbol{F}_M \cdot \Delta\vec{X} + \vec{w}.$$

Using the notation $\frac{\partial}{\partial \vec{X}} = \nabla_0$, the microscopic deformation tensor is expressed as

$$\boldsymbol{F}_m = \frac{\partial \vec{x}}{\partial \vec{X}} = \nabla_0 \left(\left\{ \vec{X}_1 - \boldsymbol{F}_M \cdot \left(\vec{X}_1 - \vec{X}_p \right) \right\} - \boldsymbol{F}_M \cdot \vec{X}_p + \boldsymbol{F}_M \cdot \vec{X} + (\vec{w} - \vec{w}_1) \right) = \boldsymbol{F}_M + \nabla_0 (\vec{w} - \vec{w}_1),$$

where $\nabla_0 = \frac{\partial}{\partial \vec{X}}$. The microscopic deformation tensor is volume averaged over the original volume (V_0) resulting in a relation between the macroscopic deformation tensor ($\boldsymbol{F}_M$) and microscopic deformation tensor ($\boldsymbol{F}_m$) arises in the form of

$$\frac{1}{V_0} \int_{V_0} \boldsymbol{F}_m dV_0 = \frac{1}{V_0} \int_{V_0} \boldsymbol{F}_M + \nabla_0 (\vec{w} - \vec{w}_1) dV_0 = \boldsymbol{F}_M + \frac{1}{V_0} \int_{V_0} \nabla_0 (\vec{w} - \vec{w}_1) dV_0.$$

Using this relation, it can be stated that

$$\frac{1}{V_0} \int_{V_0} \boldsymbol{F}_m dV_0 = \boldsymbol{F}_M, \qquad 2.1$$

if and only if the micro fluctuations integrated over the volume are zero. This constraint can be reformulated into a boundary constraint using the Gauss' Theorem (with $\vec{N}$ as the outward normal to the boundary in the undeformed configuration). Furthermore, proper selection of $\vec{x}_1$, for example a prescribed boundary, will eliminate $\vec{w}_1$. The resulting integral over the boundary in the original configuration (Γ_0) is

$$\frac{1}{\Gamma_0} \int_{\Gamma_0} \vec{w} \cdot \vec{N} d\Gamma_0 = 0. \qquad 2.2$$

This constraint can be satisfied in several ways.

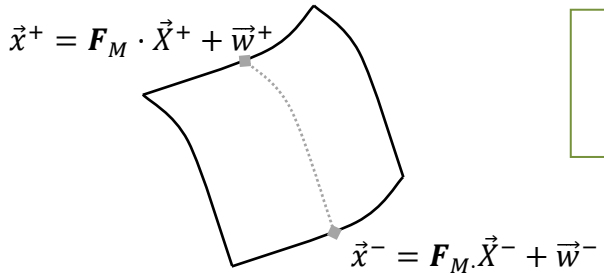
- Assume $\vec{w}$ to be zero in the entire volume *(Taylor or Voigt assumption)*
- Assume $\vec{w}$ to be zero along the boundary *(kinematically constraint boundaries or constant stress boundary conditions)*
- Assume $\vec{w} \cdot \vec{N}$ to be zero along the complete boundary *(e.g. periodic boundaries)*

Each assumption leads to a different set of boundary conditions that must be applied to the microscopic problem.

The various types of boundary conditions found above have been investigated [11] and it was found that periodic boundary conditions lead to a better estimate of the overall properties.

2.4.2 Periodic Boundary Condition Selection / Localization

Periodic boundary conditions imply that deformations on two opposing faces are identical ($\vec{x}^+ = \vec{x}^-$) and that the microfluctuations on either side are identical ($\vec{w}^+ = \vec{w}^-$). For example in a 2D quadrilateral model this is depicted as Figure 6.



Periodic Boundary Conditions

$$\vec{w}^+ = \vec{w}^- \qquad (2.3)$$
$$\vec{x}^+ - \vec{x}^- = \boldsymbol{F}_M \cdot \left(\vec{X}^+ - \vec{X}^-\right)$$

Figure 6 Periodic boundary conditions

The constraint imposed (Equation 2.2) by the volume averaging of the deformation tensor is satisfied and the use of Equation 2.1 is valid.

Furthermore as a consequence of equilibrium, the stresses on the boundaries are equal in magnitude and opposing in direction

$$\vec{p}^+ = -\vec{p}^-. \qquad (2.4)$$

This is also called anti-symmetric stress.

2.4.3 Upscaling the 1st Piola-Kirchhoff stress tensor

In order to create a consistent volume average for the stresses, the Hill-Mandel condition must be satisfied which is given by

$$\delta W_M = \delta W_m,$$

substituting the microscopic and macroscopic quantities for the stress and strain gives

$$\boldsymbol{P}_M : \delta \boldsymbol{F}_M^c = \frac{1}{V_0} \int_{V_0} \boldsymbol{P}_m : \delta \boldsymbol{F}_m^c dV_0.$$

The microscopic work can be calculated by using the tractions on the boundary and using the periodic boundary conditions to develop the right hand side into

$$\delta W_m = \frac{1}{V_0} \int_{\Gamma_0^+} \vec{p}^+ \vec{X}^+ d\Gamma_0^+ : \delta \boldsymbol{F}_M^c + \frac{1}{V_0} \int_{\Gamma_0^-} \vec{p}^- \vec{X}^- d\Gamma_0^- : \delta \boldsymbol{F}_M^c = \frac{1}{V_0} \int_{\Gamma_0} \vec{p} \vec{X} d\Gamma_0 : \delta \boldsymbol{F}_M^c.$$

When this results is plugged into the Hill-Mandel condition, we find

$$\boldsymbol{P}_M : \delta \boldsymbol{F}_M^c = \frac{1}{V_0} \int_{\Gamma_0} \vec{p}\vec{X} d\Gamma_0 : \delta \boldsymbol{F}_M^c,$$

from which the relation between the macroscopic stress tensor and the microscopic stress can be derived. This relation is given by

$$\boldsymbol{P}_M = \frac{1}{V_0} \int_{\Gamma_0} \vec{p}\vec{X} d\Gamma_0. \qquad 2.5$$

The Hill-Mandel condition imposes work equivalence in the macroscopic and microscopic level. The use of periodic boundary conditions in combination with the Hill-Mandel condition has led to a relation that allows the microscopic stresses to be translated to a macroscopic stress tensor.

Equation 2.5 shows that the volume average of the 1st Piola-Kirchoff is completely determined in terms of the underformed boundary in combination with forces in the deformed configuration. This combination is very convenient as the undeformed boundary is known beforehand and the resulting forces will be available during calculations. This exemplifies the choice for the 1st Piola-Kirchoff stress tensor and its work conjugated counterpart, the deformation tensor.

2.4.4 Determining the Macroscopic Tangent

In the numerical scheme a tangent is still required to complete the computation. The tangent relates the forces to the displacements. In a simple 1 dimensional case this tangent is the slope along the force-displacement curve, this is visualized in Figure 7.

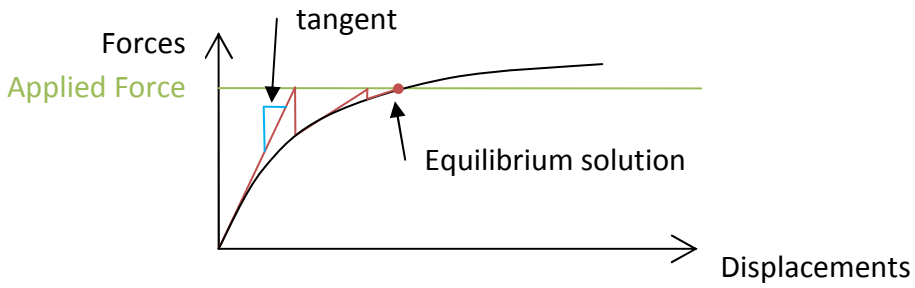


Figure 7 Visualization of the tangent that relates stress and strain

The tangent is used in the Newton-Raphson iterative method to solve for an equilibrium determined by the applied boundary condition (in this example an applied force). The nonlinear relation between force and displacement is approximated by a linearization a step is taken towards the equilibrium as long as the difference between the linear approximation and the actual stress is smaller than a set value. This steps that this method takes to

converge to a solution is indicated by the red lines in Figure 7 (see also [12]). This method and its derivates are implemented in non-linear packages such as MSC Marc.

Two methods are proposed by [11] to determine this tangent within the Computational Homogenization scheme. The first method is condensation of the global stiffness matrix (of the microscopic problem). In any system, the global stiffness matrix can be condensed by making the distinction between prescribed and free nodes. The resulting condensed stiffness matrix is the macroscopic stiffness matrix of the microscopic problem. Unfortunately this matrix is usually not available for calculations.

The alternative is numerical differentiation. Numerical differentiation predicts the response of the prescribed nodes for a small perturbation. The result is again the macroscopic response of the microscopic problem. Miehe [13] introduced a numerical differentiation scheme that reduces the number of calculations that are needed to create the macroscopic stiffness matrix.

If the global stiffness matrix is available, condensation of the global stiffness matrix is preferred. Numerical condensation is a computationally less efficient method that will introduces additional errors.

2.4.5 Limitations of 1st Order Computational Homogenization

Although 1st order Computational Homogenization is capable of handling non-linear simulations, there are restrictions. The linearization of the macroscopic deformation tensor is responsible for a limitation on the algorithm. The macroscopic deformation gradient must remain small with respect to the microscale. There is some question on the effectiveness of the computational homogenization approach. For each element, a microstructural analysis must be performed. Therefore most of the computational advantages will be gained in the potential parallelization of the numerical simulations [11].

3 Original Bone In-Growth Model

Before examining various model optimization schemes, the original model by Andreykiv will be detailed. In order to identify possible methods of increased performance it is imperative that its background and implementation is well understood.

3.1 Theoretical Setup

Based on a model for fracture healing [5], this model links a mechanical and a biological system in order to determine the evolution of the tissues. The 2 systems are weakly coupled by means of parameters. This system is, in its simplest form, described as Figure 8

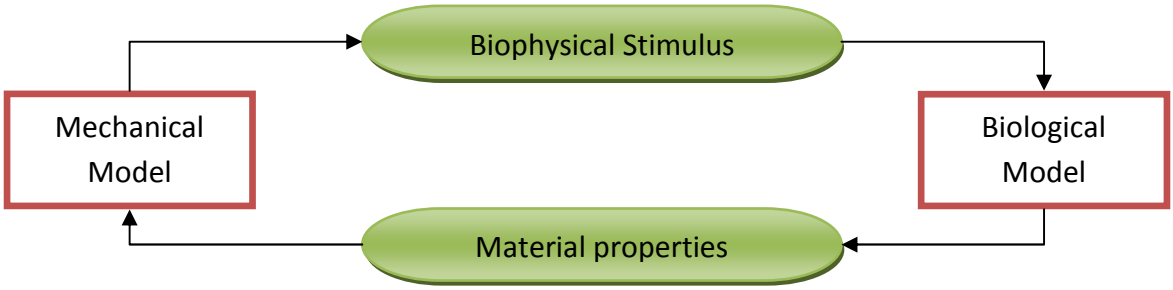


Figure 8 Coupling between mechanical and biological model.

In the following sections the theory behind the bone ingrowth model will be further elaborated. This theory is obtained from the dissertation of A. Andreykiv [1].

3.1.1 The Mechanical model

The mechanical response of the model is modeled using a biphasic model. Each constituent occupies a certain volume (V^α). To describe the density of a constituent a distinction is made between the true density (ρ_*^α) and the apparent density (ρ^α), where the first describes the density with respect to its own volume and the second is taken with respect to the total volume. This is reflected in the equations

$$\rho^\alpha = \frac{m^\alpha}{V} \qquad \rho_*^\alpha = \frac{m^\alpha}{V^\alpha} \qquad \Rightarrow \ \rho^\alpha = \frac{V^\alpha}{V}\rho_*^\alpha = \ n^\alpha \rho_*^\alpha .$$

The solid and fluid components of the biphasic material will be indicated by s or f indices.

Using the principle of virtual power, the balance of forces for each phase is given by

$$\vec{v}^s \, \rho^s - \nabla \cdot \boldsymbol{\sigma}^s - \rho^s \vec{f}^s - \vec{\Pi}^s = 0, \tag{3.1}$$

$$\vec{v}^f \, \rho^f - \nabla \cdot \boldsymbol{\sigma}^f - \rho^f \vec{f}^f - \vec{\Pi}^f = 0, \tag{3.2}$$

in which 4 terms are identified. The first term ($\vec{v}^\alpha \, \rho^\alpha$) expresses inertia forces. The second term ($\nabla \cdot \boldsymbol{\sigma}^\alpha$) describes the stress in the constituent. The 3rd term ($\rho^\alpha \vec{f}^\alpha$) shows the applied forces, such as gravity and the last term ($\vec{\Pi}^\alpha$) expresses the drag forces.

These equations are supplemented with the traction boundary conditions, where tractions are prescribed on the solids and a pressure is prescribed on the fluids as follows

$$\boldsymbol{\sigma}^s = \vec{n} \cdot (\boldsymbol{\sigma}_E^s - n^s p \boldsymbol{I}) = \boldsymbol{t}_s \; on \; \Gamma_{t_s},$$

$$\vec{n} \cdot \boldsymbol{\sigma}^f = \vec{n} \cdot -n^f p \boldsymbol{I} = \boldsymbol{t}_f \Leftrightarrow p = \tilde{p}(x) = -\frac{|\boldsymbol{t}_f|}{n^f} \; on \; \Gamma_{t_s}.$$

By virtue of the fact that the two volume fractions always constitute the total materials $(n^f + n^s = 1)$ and using the fact that the drag forces should cancel out $\left(\vec{\Pi}^f + \vec{\Pi}^s = 0\right)$, equations 3.1 and 3.2 can be combined into a single equation. Furthermore we assume that gravity and inertia forces are negligible. This leads to the simplified momentum Equation

$$\nabla \cdot \boldsymbol{\sigma} = 0. \qquad 3.3$$

This Equation 'masks' many the underlying equations, the most of important of which is the biphasic nature. Unlike a more conventional single constituent material an additional degree of freedom is introduced in the form of the fluid pressure. More specifically, the stresses (and pressure) in Equation 3.3 are divided as

$$\boldsymbol{\sigma} = \sum_\alpha \boldsymbol{\sigma}^\alpha = \boldsymbol{\sigma}_E^s - p\boldsymbol{I},$$

Where the Cauchy stress of the solid state $\boldsymbol{\sigma}_E^s$ is defined as

$$\boldsymbol{\sigma}_E^s = \boldsymbol{F}^s \cdot \boldsymbol{\tau}_E^s \cdot (\boldsymbol{F}^s)^T. \qquad 3.4$$

In Equation 3.4 $\boldsymbol{F}$ is the deformation tensor and $\boldsymbol{\tau}_E^s$ is the second Piola-Kirchoff stress tensor, which is given by the neo-Hookean model for hyperelastic solids. The neo-Hookean material model describes a compressible hyperelastic material. In combination with a fluid phase it exhibits many properties that are attributed to biological tissue [7]. The stress tensor for neo-Hookean material is

$$\boldsymbol{\tau}_E^s = \lambda^s ln J \boldsymbol{C}^{-1} + \mu^s (\boldsymbol{I} - \boldsymbol{C}^{-1}),$$

Where the coefficients λ^s and μ^s are Lamé's elasticity constants that represent the material properties, furthermore the Right Cauchy-Green tensor is given by $\boldsymbol{C} = \boldsymbol{F}^T \cdot \boldsymbol{F}$ and the determinant of the deformation tensor given the Jacobian $J = \det(\boldsymbol{F})$.

The resulting momentum Equation gives (in 3D) 3 equations, but has 4 unknowns, namely displacements in all dimensions and the fluid pressure. The additional Equation to solve this incomplete system comes from the mass balance over the whole poroelastic domain. The Equation for the mass balance can be found by taking the material time derivative

$$\frac{\partial \rho^\alpha}{\partial t} + \nabla \cdot (\vec{v}^\alpha \rho^\alpha) = c^\alpha.$$

Substituting $\rho^\alpha = n^\alpha \rho_*^\alpha$, assuming that the there will be no production of solid material in the system ($c^\alpha = 0$) and incompressibility of the solid phase $\left(\frac{\partial \rho_*^s}{\partial t} = 0\right)$ the following equations for the whole system are derived[5]

$$\rho_*^s \frac{\partial n^s}{\partial t} + n^s \rho_*^s \nabla \cdot \vec{v}^s = 0 \quad 3.5$$

$$\rho_*^f \frac{\partial n^f}{\partial t} + n^f \frac{\partial \rho_*^f}{\partial t} + n^f \rho_*^f \nabla \cdot \vec{v}^f = 0 \quad 3.6$$

A 1st order approximation of the state Equation for the fluid phase given by Fernandez [14]. The result

$$\frac{\partial \rho_*^f}{\partial t} = \frac{1}{K^f} \rho_{*_0}^f \frac{\partial p}{\partial t},$$

will be substituted in Equation 3.6. The constant K^f specifies the Bulk modulus of the fluid. Furthermore we can make the assumption that $\rho_{*_0}^f \approx \rho_*^f$ [15] and using this in combination with the previous assumption that $n^s + n^f = 1$, reduces the mass balance Equation to

$$\frac{n^f}{K^f} \frac{\partial p}{\partial t} + \nabla \cdot \vec{v}^s - \nabla \cdot \left(\frac{\kappa}{\mu} (\nabla p) \right) = 0. \quad 3.7$$

Here κ is the permeability of the fluid and μ is the viscosity. This equations expresses the physical relation that the flux of solid material ($\nabla \cdot \vec{v}^s$) and the flux of fluid (due to pressure differences) $\left(\nabla \cdot \left(\frac{\kappa}{\mu} (\nabla p) \right) \right)$ should be in balance with the change in volume. This relation is known as Darcy's law.

This completes the system of equations for the mechanical part of the model. Only the initial and boundary conditions must be prescribed. The initial conditions, defined for $t = t_0$ are

$$\vec{u}^s|_{t=t_0} = \vec{u}_0^s(\vec{x}),$$

$$p|_{t=t_0} = p_0(\vec{x}).$$

The boundary conditions consist of an applied displacement on a part of the boundary and a boundary condition defining the fluid influx

$$\vec{u}^s = \vec{g}^s(\vec{x}) \; on \; \Gamma_u,$$

$$-\rho_*^f \frac{\kappa}{\mu} (\nabla p)^T \cdot \vec{n} = q^f(\vec{x}) \; on \; \Gamma_f^q.$$

This completes the mechanical model.

3.1.2 The Biophysical Stimulus

From the mechanical model a mechano-stimulus is transferred to the biological model. This biophysical stimulus will determine the tissue evolution in the system. The stimulus combines the maximal shear strain (γ) and the interstitial fluid velocity (v) [16] as follows

[5] It should be noted that n in this case represents the constituent mass fraction and is not the element outward normal

$$S = \frac{\gamma}{a} + \frac{\nu}{b},$$

where a = 0.0375 ,b $= 3\ {}^{\mu m}/_{s}$ and the limits of S are $S_{max} = 3, S_{min} = 1$ [5].

The biophysical stimulus will determine what cells and material are being produced or destroyed. I shall discuss this aspect in more detail in the next section.

The maximum shear strain can be obtained by using the strain tensor and finding its principal values[6]. The relative fluid velocity can be calculated using Darcy's Law

$$v^f = \left\|\vec{v}^f\right\| = \|\kappa_{ii}\nabla p\| \quad \text{for i=1..3},$$

which is dependent on the fluid pressure.

3.1.3 The Biological Model

Using the biophysical stimulus as an input, the biological model determines the tissue differentiation. This differentiation is modeled by the cells dynamics that result in tissue formation. The cavities are initially filled with granulation tissue. The penetration of mesenchymal stem cells from the bone into the granulation tissue is modeled as a diffusion process. The mesenchymal cells may, under the influence of the stimulus and depending on various concentrations of cells or tissues, differentiate and proliferate into either fibroblasts, chondrocytes or osteoblasts which in turn produce respectively fibrous tissue, cartilage or bone (see Figure 9).

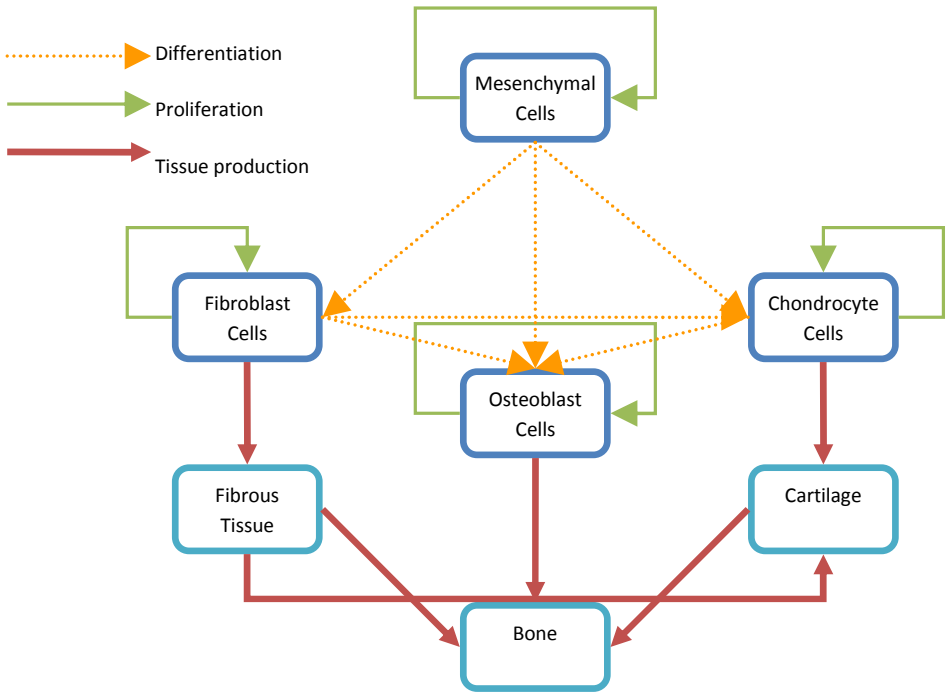


Figure 9 Schematic representation of the tissue differentiation paths as defined by the Prendergast tissue differentiation model.

[6] Note that $\gamma_{ij} = \epsilon_{ij} + \epsilon_{ji} = 2\epsilon_{ij}$, thus for the implementation the maximum shear strain should be divided by 2.

In Figure 9 each box is represents the governing differential Equation that drives the evolution of that specific type of cell or tissue. In total there are 7 governing equations, 4 control the evolution of cell concentrations and 3 control the evolution of tissues. These differential Equation that control the cell concentrations are

$$\frac{dc_m}{dt} = D_m\nabla^2 c_m + P_m(1-c_{tot})c_m - F_f\left(1-c_f\right)c_m - F_c(1-c_c)c_m - F_b(1-c_b)c_m, \quad 3.8$$

$$\frac{dc_f}{dt} = D_f\nabla^2 c_f + P_f(1-c_{tot})c_f + F_f\left(1-c_f\right)c_m - F_c(1-c_c)c_f - F_b(1-c_b)c_f, \quad 3.9$$

$$\frac{dc_c}{dt} = P_c(1-c_{tot})c_c + F_c(1-c_c)\left(c_m+c_f\right) - F_b(1-c_b)c_c, \quad 3.10$$

$$\frac{dc_b}{dt} = P_b(1-c_{tot})c_b + F_b(1-c_b)\left(c_m+c_f+c_c\right). \quad 3.11$$

The 3 equations that control tissue production are

$$\frac{dm_b}{dt} = Q_b(1-m_b)c_b, \quad 3.12$$

$$\frac{dm_c}{dt} = Q_c(1-m_b-m_c)c_c - D_b c_b m_c m_{tot}, \quad 3.13$$

$$\frac{dm_f}{dt} = Q_f\left(1-m_b-m_c-m_f\right)c_f - (D_b c_b + D_c c_c)m_f m_{tot}. \quad 3.14$$

These equations need to be programmed in a numerical method due to the diffusion terms $(D_\alpha\nabla^2 c_\alpha)$ that control the diffusion of mesenchymal cells and fibroblasts. The problem is further complicated by the non-linear terms that are present in these equations. Each tissue evolution process (indicated in Figure 9 by the arrows) is controlled by a coefficient. These coefficients are in turn dependent on the biophysical stimulus. The various coefficients are clarified in the following sections.

3.1.4 Diffusion coefficients (D_i)

The two diffusion coefficients are dependent on the concentration of cartilage and bone, as expressed by

$$D_m = D_{m0}(1-m_c-m_b),$$

$$D_f = D_{f0}(1-m_c-m_b).$$

This formulation implies that the diffusion takes place homogeneously along all parts of the boundary. This assumption is acceptable for small problems, but when may not be completely correct on a larger scale.

In these equations, the constants D_{m0} and D_{f0} are $D_{m0} = 240\,\frac{\mu m^2}{min} = 0.3456\,\frac{mm^2}{day}$ and $D_{f0} = 60\,\frac{\mu m^2}{min} = 0.1152\,\frac{mm^2}{day}$.

3.1.5 Cell Proliferation coefficients (P_i), Cell Differentiation coefficients (F_i), Tissue production coefficients (Q_i) and Tissue Degradation coefficients (D_i)

The proliferation can be modeled similarly as

$$P_m = P_{m0}(1 - m_c - m_b),$$

$$P_f = P_{f0}(1 - m_c - m_b),$$

$$P_c = P_{c0}(1 - m_c - m_b),$$

$$P_b = P_{b0}(1 - m_c - m_b).$$

The initial coefficients (with subscript 0) are piecewise-linear functions of the biophysical stimulus.

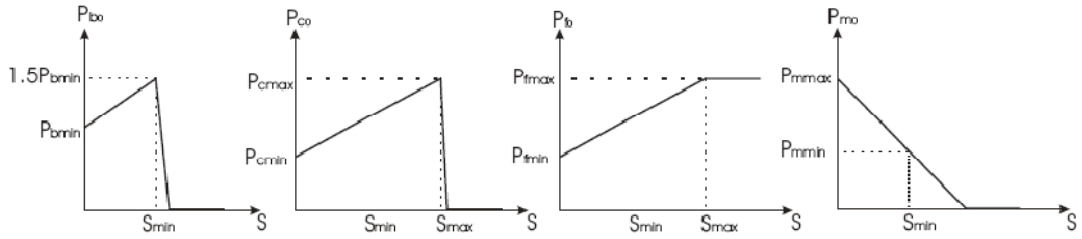


Figure 10 Variation of proliferation coefficients with varying biophysical stimulus [1]

The differentiation coefficients F_b, F_c, F_f are not dependent on any mass densities but only on the biophysical stimulus.

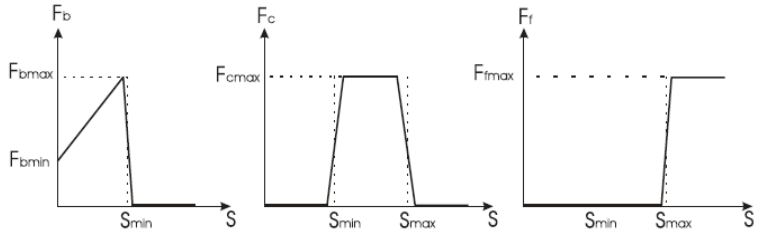


Figure 11 Variation of differentiation coefficients with varying biophysical stimulus [1]

The dependence of the tissue production coefficients is similar to the cell differentiation and tissue degradation is equal to its corresponding production coefficient.

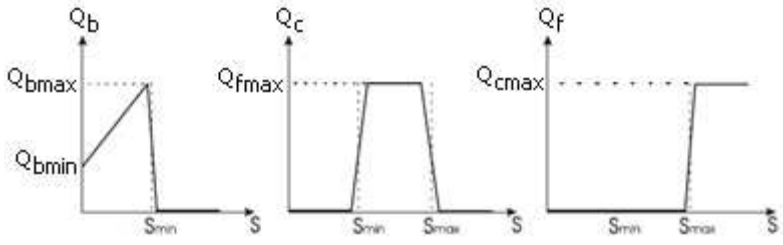


Figure 12 Variation of production coefficients with varying biophysical stimulus [1]

The parameters in the graphs are obtained from previous literature as well as a calibration (see chapter 5 of the thesis of A. Andreykiv)

Acquired values	Calibrated values
$P_{bmin} = 0.5/\,day$	$F_{bmax} = 0.15/\,day$
$P_{cmin} = 0.75/\,day$	$F_{bmin} = 0.005/\,day$
$P_{cmax} = 0.925/\,day$	$F_{cmax} = 0.3/\,day$
$P_{fmin} = 0.6/\,day$	$F_{fmax} = 0.01/\,day$
$P_{fmax} = 0.1/\,day$	$Q_{bmax} = 0.1/\,day$
$P_{mmin} = 0.5/\,day$	$Q_{cmax} = 0.2/\,day$
$P_{mmax} = 1.2/\,day$	$Q_{fmax} = 0.06/\,day$

3.1.6 Loading

The loading of the original model is chosen is accordance with animal experiments by Simmons and Pilliar [17]. The experiments were performed on a canine mandible to investigate the effect of implant surface geometry on the bone formation around dental implants. The growth of the bone around the implants was tracked for 28 days. The loading in the first week was a displacement and during the consequent 3 weeks, the average force recorded in the first week was applied. The results of this study were that displacements up to 50µm led to bone growth into the porous implant surface that was covered with sintered spheres. This loading of 50µm is used in the investigations in this thesis.

3.2 Numerical implementation

3.2.1 Mechanical Model

After discretization, the momentum Equation (Equation 3.3), together with the mass balance (Equation 3.7) are implemented as the system of equations

$$\mathbf{P}_{\mathrm{I}} - \mathbf{Q}\vec{\mathrm{p}} - \vec{\mathrm{f}}^{\mathrm{u}} = 0,$$

$$\mathbf{S}\frac{\partial \vec{\mathrm{p}}}{\partial \mathrm{t}} + \mathbf{Q}^{\mathrm{T}}\frac{\partial \vec{u}^{s}}{\partial t} + \mathbf{H}\vec{\mathrm{p}} - \vec{\mathrm{f}}^{\mathrm{p}} = 0.$$

In this system, the following matrices are identified

$\mathbf{P}_{\mathrm{I}}$ - The internal force vector for the solid phase

$\mathbf{Q}$ - The coupling matrix

$\vec{\mathrm{f}}^{\mathrm{u}}$ - The vector of traction forces

$\mathbf{S}$ - The compressibility matrix

$\mathbf{H}$ - The permeability matrix

$\vec{\mathrm{f}}^{\mathrm{p}}$ - The applied fluid mass influx vector

This system is solved in an iterative manner, according to

$$\begin{bmatrix} \mathbf{K}_{\mathrm{T}} & -\mathbf{Q} \\ -\mathbf{Q}^{\mathrm{T}} & -(\mathbf{S} + \Delta \mathrm{t}\mathbf{H}) \end{bmatrix}_{k,n+1} \begin{bmatrix} \Delta\vec{\mathrm{u}}^{s} \\ \Delta\vec{\mathrm{p}} \end{bmatrix}_{k+1} = \begin{bmatrix} \mathbf{f}^{\mathrm{u}}_{\mathrm{n+1}} \\ -\Delta \mathrm{t}\mathbf{f}^{\mathrm{p}}_{\mathrm{n+1}} \end{bmatrix} - \begin{bmatrix} \mathbf{P}_{\mathrm{I}_{\mathrm{n+1}}} - \mathbf{Q}\vec{\mathrm{p}}_{\mathrm{n+1}} \\ -(\mathbf{S}\Delta\vec{\mathrm{p}} + \mathbf{Q}^{\mathrm{T}}\Delta\vec{\mathrm{u}}^{\mathrm{s}} + \Delta \mathrm{t}\mathbf{H}\vec{\mathrm{p}}_{\mathrm{n+1}}) \end{bmatrix},$$

where $\boldsymbol{K}_T$ is the structural tangent matrix and is described by

$$\boldsymbol{K}_{T_{IJ}} = \int_V \boldsymbol{B}^T \boldsymbol{C}^{\sigma\tau} \boldsymbol{B} dV + \boldsymbol{I} \int_V \boldsymbol{\beta}_I^T \boldsymbol{\sigma}_E^s \boldsymbol{\beta}_J dV,$$

which is the division of geometric and material contributions to the stiffness matrix.

3.2.2 Biological Model

Of the 7 equations that describe the tissue evolution, 5 can be determined on element level, because all quantities are known. Only equations 3.8 and 3.9 need to be calculated using the finite element method because of the spatial dependency in the diffusion term.

This non-linear system is discretized and linearities to give

$$\begin{bmatrix} \boldsymbol{K}_{m_{stiff}} & 0 \\ 0 & \boldsymbol{K}_{f_{stiff}} \end{bmatrix} \begin{bmatrix} \Delta\vec{c}_{m_{n+1}} \\ \Delta\vec{c}_{f_{n+1}} \end{bmatrix} = \begin{bmatrix} -\vec{F}_{I_m} \\ -\vec{F}_{I_f} \end{bmatrix}.$$

Where the given matrices are defined as:

$$\boldsymbol{K}_{m_{stiff}} = \frac{\partial \vec{R}_m}{\partial \vec{c}_{m_{n+1}}}$$

$$\boldsymbol{K}_{f_{stiff}} = \frac{\partial \vec{R}_f}{\partial \vec{c}_{f_{n+1}}}$$

$$\vec{F}_{I_m} = \vec{R}_m$$

$$\vec{F}_{I_f} = \vec{R}_f$$

And $\vec{R}_m$ and $\vec{R}_f$ are described by

$$\boldsymbol{R}_m = \int_V \vec{N}^T \vec{N} \frac{\vec{c}_{m_{n+1}} - \vec{c}_{m_n}}{\Delta t} dV + \int_V \nabla\vec{N}^T \left[D_m \nabla\vec{N}\vec{c}_{m_{n+1}} \right] dV +$$

$$+ \left(P_m \left(1 - c_{c_n} - c_{b_n}\right) - F_f - F_c \left(1 - c_{c_n}\right) - F_b \left(1 - c_{b_n}\right) \right) \int_V \vec{N}^T \vec{N} \vec{c}_{m_{n+1}} dV +$$

$$+ \left(P_m - F_f\right) \int_V \vec{N}^T \vec{N} \vec{c}_{f_n} \vec{N} \vec{c}_{m_{n+1}} dV + P_m \int_V \vec{N}^T \left(\vec{N}\vec{c}_{m_{n+1}}\right)^2 dV$$

$$\boldsymbol{R}_f = \int_V \vec{N}^T \vec{N} \frac{\vec{c}_{f_{n+1}} - \vec{c}_{f_n}}{\Delta t} dV + \int_V \nabla \vec{N}^T \left[D_f \nabla \vec{N} \vec{c}_{f_{n+1}} \right] dV +$$

$$+ \left(P_f \left(1 - c_{c_n} - c_{b_n} \right) - F_c \left(1 - c_{c_n} \right) - F_b \left(1 - c_{b_n} \right) \right) \int_V \vec{N}^T \vec{N} \vec{c}_{f_{n+1}} dV +$$

$$+ \left(P_f + F_f \right) \int_V \vec{N}^T \vec{N} \vec{c}_{f_{n+1}} \vec{N} \vec{c}_{m_n} dV + P_f \int_V \vec{N}^T \left(\vec{N} \vec{c}_{f_{n+1}} \right)^2 dV$$

The remaining PDE's

The remaining PDE's need not be solved using a finite element approach. Take for example Equation 3.10. Applying a discretization and organizing the terms results in

$$\frac{dc_c}{dt} = P_c (1 - c_{tot}) c_c + F_c (1 - c_c) \left(c_m + c_f \right) - F_b (1 - c_b) c_c = 0$$

$$P_c \Delta t c_{c_{n+1}}^2 + (1 - \left(P_c \left(1 - c_{m_n} - c_{f_n} - c_{b_n} \right) + F_c \left(c_{m_n} + c_{f_n} \right) + F_b \left(1 - c_{b_n} \right) \right) \Delta t c_{c_{n+1}} + \left(-F_c \left(c_{m_n} + c_{f_n} \right) \Delta t - c_{c_n} \right) = 0$$

And this simple quadratic function can easily be solved for $c_{c_{n+1}}$. This procedure will be repeated for the other PDE's

3.3 Implementation into MSC Marc

The bone ingrowth model has been reformulated to a finite element implementation. The next phase is to implement it in a commercial code. The following sections will elaborate on the implementation within MSC Marc. The overall simulation setup is given in

Again the model is split in a mechanical and a biological model. Each model is programmed in a separate model. The two models run sequentially and communicate by writing out information that is read by the next model. Both models have been implemented in a user subroutine that is called by MSC Marc.

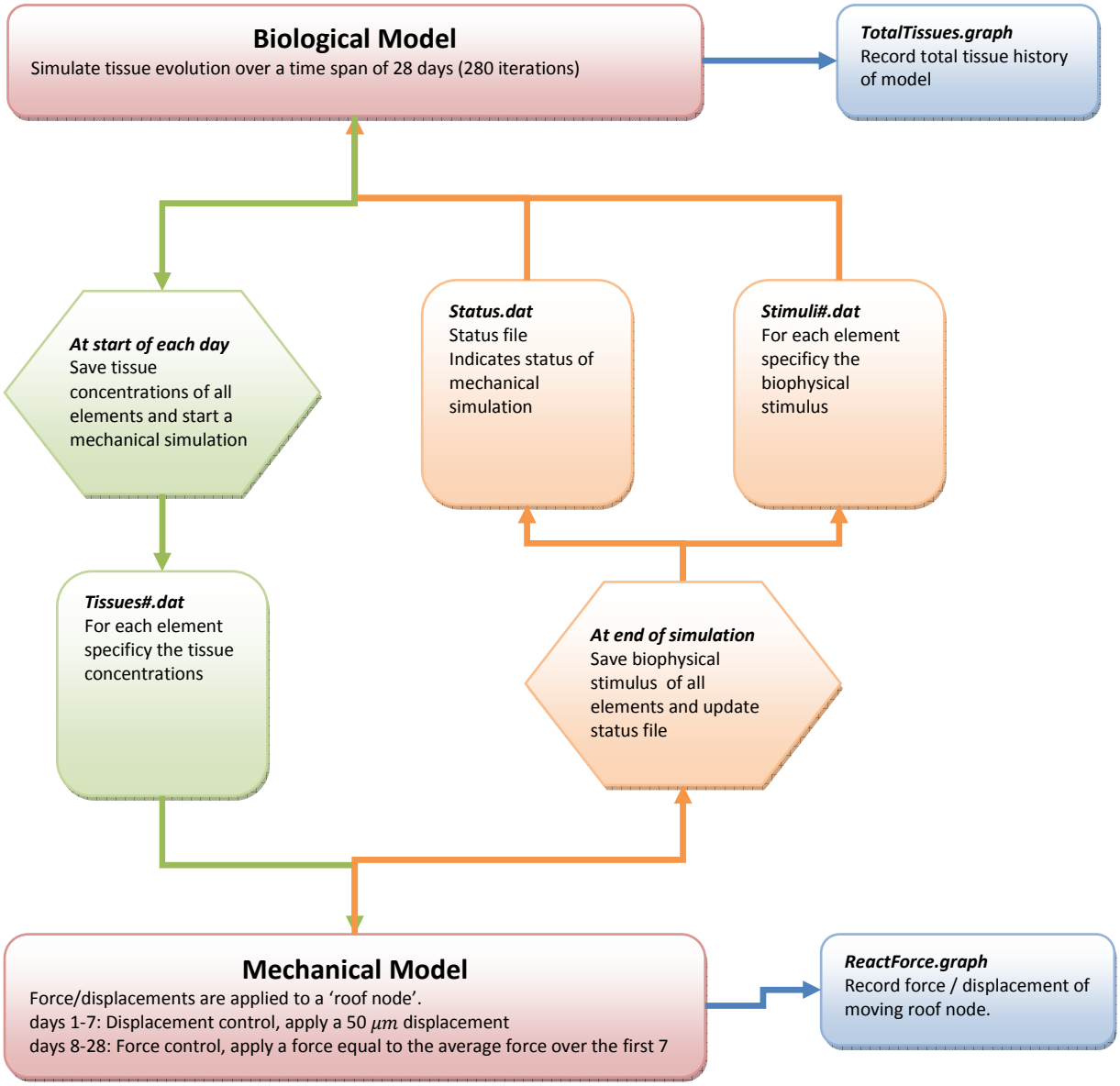


Figure 13 **Simulation setup and communication between biological and mechanical model**

The simulation is controlled by the biological model. The biological model calculates the tissue evolution over a period of 28 days. At the start of every day a mechanical simulation is started.

The mechanical model is loaded with an applied displacement or force, depending on time in the simulation. The displacement or force is applied to a 'roof node' that links to all surface nodes that would normally be in contact with the implant. The 'roof node' ensures a uniform loading along the surface of the implant. When the mechanical model is finished the resulting biophysical stimulus is written out and a status file is updated such that the biological model knows that it can continue.

The total tissue concentrations are recorded over the 28 days, as well as the displacement and force in the roof node.

3.3.1 Mechanical Model

The following flow diagram illustrates how the mechanical model is implemented in MSC Marc.

The subroutine specifies actions to be taken at specific moments in the simulation. Some are initiated at the start of the simulation, some at the start of a time step and other are called on element level. Figure 14 shows the flow diagram for the mechanical simulation, the blue section is called on element level.

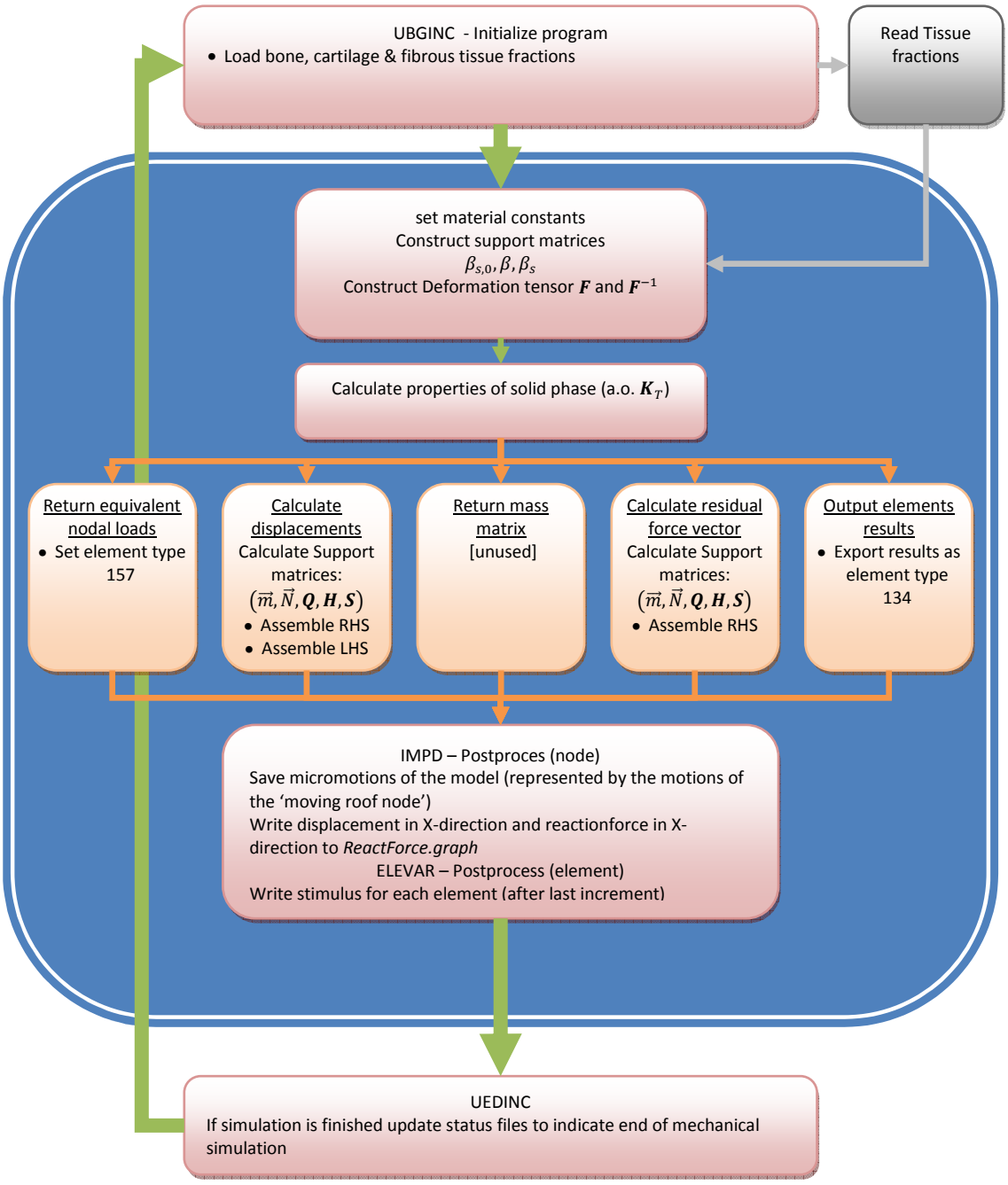


Figure 14 Flow chart of the mechanical model in MSC Marc

3.3.2 Biological Model

The implementation of the biological model is also done in the user subroutines. The scheme of the subroutine is shown in Figure 15.

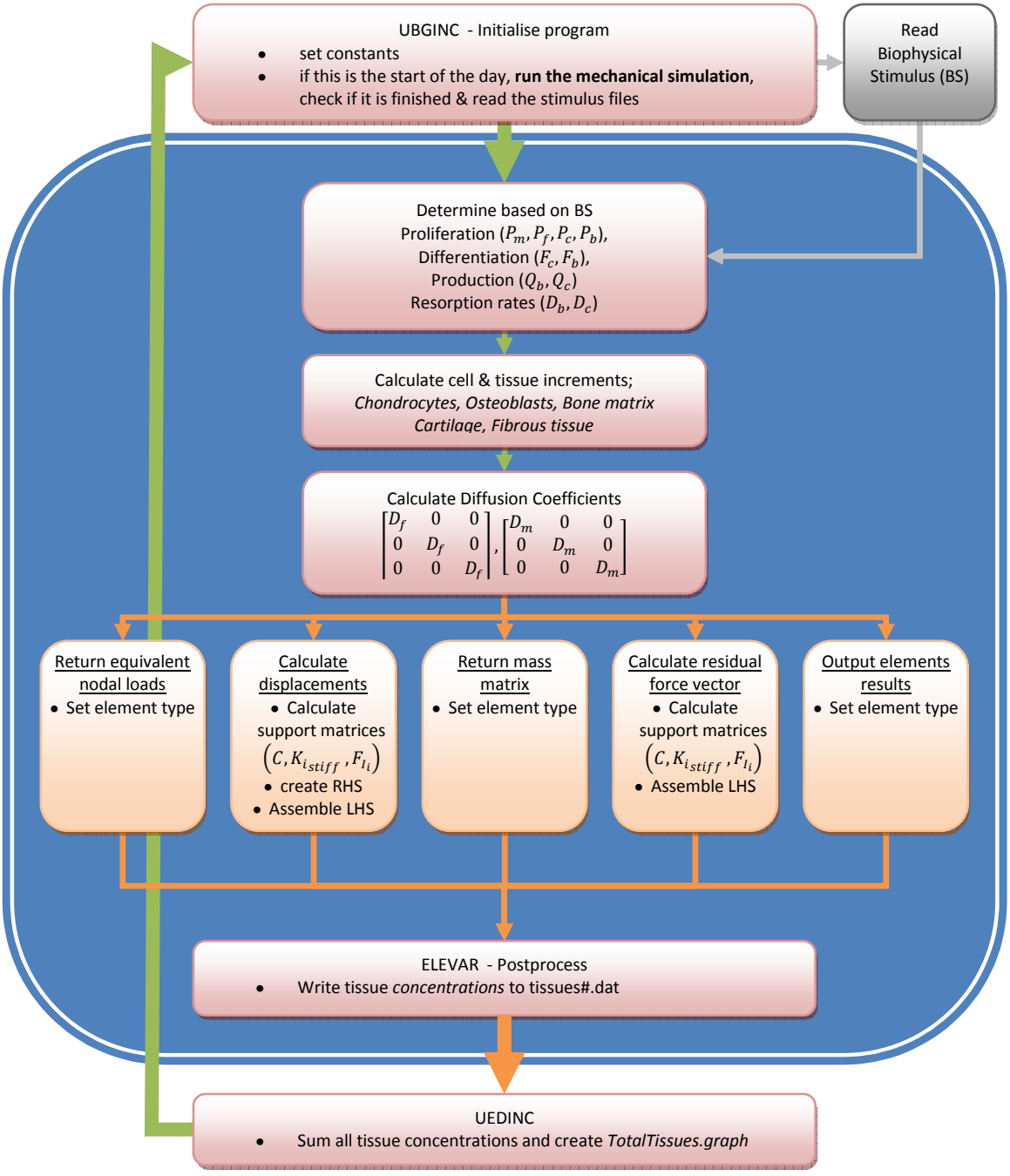


Figure 15 Flow chart of the biological model in MSC Marc

4 Model optimization opportunities

Based on the theory and the implementation of the bone ingrowth model, 3 possible opportunities are identified that may enable an increased speed of the simulation. They are model optimizations, code optimizations and the application of computational homogenization.

4.1 Model optimizations

4.1.1 Mechanical Model – Elimination of the fluid phase

Looking the theoretical implementation of the model has shown some possibilities for improving performance. The mechanical model is a biphasic model. The biophysical stimulus is constructed using inputs from the solid phase (maximum shear strain) and the fluid phase (fluid flow), see Figure 16.

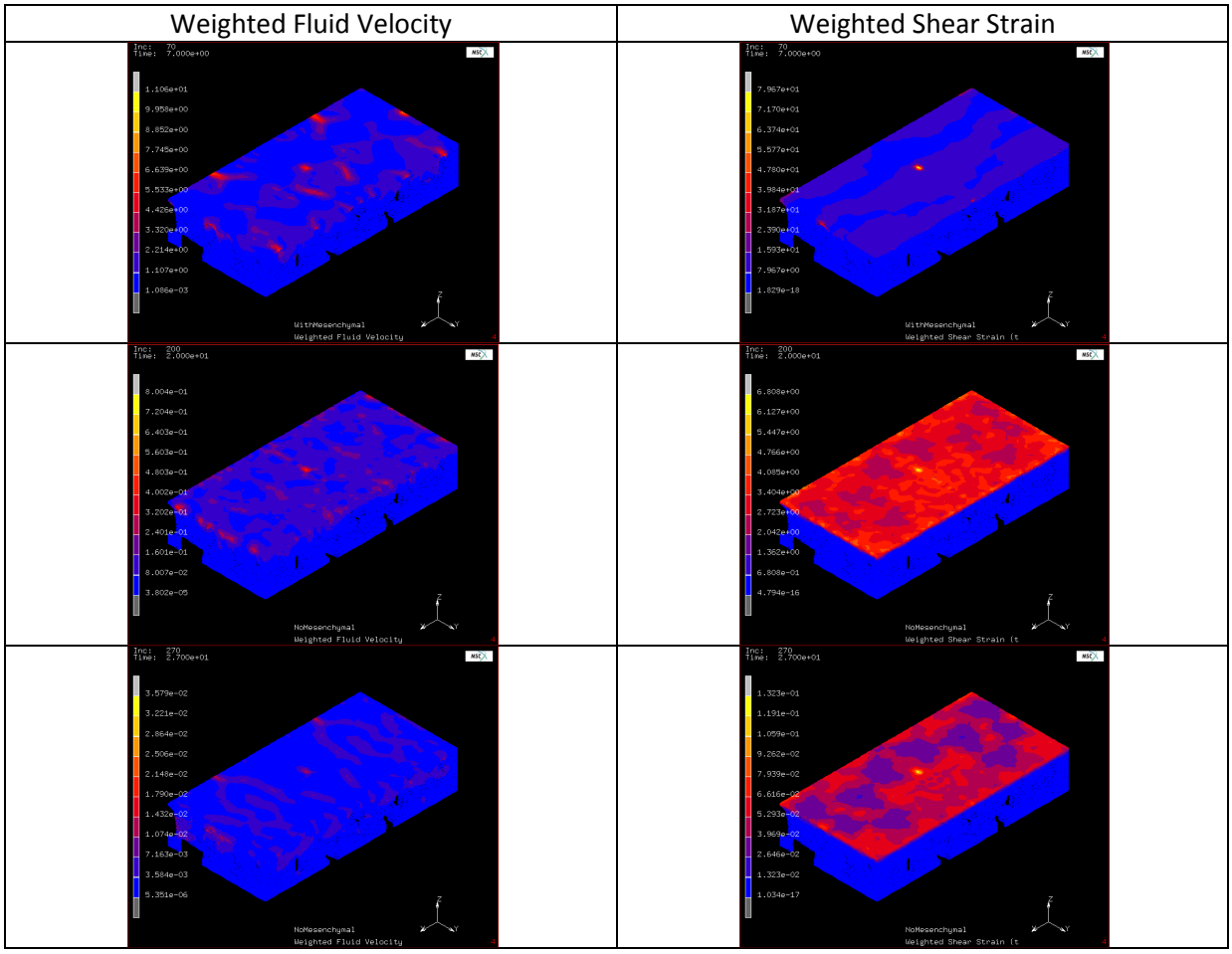


Figure 16 Components of the biophysical stimulus for a selected number of days. The ratios in which the two components contribute to the total biophysical stimulus appears to be relatively consistent throughout the complete simulation.

Investigation of the components of the biophysical stimulus shows that the contribution of the shear strain is generally almost 1 order of magnitude larger than the contribution of the fluid velocity. This might indicate that the fluid phase is of lesser importance, even though

The biological tissue is considered to consist of 80% fluid [1]. If the fluid phase may be ignored, the implementation of the mechanical model can be greatly simplified and a large gain in computational speed could be achieved.

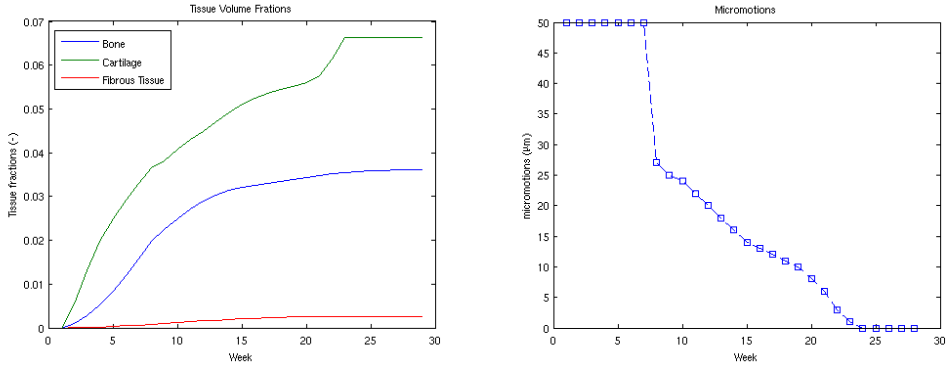


Figure 17 **Left: Total tissue fractions of the complete model. Right: Micromotions measured in the roof node.**

It is imperative that the overall results are not severely altered through such a change. One measure for the quality of the simulation is the total tissue concentration and the micromotions of the system (see Figure 17).

4.1.2 Biological Model – Linear approximation

The non-linear nature of the biological model causes the largest decrease in computational speed. Examing and rewriting Equations 3.8 and 3.9, the effects of this non-linearity can be exposed.

$$\frac{dc_m}{dt} = D_m\nabla^2 c_m + \Big(P_m(1-c_c-c_b) - F_f - F_c(1-c_c) + F_b(1-c_b)\Big)\,c_m + \big(F_f - P_m\big)c_f c_m - P_m c_m c_m$$

$$\frac{dc_f}{dt} = D_f\nabla^2 c_f + \Big(P_f(1-c_c-c_b) - F_c(1-c_c) - F_b(1-c_b)\Big)\,c_f + F_f c_m - \big(F_f + P_f\big)c_m c_f - P_f c_f c_f$$

In red are the terms causing the non-linear behavior. An investigation will be made what the effects are of removing the higher order terms.

4.1.3 Biological Model – Diffusion approximation

Another possibility for optimization lies in the diffusion term. Diffusion currently takes place in all directions. Given the geometry of the model, as well as the uniform seeding of mesenchymal cells, the diffusion could be approximated using a unidirectional diffusion.

The consideration of such a 1-dimentional is further strengthened by the observation that mesenchymal and fibroblast cells are quickly uniformly distributed throughout the model (see Figure 18).

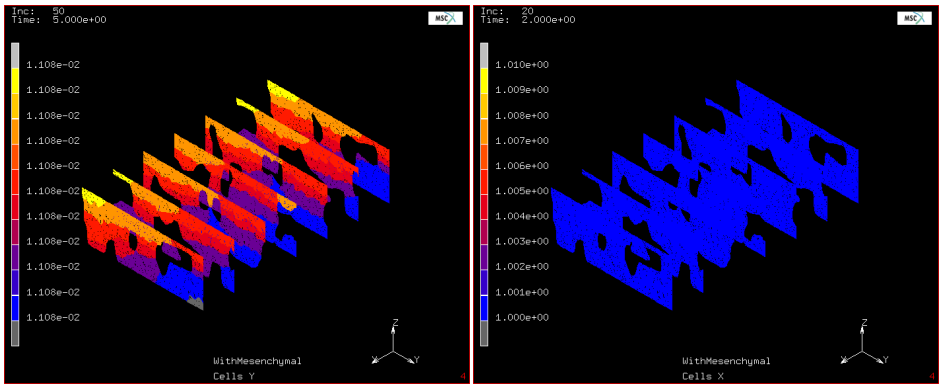


Figure 18 Fibroblast (left) and mesenchymal (right) concentration distribution in day 5

The mesenchymal cells are completely uniformly distributed and although the fibroblast concentrations appear to have a gradient, examining the scale shows that this gradient is minimal.

Unfortunately, the implementation reveals that only very little efficiency will be gained. The diffusion is controlled through the integral

$$\boldsymbol{K} = \int_{V_{el}} \nabla \boldsymbol{N}^T \boldsymbol{D}\, \nabla \boldsymbol{N} dV_{el},$$

resulting in a matrix with dimensions 4x4 (for a 4-node element). 2 matrix multiplications will be performed to construct $\boldsymbol{K}$, in which a 4x3 is multiplied with a 3x3 matrix and secondly, the resulting 4x3 matrix is multiplied with a 3x4 matrix to construct to final 4x4 matrix. A unidirectional approach will still require these same multiplications, except the input matrix containing the gradient of the shape functions ($\nabla \boldsymbol{N}$) is a 4x1 matrix, rather than a 4x3 matrix.

4.2 Code Optimizations

4.2.1 Sleep

In order to share data between the biological and the mechanical simulation, data is written to files on disk. One of the issues is that simultaneous file operations are not permitted by the system and thus these situations must be avoided. A solution is to attempt a file operation and assess the response of the computer system. If a file is in use, the simulation must wait until the file is free for use. The aforementioned method is employed at several stages in the subroutine, as is it used when reading and when writing to files. The following example is a loop in the biological simulation (see Figure 19), where the subroutine has to read all the biophysical stimuli for each element.

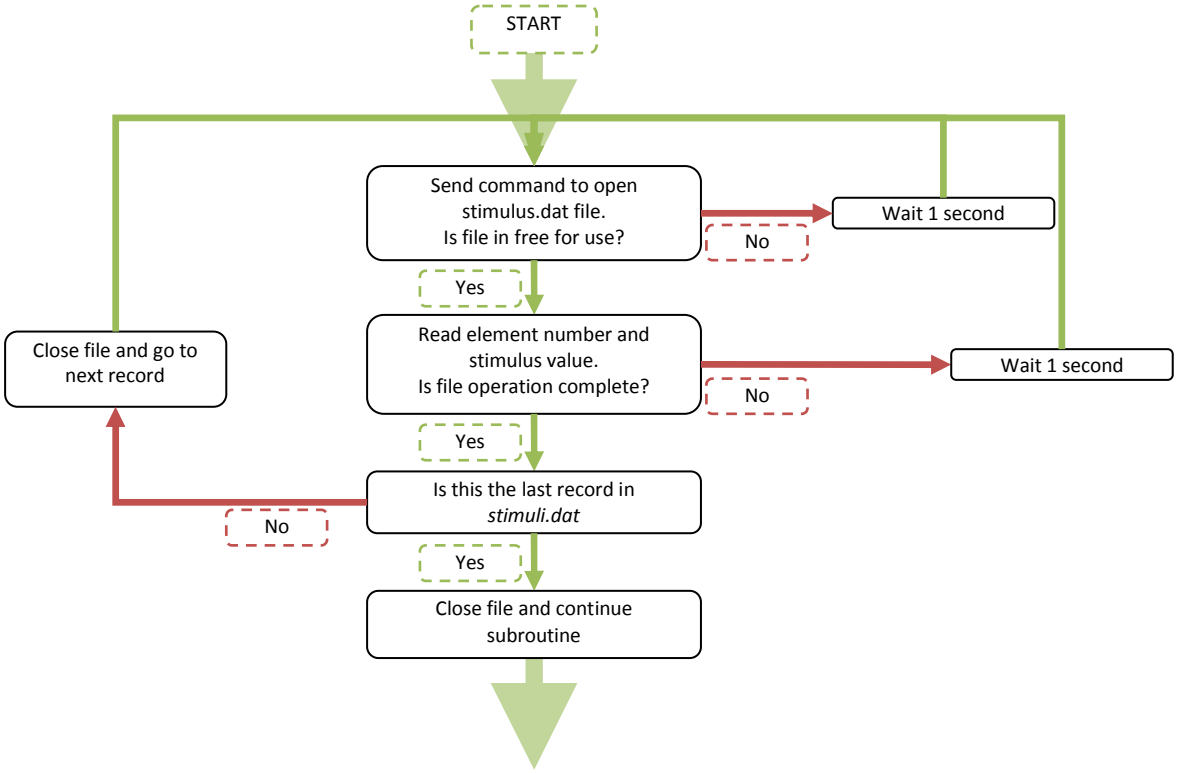


Figure 19 Flowchart depicting the steps when the Stimuli#.dat files are being read

A 1 second waiting time is invoked if the file is busy and a disk operation cannot be performed. Note that this loop will run for all elements, thus greatly increasing the waiting time. Of course this wait is not called for at every instance but reducing the waiting period should save a lot of time.

Similar comparable loops are present in the code and reducing the waiting times will greatly affect the simulation time.

4.2.2 Writing sequence

Further investigating Figure 19 it can be seen that for a single element the complete stimuli.dat is opened and completely read. During this process the subroutine will continuously open and close the file, resulting in many disk operations with possible waiting times that accompany this process. If a better process can be designed, the number of disk operations and consequent waiting times can be further reduced.

4.3 Computational Homogenization

An attempt is made to code computational homogenization in MSC Marc. The application of computational homogenization will be done in a restricted setting, namely only the mechanical model.

One of the goals of this thesis was to assess the feasibility of the application of the computational homogenization scheme on the bone ingrowth model. As an initial step in this process, the computational homogenization scheme should then also be applicable to

one of its core features (i.e. the mechanical model). Since the computational homogenization scheme was developed with a purely mechanical setting in mind, the most logical initial step is to see how well the scheme can be applied to the mechanical aspects of the bone ingrowth model.

As a result of this restricted setting, a number of other parameters had to be fixed as well. The following settings are taken:

- Displacement control (an applied displacement of 50 μm) is used
- The tissue fractions are fixed at realistic values obtained during runs of the original model. The tissue fractions are:
  - Bone = 0.01394211
  - Cartilage = 0.01865109
  - Fibrous Tissue = 0.0008101763

The mechanical model supplies the biophysical stimulus to the biological model. This stimulus consists of the maximum shear strain and the fluid flow. Since the fluid phase is ignored in this setting, the current relevant parameter to be upscaled is the maximum shear strain.

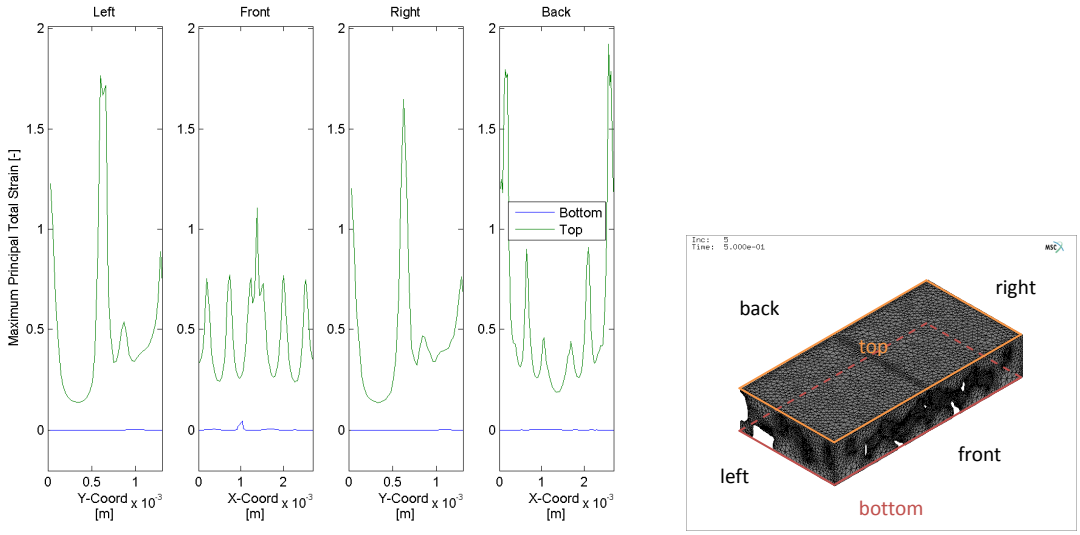


Figure 20 Left: Maximum principal strain along the Top & Bottom edges. Right: identification of the edges

The maximum principle strain does appear to show some oscillations that may be averaged out using a homogenization approach. However, it should be noted that the biophysical stimulus relies on the maximum shear strain. An averaging procedure, such as homogenization, may influence the results if these maxima are affected.

If the computational homogenization scheme can be implemented 2 main questions must be dealt with. How does the averaging procedure affect the maximum shear strain? Furthermore, as discussed in the literature survey [7], is the assumption of separation of scales applicable.

5 Results - Model optimizations

5.1 Mechanical Model – Elimination of the fluid phase

In order to assess the importance of the fluid phase a run was made that has the fluid phase removed (see Figure 21).

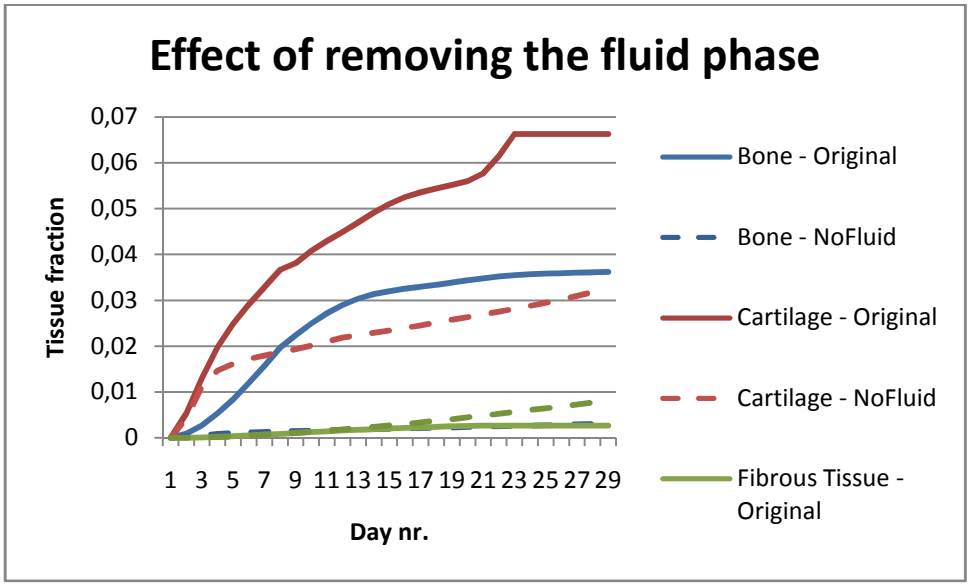


Figure 21 Tissue evolution with the biphasic model and a single constituent model

The importance of the fluid phase becomes apparent. Without the fluid phase a much lower amount of tissue is produced. In addition, the fractions in which they appear are also different. A much smaller fraction of the stiffer materials is being produced. A serious reduction in cartilage and a very large reduction in bone tissue are the result of the elimination of the fluid phase. The less stiff fibrous tissue is still calculated reasonably accurate without the fluid phase.

Regarding the simulation time, the single constituent model does greatly reduce run times

```
timing information:                        wall time      cpu time

total time for input:                        27.68         26.22
total time for stiffness assembly:         5711.05       5670.46
total time for stress recovery:            3648.77       3624.95
total time for matrix solution:           44341.22      43949.75
total time for restart:                      74.37         14.00
total time for output:                     1709.29       1627.41
total time for miscellaneous:             16639.61       2067.30
-------------------------------------------------------------------
total time:                               72151.98      56980.09
```

Clearly a single constituent model would greatly aid the computational speed, however it has been shown that the fluid phase is essential to the tissue evolution and cannot be ignored.

5.2 Biological Model – Linear approximation

Removing the higher order terms in the differential equations (Equations 3.8 and 3.9) results in

$$\frac{dc_m}{dt} = D_m \nabla^2 c_m + \left(P_m(1 - c_c - c_b) - F_f - F_c(1 - c_c) + F_b(1 - c_b)\right) c_m,$$

$$\frac{dc_f}{dt} = D_f \nabla^2 c_f + \left(P_f(1 - c_c - c_b) - F_c(1 - c_c) - F_b(1 - c_b)\right) c_f + F_f c_m.$$

The effect on the tissue differentiation scheme is depicted in Figure 21. The production of fibroblast cells will cease and as a consequence no more fibrous tissue will be produced. The input from the fibroblasts and the fibrous tissue will affect the other cell concentrations and tissue concentrations.

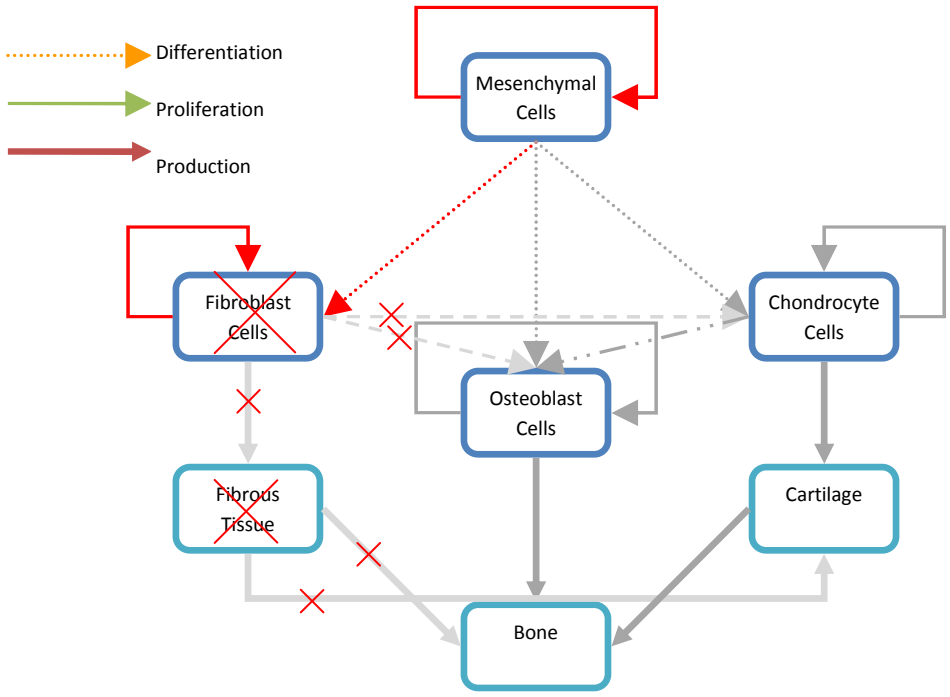


Figure 22 Alterations in the Tissue differentiation model due to the removal of non-linear terms.

Figure 23 shows this effect. No fibrous tissue is produced, but more bone and cartilage are produced.

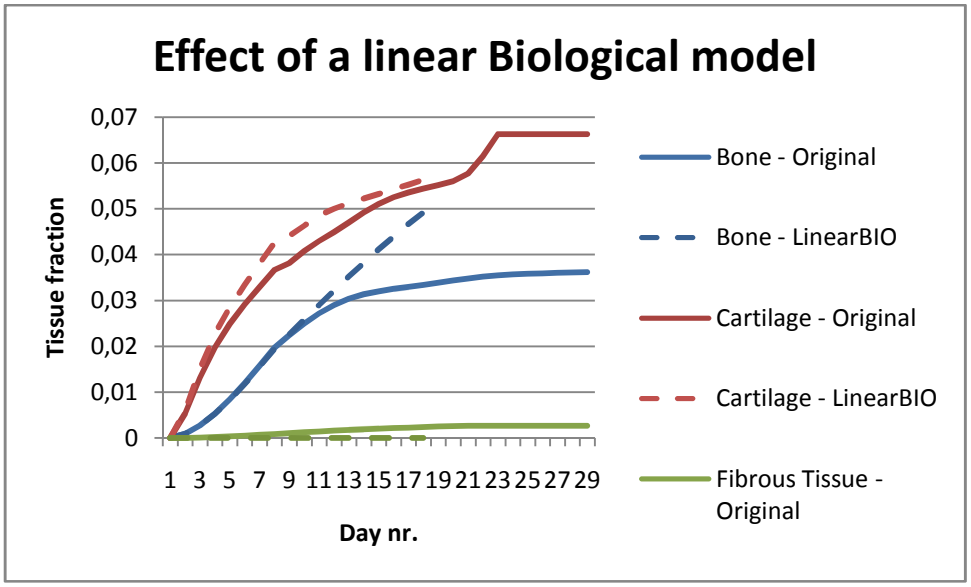


Figure 23 First 18 days of a simulation using a linear biological model.

This increased concentration of bone and cartilage results in an overestimate of the stiffness. Although cartilage is reasonably estimated by the linear model, the absence of fibrous tissue and the enormous overestimate of bone prohibits the removal of the non linear terms from Equations 3.8 and 3.9.

5.3 Biological Model – Diffusion approximation

Attempts to simplify the diffusion model to a 1 dimensional problem failed for unknown reasons. The 1D model was initially simply modeled by setting the diffusion coefficients in X and Y direction to zero, but computations in MSC Marc halted after 1 iteration with the error message that the stiffness matrix became non-positive definite. It could not be established what is the cause of this non-positive definiteness.

However, it is not expected that such a simplification will gain a lot of computational speed. The diffusion matrix is compiled by performing the calculation of

$$\boldsymbol{K} = \int_{V_{el}} \nabla \boldsymbol{N}^T \boldsymbol{D}\, \nabla \boldsymbol{N} dV_{el},$$

which in requires 2 matrix multiplactions in the subroutine. Using the 1 dimensional approximation, this can be reduced to a single matrix multiplication and the multiplication with a constant. A comparison in Matlab was made between the average computation time required for both sets of computations. Given the nr of elements and the number of iterations in the biological model, the gain was on average only 200 seconds. This is marginal compared to the total run time of the complete simulation.

6 Results - Code Optimizations

An analysis of the programming was made and several options for improvement have been identified in section 4.2. The results of these code optimizations are given below. It will be shown that a large increase in efficiency is gained simply by optimizing disk writes. These changes do not affect the results of the simulations.

6.1 Minisleep

FORTRAN's intrinsic command 'SLEEP' only accepts integers and thus the smallest sleep is 1 second. To overcome this issue I have created a small subroutine called MINISLEEP. It simply calls the intrinsic function 'DATE_AND_TIME' that records the current time (up to milliseconds) and forces a dummy loop of a specified number of milliseconds.

```
C=====================================================
C    MINISLEEP small subroutine to sleep for nms millisec.
C=====================================================
      subroutine minisleep(nms)
      integer time1 (8), time2 (8), deltasleep
      CHARACTER (LEN = 12) tmp1 (3)
      CALL DATE_AND_TIME (tmp1 (1), tmp1 (2), tmp1 (3), time1)
50    CALL DATE_AND_TIME (tmp1 (1), tmp1 (2), tmp1 (3), time2)
       deltasleep = (time2(7)*1000.d0+time2(8))-(time1(7)*1000.d0+time1(8))
       if (deltasleep.LT.nms) then
         goto 50
       endif
       return
       end
```

The question now becomes what value to select for MINISLEEP. It might be possible that if a too small value is chosen, the time needed for the additional disk operations outweighs a smaller sleep time. Thus various values for MINISLEEP will be tested.

6.1.1 Comparison of minisleep values

Unmodified Simulation

The following is a copy of the end of the log file after a successful run.

```
timing information:                      wall time      cpu time

total time for input:                      26.89         25.15
total time for stiffness assembly:      10131.55       5471.87
total time for stress recovery:          4103.51       3559.33
total time for matrix solution:         42604.60      34629.30
total time for restart:                    53.64          5.63
total time for output:                  30050.08       2142.14
total time for miscellaneous:          112628.62       1208.83
-------------------------------------------------------------
total time:                            199598.89      47042.25
```

As can be seen, the total time is almost 250,000 seconds, which equals about 68 hours.

On average, a poroelastic simulation will take about 4400 seconds and a biological simulation (consisting of 10 increments) will take about 2700 seconds.

MINISLEEP – 1 ms

Incorporating a 1 millisecond sleep instead of the 1 second sleep should lead to a large reduction in wall time. CPU time should not be affected as the sleep function does not influence the calculations.

```
timing information:                      wall time      cpu time

total time for input:                      25.70         25.42
total time for stiffness assembly:       9455.66       5444.41
total time for stress recovery:          3619.36       3535.15
total time for matrix solution:         32467.87      31268.41
total time for restart:                    78.29          2.63
total time for output:                  18681.64       1968.34
total time for miscellaneous:           87025.32       1157.73
-------------------------------------------------------------
total time:                            151353.84      43402.10
```

As expected, there is a great reduction in wall time. Also the CPU time is reduced. Together the time saving adds up to about 21%, with a total computing time of 54 hours. Apparently the CPU time is also positively affected by the MINISLEEP function.

The average time spent on the poroelastic simulation has decreased to about 3400 seconds and the average time spent on the biological simulation has decreased to approximately 2000 seconds. Thus the poroelastic simulation saw a decrease of 22% and the biological simulation was reduced by 25%. It seems that both processes benefit equally from the reduced sleep time.

MINISLEEP – 10 ms

Increasing the sleep time from 1 millisecond to 10 milliseconds showed no effect.

```
timing information:                      wall time      cpu time

total time for input:                      25.78         25.37
total time for stiffness assembly:       9084.10       5438.27
total time for stress recovery:          3600.53       3496.23
total time for matrix solution:         32024.05      31258.48
total time for restart:                    47.42          2.82
total time for output:                  18815.55       1989.05
total time for miscellaneous:           86642.52       1169.95
-------------------------------------------------------------
total time:                            150239.95      43380.17
```

The numbers actually show a decrease in time, but this decrease accumulates to 0.5%. I believe that this falls within a range of variations that may occur from simulation to simulation and thus I cannot conclude that this decrease in time may be attributed to the longer sleep time.

MINISLEEP – 250 ms

The slight increase in sleep did not seem to have notable effects. Increasing the sleep to 250 milliseconds showed an increase in computing time again.

```
timing information:                         wall time      cpu time

total time for input:                         25.80         25.56
total time for stiffness assembly:         10021.93       5512.17
total time for stress recovery:             3634.99       3542.79
total time for matrix solution:            32105.49      31276.15
total time for restart:                       40.68          2.43
total time for output:                     23050.94       2005.11
total time for miscellaneous:             106662.51       1169.46
-------------------------------------------------------------------
total time:                               175542.34      43533.68
```

6.1.2 Results

It seems that increasing the sleep time slightly has no effect. Increasing the sleep time to 250 milliseconds had a detrimental effect. From this it can be concluded that an increased number of disk operations due to a decreased sleeping time is of minimal influence total computing time. Thus a shortest sleep time is preferred.

For comparison I have collected the logs and processed them to show the time at each iteration.

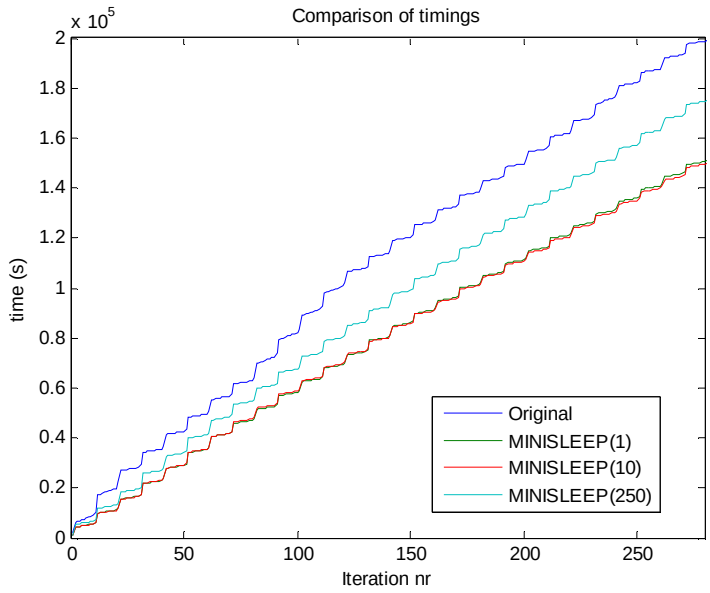


Figure 24 Influence of reduced sleep times (1, 10, 250 ms) compared to the original code

Figure 24 shows the impact of the MINISLEEP routine and plot the iteration number versus the wall time. The staggered graph clearly shows the initiation and end of the poroelastic and the biological simulations. Using the log files I was able to extract average computing times for the biological and mechanical simulation, they are summarized in Table 1.

	Biological simulation	Mechanical simulation
Original model	2707 s	4355 s
MINISLEEP(1)	1976 s	3407 s
MINISLEEP (10)	1978 s	3365 s
MINISLEEP (250)	2114 s	4115 s

Table 1 Average run times (seconds) for the biological and mechanical simulations

Based on these finding a 1 millisecond sleep time is selected. This routine will reduce the computation time by roughly 20%.

6.2 Batchwrites

Instead of writing almost continuously to the disk, the change was made to save all variables in memory and then write them to the disk only at the end of each iteration. This saves disk writes, but will also save disk read activity as the original loops had to read the complete array before being able to append the array.

The implementation of this system is described in Appendix D.

It should be noted that the implementation of the batch writes is in conjunction with the 1ms minisleep function. The effect on simulation speed is represented in the log file.

```
timing information:                      wall time      cpu time

total time for input:                      27.68         26.22
total time for stiffness assembly:       5711.05       5670.46
total time for stress recovery:          3648.77       3624.95
total time for matrix solution:         44341.22      43949.75
total time for restart:                    74.37         14.00
total time for output:                   1709.29       1627.41
total time for miscellaneous:           16639.61       2067.30
-------------------------------------------------------------
total time:                             72151.98      56980.09
```

Compared to the results of the 1 ms minisleep, the batch writes accomplish a reduction in computation time of more than 50% (compared to the model with a 1ms sleep time implemented), showing the effectiveness of minimizing disk activity. Compared to the original model a reduction in computation time of nearly 65% is realized.

7 Implementation and Results - Computational Homogenization

The theory of computational homogenization has been outlined in section 2.4. This chapter will perform a feasibility study into the implementation of computational homogenization in MSC Marc for the bone ingrowth model.

Section 4.3 showed that the original model shows possibilities that computational homogenization may be applied to the reduced model. For the feasibility study a geometrically simpler model is chosen. Simplifications to the model are discussed in section 7.1.

7.1 Model Simplifications

As outlined in section 4.3, several simplifications have been applied in order to better study the feasibility of implementing computational homogenization in MSC Marc. The applied simplifications are that only the solid part of the mechanical model is considered. It is loaded with a 50um displacement and has fixed material properties.

7.1.1 Geometry

The geometry of the original model may not lend itself well for homogenization. In fact it is already a volume that is chosen to be representative, thus already is an RVE. In order to test the computational homogenization scheme a simpler geometry has been selected.

The model is chosen to be a stacked series of beads. Beads are a common geometry in the coating of implants[7], however it would be more likely that they appear in a closest-stacking configuration. A simple stacked model would allow the straightforward RVE selection of a cube with a spherical void. This eased RVE selection is the reason that this geometry was selected.

The geometry of the reference model is shown in Figure 25.

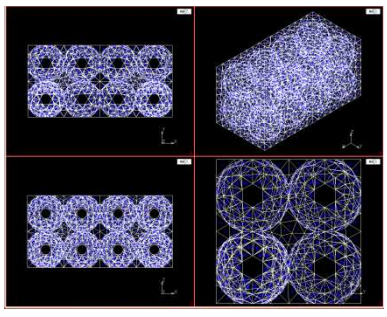


Figure 25 Geometry of the reference model. 8 beads in a stacked configuration.

The extracted RVE and the resulting macroscopic model are shown in Figure 26.

[7] See for example http://www.mcminncentre.co.uk/ or http://www.zimmer.co.uk

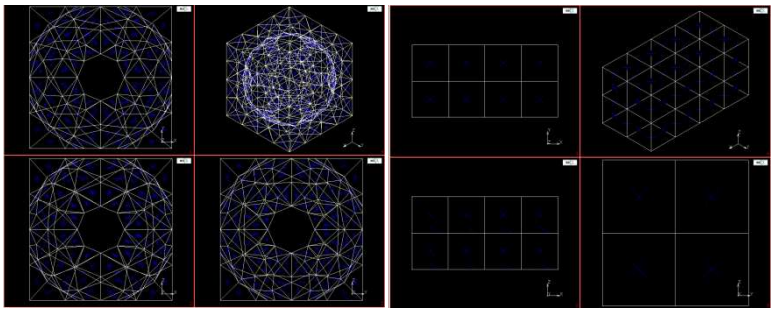


Figure 26 Left: RVE geometry. Right: Macroscopic model, consisting of 8 hexahedral elements.

Figure 26 shows the essence of homogenization. To the right in Figure 26, a macroscopic model is shown that aims to convey the structural response of a model like Figure 25.

7.1.2 Loading of the model

One of the restrictions that became apparent is the loading of the model. On the interface between bone and implant, it was assumed that the application of the load is done uniformly over the surface where the implant interfaces with the biological tissue. When a homogenized approach is selected, this is no longer possible since the geometry of the implant surface is assumed to be flat and the loading must applied on the top surface of the model. This may be a severe restriction on the applicability of homogenization. In order to investigate this, a short simulation was performed where the model (with the complete geometry) was loaded on the top surface, rather than the surface that is in assumed to be in contact with the implant.

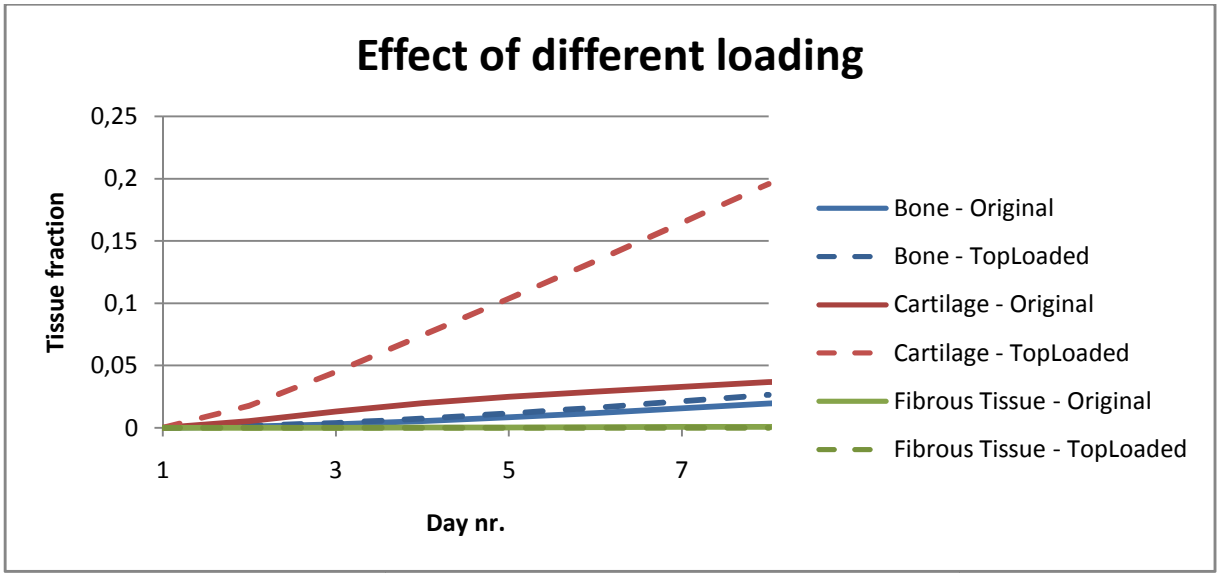


Figure 27 Effect of the same load applied to the top surface, rather than the implant surface.

Clearly different results appear as a consequence of the altered loading application.

7.2 Algorithm Implementation in MSC Marc

A homogenization approach consists of 2 models. 1 model is the macroscopic model, which aims to represent the global geometry. The second model is the RVE, which models a periodic or representative structure that is distributed throughout the macroscopic model.

The geometry of the RVE and the macroscopic model were shown in Figure 26. These sections will go into the implementation in MSC Marc and discuss the results.

The general structure of the computational homogenization scheme starts with the macroscopic model. The general loading is applied to macroscopic model. From this loading, the microscopic model will determine its structural response. This response (in the form of the stress tensor and the related stiffness matrix) are determined and taken to the macroscopic level.

7.2.1 Microscopic element

The RVE element forms the core of the computational scheme. The microscopic element uses the macroscopic deformation tensor to formulate a loading condition. The boundary conditions are periodic boundary conditions and are completed with 1 fixed node (u_{000}) and 3 nodes that are prescribed using the macroscopic deformation tensor (u_{001} , u_{100} , u_{010}).

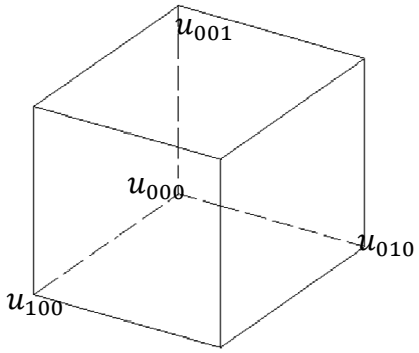


Figure 28 Corner node identification in the microscopic model

Periodic Boundary Conditions

The periodic boundary conditions are applied to all non-prescribed boundary nodes. They shall move in accordance with the corner nodes. The periodicity constraints are formulated as Equation 2.3, resulting in

$$u_{TOP} - u_{BOTTOM} = u_{001} - u_{000},$$

$$u_{RIGHT} - u_{LEFT} = u_{100} - u_{000},$$

$$u_{BACK} - u_{FRONT} = u_{010} - u_{000}.$$

These periodic boundary conditions are prescribed in MSC Marc/Mentat using servo links. They are applied using an automated python routine (see Appendix C). Care should be taken that some nodes do not obtain a double periodic constraint.

Prescribed Nodes

The deformation on the prescribed nodes (subscript p) can be calculated as the difference between the deformed and the undeformed configuration. The microscopic model is assumed to deform in accordance with the macroscopic model i.e. the macroscopic deformation tensor governs the deformation of the microscopic model. The displacement of the prescribed nodes can be described as the change between the undeformed and the deformed configuration.

$$\vec{u}_p = \vec{x}_p - \vec{X}_p = \boldsymbol{F}_M \vec{X}_p - \vec{X}_p = (\boldsymbol{F}_M - \boldsymbol{I}) \cdot \vec{X}_p \tag{7.1}$$

This displacement can be applied to the prescribed nodes using the subroutine FORCDT in MSC Marc. According to Equation 7.1, the subroutine will be coded as

$$\begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = (\boldsymbol{F}_M - \boldsymbol{I}) \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} = \begin{pmatrix} F_{M,11} - 1 & F_{M,21} & F_{M,31} \\ F_{M,12} & F_{M,22} - 1 & F_{M,32} \\ F_{M,13} & F_{M,23} & F_{M,33} - 1 \end{pmatrix} \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix}.$$

These displacements will be applied at every iteration. The model is constructed to apply the load in 5 steps, thus the displacements must be applied in 5 increments resulting in

$$\begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \frac{1}{5} (\boldsymbol{F}_M - \boldsymbol{I}) \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix}.$$

This finalizes the localization of the homogenization scheme. The macroscopic deformation is translated into a loading condition for the RVE.

Extracting the macroscopic stress tensor

The macroscopic stress tensor is given by Equation 2.5 and requires only the reaction forces and the position of the prescribed nodes.

$$\boldsymbol{P}_M = \frac{1}{\mathrm{V}_0} \int_{\Gamma_0} \vec{p} \vec{X} d\Gamma_0 = \frac{1}{\mathrm{V}_0} \sum_{p=1}^{n_p} \vec{f}_p \vec{X} = \frac{1}{\mathrm{V}_0} \sum_{p=1}^{n_p} \begin{bmatrix} f_1 X_1 & f_1 X_2 & f_1 X_3 \\ f_2 X_1 & f_2 X_2 & f_2 X_3 \\ f_3 X_1 & f_3 X_2 & f_3 X_3 \end{bmatrix}_p$$

This summation will result in a 3x3 matrix that represents that macroscopic stress state of the system. When employing periodic boundary conditions, this summation is only required to be taken over the prescribed corner nodes [11].

The information that is available at specific nodes can be accessed in MSC Marc using the subroutine IMPD.

MSC Marc uses second PK to calculate stresses for large strain models. Having found the macroscopic 1st PK stresses, the 2nd PK tensor is easily found by using the macroscopic deformation tensor to transform the stresses

$$\boldsymbol{S_M} = \boldsymbol{F_M^{-1}} \boldsymbol{P_M},$$

resulting in the macroscopic 2nd PK stress tensor.

7.2.2 Macroscopic element

The macroscopic element is the main element that will control the homogenization procedure. The function of this element is to transfer the macroscopic deformation tensor to the RVEs and build the stresses and the tangent on the macroscopic level. This will be done using the subroutine HYPELA2.

The stresses are obtained by starting a RVE simulation that returns the macroscopic stresses. The tangent will be computed using a numerical differentiation scheme.

Calculating the macroscopic tangent

Now that the macroscopic deformation tensor and the macroscopic stress tensor are available, the macroscopic tangent can be computed. The option that is presented in [11] is to condense the stiffness matrix in accordance with a partitioning between prescribed and free nodes.

Unfortunately, this method is not possible in Marc as the full stiffness matrix is not available for manipulation and reassembling a custom full stiffness matrix in marc is highly inefficient. Another option is proposed, namely a numerical differentiation proposed by Miehe [13].

The idea for this forward numerical differentiation is quite simple. Calculate the stress state, apply a small perturbation, calculate the perturbed stress state and from these two we can determine the tangent. Traditionally, this routine should be repeated for every component of the tangent, but Miehe introduced a method to reduce the number of required steps.

The 2nd Piola-Kirchhoff stress tensor can be calculated using either the deformation tensor $(\boldsymbol{F})$ or the Right Cauchy-Green deformation tensor $(\boldsymbol{C})$, which gives the relation for the tangent

$$\mathbb{C} = 2\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{C}} \Leftrightarrow \partial \boldsymbol{S} = \mathbb{C}{:}\frac{1}{2}\partial \boldsymbol{C}.$$

A small perturbation (in the form of a perturbation to the deformation tensor)

$$\boldsymbol{F}^{\epsilon}_{(CD)} = \boldsymbol{F} + \Delta \boldsymbol{F}^{\epsilon}_{(CD)},$$

will be applied. Next the assumption is made that the resulting change in stresses that are caused by the perturbation can be approximated linearly according to

$$\Delta \boldsymbol{S} \approx \boldsymbol{S}\left(\boldsymbol{F}^{\epsilon}_{(CD)}\right) - \boldsymbol{S}(\boldsymbol{F}) = \boldsymbol{S}\left(\Delta \boldsymbol{F}^{\epsilon}_{(CD)}\right) = \mathbb{C}{:}\frac{1}{2}\Delta \boldsymbol{C} = \mathbb{C}{:}\frac{1}{2}(\Delta \boldsymbol{F}^{T}\boldsymbol{F} + \boldsymbol{F}^{T}\Delta \boldsymbol{F}). \quad 7.2$$

Assuming a specific form for the perturbation

$$\Delta \boldsymbol{F}^{\epsilon}_{(CD)} = \frac{\epsilon}{2}\left[\left(\boldsymbol{F}^{-T}\vec{E}_C\right)\otimes\vec{E}_D + \left(\boldsymbol{F}^{-T}\vec{E}_D\right)\otimes\vec{E}_C\right], \tag{7.3}$$

causes terms to fallout nicely due to symmetry properties in the basis vectors. The basis vector ($\vec{E}_{100}$, $\vec{E}_{010}$, $\vec{E}_{001}$) are the unit vectors in Cartesian space, further reducing Equation 7.2 to

$$\boldsymbol{S}\left(\boldsymbol{F}^{\epsilon}_{(CD)}\right) - \boldsymbol{S}(\boldsymbol{F}) \approx \mathbb{C}:\frac{\epsilon}{2}\left[\vec{E}_C\otimes\vec{E}_D + \vec{E}_D\otimes\vec{E}_C\right] = \mathbb{C}\epsilon,$$

From which we find the relation

$$\mathbb{C}^{AB(CD)} = \frac{1}{\epsilon}\left[\boldsymbol{S}^{AB}\left(\boldsymbol{F}^{\epsilon}_{(CD)}\right) - \boldsymbol{S}^{AB}(\boldsymbol{F})\right].$$

The specific choice of $\Delta\boldsymbol{F}^{\epsilon}_{(CD)}$ (in the form of Equation 7.3) allows a quicker estimate for the tangent. Furthermore, the article elaborates on the importance of ϵ and determines that $\epsilon = 1\cdot 10^{-8}$ gives good

One of the problems associated with numerical differentiation is efficiency. The complete problem must be rerun with a small perturbation to the macroscopic tangent. This is reflected in Figure 29.

7.2.3 Detailed Flow-Chart of implementation in Marc

The implementation that is described in sections 7.2.1 and 7.2.2 is summarized here in Figure 29.

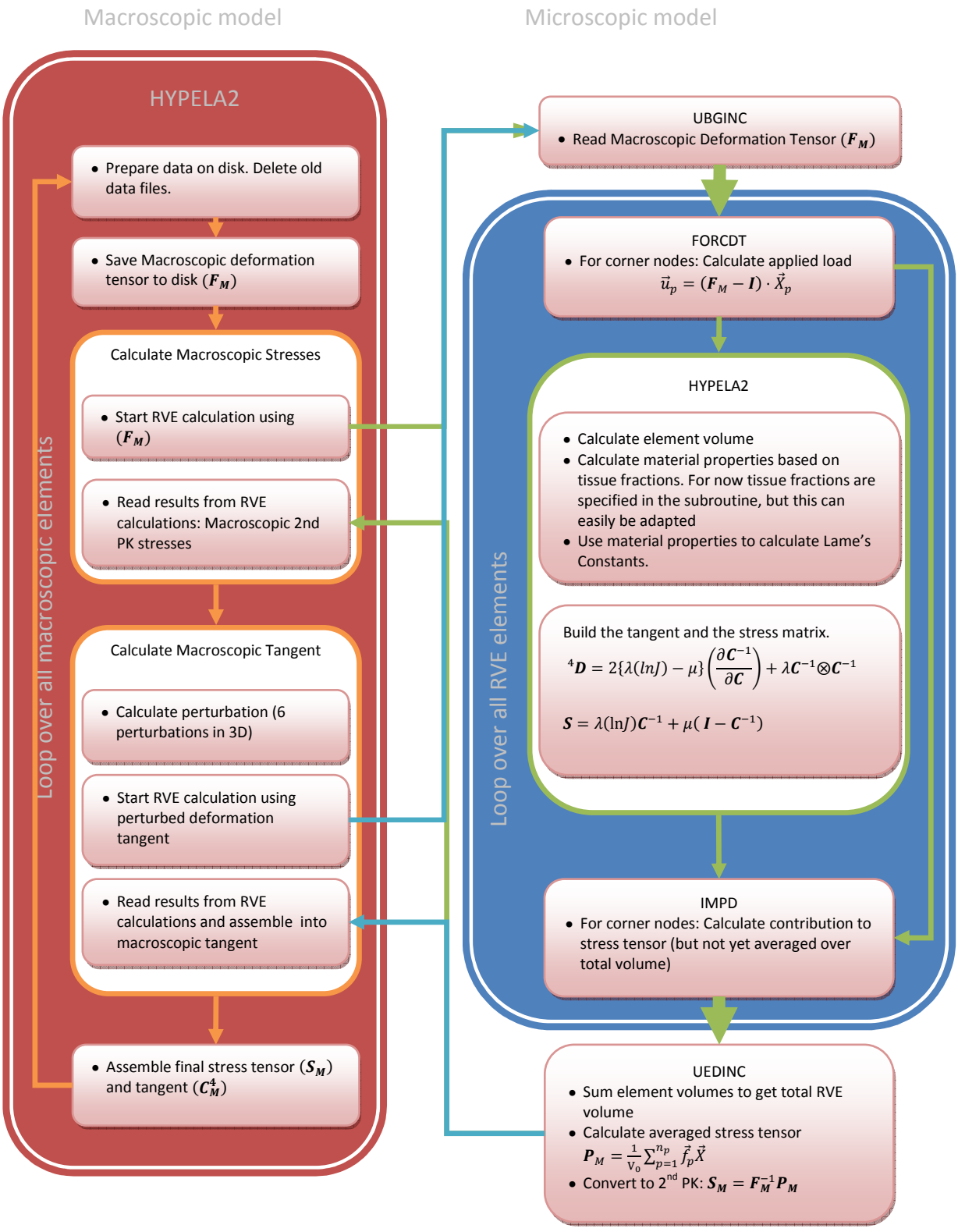


Figure 29 **Flow diagram for a computational homogenization setup.**

7.3 Results

Although the model does show some potential for the application of a computational homogenization scheme, the implementation caused manyproblems.

Unfortunately a fully correct implementation has not been realized. But run times already indicate that even if the run would be successful, an increase in computation time will never be achieved.

As seen in Figure 29 the microscopic simulation must be called 7 (!) times. This greatly reduces the effectiveness of a homogenization approach in a commercial code.

The calculations and disk activity have been minimized to assess the homogenization approach. However, the constant need to start another instance of MSC Marc is a major source of computational overhead. This leads to tremendous run times.

The largest source of overhead lies in the numerical approximation of the macroscopic tangent. It requires 6 microscopic simulations. Condensation of the microscopic stiffness matrix would be a better a solution, but this is unfortunately not possible in MSC Marc.

Timing information for the homogenization model with only 16 elements shows that computational homogenization does not benefit the computation time. The total time for 1 iteration increased to almost 290000 seconds.

The computational homogenization scheme could benefit from extreme parallelization. The global set up and independent RVE calculations can all be sent to a different computer. This requires massive parallelization, available hardware and available software. The question then becomes whether such a solution is cost effective.

In addition to these implementation problems, the problem of loading further deteriorates a homogenized solution. A simulation showed that the different loading that would be required as a result of the homogenization scheme will lead to significantly different results.

8 Summary & Conclusions

In the course of this thesis an investigation was made into the feasibility of accelerating and upscaling the bone ingrowth model developed by A. Andreykiv [1].

A feasibility study on the application of a computational homogenization scheme was performed. The original model does show potential for a homogenization approach. Periodic fluctuations can be observed in the relevant parameter (maximum shear strain). These fluctuations are an indication that a averaged solution may be extracted.

The computational homogenization scheme has been unsuccessfully implemented in MSC Marc. The resulting run times of a simulation are extremely long. These long run times are due to the numerical differentiation scheme that requires many RVE simulations and that for every simulation a complete instance of MSC Marc must be started, resulting in massive computational overhead. In addition the loading cannot be applied in a similar manner and this altered loading leads to different results.

An investigation on model base was also performed. It showed the necessity of the fluid phase and thus the necessity of the biphasic approach to soft-tissue modeling. Although the fluid has been shown to contribute less to the biophysical stimulus it nonetheless had a tremendous impact on the results.

A second investigation into the linearization of the biological model showed that a linearized biological model is not an acceptable approximate for the biological model. The tissue production of fibrous tissue is completely halted and bone production increases.

Investigation of the diffusion of the fibroblasts and mesenchymal stem cells suggested that a one-dimensional approximation of the diffusion was acceptable. Implementation in the subroutine inexplicably led to non-positive-definiteness of the stiffness matrix. It could not be established why this was the case. An estimate of the computation time revealed that a decrease of only 200 seconds is expected, which is relatively little compared to the complete simulation.

In spite of the previous results, a large gain (~65%) in computational speed was gained. This increase in speed is completely attributed to coding optimization. The communication between the biological model and the mechanical model was coded using files on the hard drive and waiting times that were necessary to prevent simultaneous access to these files.

Investigation of the loops that control the disk activity revealed that the waiting times (necessary for monitoring files) could be reduced. The reduction of the waiting time from 1 second to 1 millisecond led to a overall decrease in computation time of about 20%.

The second improvement reduced the total amount of disk writes. The original model wrote the results of an element immediately to the disk. The consequent reading and writing as a

result of this setup up greatly reduced the speed of the simulation. The improved coding stores all results in memory and writes the results of elements to the disk in 1 operation at the end of an iteration. This approach reduced the overall computation time by 65% (compared to the original model)

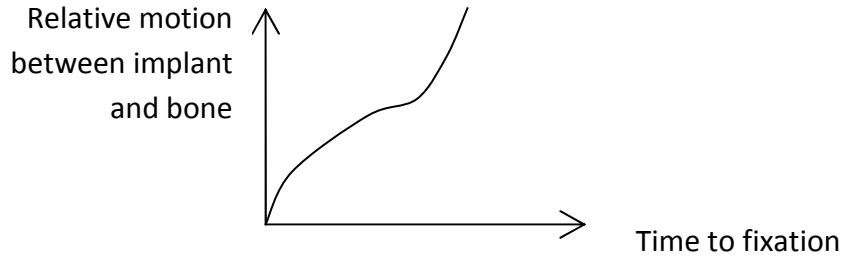
9 Recommendations

The model investigations showed that all implemented features are an essential part of the simulation. The results in this thesis have shown that model simplification does not lead to acceptable results.

The application of computational homogenization also did not result in the desired increase in speed and thus not the upscalability of this system. Clearly homogenization is not the route that should be taken in order to apply the findings of the bone ingrowth simulation on a larger scale.

It is suggested that a new point of view, where the results of these detailed simulations may be incorporated. Such a new approach should be formulated in terms of the requirements in a macroscopic model. In a macroscopic setting, where for example a complete implant is modeled, there is no need for the detailed geometry of the interface. Furthermore there is no need for the stress state or the tissue concentrations of the interface section. What is needed is the response of the interface to a loading and the consequent result for the fixation of the implant.

The following suggestion is made. At the interface between implant and bone, a certain amount of motion will be present after surgery. This amount of motion will differ as a function of location on the surface of the implant. The current model for bone ingrowth can be used to evaluate different applied displacements to assess time to fixation (i.e. micromotion = 0) Correlate stiffness of the system to this. Such an investigation would lead for example to a graph that might resemble the figure below.



Such a result can be implemented in a full scale implant simulation. Along the implant, the local micromotions will give an indication of the time required for fixation. This should be a good indication for the quality of the implant.

In a time dependent analysis, a locally evolving stiffness approach may be adopted, where the evolution of the stiffness ensures that the ‘time to fixation’ is achieved.

It should be noted that the validation of such a model will require experimental data for complete implants.

References

1. Andreykiv, A., *Simulation of bone ingrowth*. 2006, TU Delft: Delft.
2. Liu, X. and G.L. Niebur, *Bone ingrowth into a porous coated implant predicted by a mechano-regulatory tissue differentiation algorithm.* 2008. **7**(4): p. 335-344.
3. Folgado, J., et al., *Influence of femoral stem geometry, material and extent of porous coating on bone ingrowth and atrophy in cementless total hip arthroplasty: an iterative finite element model.* Computer Methods in Biomechanics and Biomedical Engineering, 2009. **12**(2): p. 135 - 145.
4. Perren, S.M. and J. Cordey, *The concept of interfragmentary strain.*, in *Current Concepts of Internal Fixation of Fractures*. 1980, Springer: Berlin.
5. Huiskes, R., et al., *A biomechanical regulatory model for periprosthetic fibrous-tissue differentiation.* Journal of materials science: Materials in medicine, 1997. **8**: p. 785-788.
6. Claes, L.E. and C.A. Heigele, *Magnitudes of local stress and strain along bony surfaces predict the course and type of fracture healing.* Journal of Biomechanics, 1999. **32**: p. 255-266.
7. Khoe, Y.S., *Literature survey: Bone Ingrowth & Upscaling*. 2009, TU Delft.
8. Lacroix, D., L.A. Murphy, and P. Prendergast, *Threedimensional finite element analysys of glenoid replacement prosthesis: a comparison of keeled and pegged anchorage systems.* Journal of Biomechanical Engineering 2000. **122**(4): p. 430-436.
9. Wirth, M.A. and C.A.R. Jr., *Advantages and disadvantages of current glenoid designs*. 2008, University of Texas.
10. Fung, Y.C., *Foundations of Solid Mechanics*. 1965: Prentice Hall.
11. Geers, M.G.D., V.G. Kouznetsova, and W.A.M. Brekelmans, *Multi-scale modelling:Computational homogenization in solid mechanics*. 2007, Eindhoven University of Technology: Eindhoven.
12. Borst, R.d. and L.J. Sluys, *Lecture Notes - Computational Methods in Non linear Solid Mechanics (CT5142)*. 2002.
13. Miehe, C., *Numerical computation of algorithmic (consistent) tangent moduli in large-strain computational inelasticity.* Computational Methods Applied Mechanical Engineering, 1996. **134**: p. 223-240.
14. Fernandez, R., *Natural convection from cylinders buried in porous media*. 1972, University of California.
15. Lewis, R. and B. Screfler, *The Finite Element Method in Static and Dynamic Deformation and Consolidation of Porous Media.* 1998: John Wiley & Sons Ltd.
16. Prendergast, P.J., R. Huiskes, and K. Søballe, *Biophysical stimuli on cells during tissue differentiation at implant interfaces.* Journal of Biomechanics 1997. **30**(6): p. 539-548.
17. Simmons, C. and R. Pilliar, eds. *Bone Engineering Chapter A Biomechanical Study of Early Tissue Formation around Bone-Iterface Implants: The Effects of Implant Surface Geometry*. ed. J.E. Davies. 2000. 369–379.

Appendix A Macroscopic element subroutine

This subroutine is used to control the macroscopic element in the computational homogenization scheme.

```
C<<<<<<<<< Computational Homogenization Macroscopic element >>>>>>>>
C<<<<<<<<< writen by: Y.S. Khoe                              >>>>>>>>
C<<<<<<<<< email: y.s.khoe@student.tudelft.nl                >>>>>>>>
C<<<<<<<<<                                                   >>>>>>>>
C=====================================================
C     HYPELA2
C=====================================================
      SUBROUTINE HYPELA2(D,G,E,DE,S,T,DT,NGENS,N,NN,KC,MATUS,NDI,
     * NSHEAR,DISP,DISPT,COORD,FFN,FROTN,STRECHN,EIGVN,FFN1,
     * FROTN1,STRECHN1,EIGVN1,NCRD,ITEL,NDEG,NDM,NNODE,
     * JTYPE,LCLASS,IFR,IFU)
      IMPLICIT REAL *8 (A-H, O-Z)
      integer statusRVE
      DIMENSION E(1),DE(1),T(1),DT(1),G(1),D(NGENS,NGENS),S(1)
      DIMENSION N(2),COORD(NCRD,NNODE),DISP(NDEG,NNODE),
     * DISPT(NDEG,NNODE),FFN(ITEL,ITEL),FROTN(ITEL,ITEL),
     * STRECHN(ITEL),EIGVN(ITEL,ITEL),FFN1(ITEL,ITEL),
     * FROTN1(ITEL,ITEL),STRECHN1(ITEL),EIGVN1(ITEL,ITEL)
      DIMENSION MATUS(2)
      DIMENSION FMt(3,3),iFMt(3,3),Ei(3,3),Ci(3),Di(3),
     * deltaFcd(3,3),Fcd(3,3),FtEc(3),FtEd(3),CMe(3,3)
      DIMENSION PM(3,3),CM(3,3,3,3)
      CHARACTER tmp1(12)
      include  '../common/concom'

C      Do not read from tissues.dat, but instead set a fixed tissue composition
C      Ususally interface here to communicate tissue fractions to MICRO for
C      example by writing tissue fractions to disk.
C      In this case these settings actually do nothing
      bone=0.1394211E-01
      cartilage=0.1865109E-01
      fibrous=0.8101763E-03

C     [1] Delete old data files
      isystem=system('rm -f MacroFM.dat')

C     [2] Save Deformation Tensor To Disk
C     ---- Write Macroscopic Deformation Tensor from disk - MacroFM.dat ----
C      if (inc.le.0) then C set FM to eye(3) at inc=0
C         do i=1,3
C            do j=1,3
C               FFN(i,j)=0.d0
C               if (i.eq.j) FFN(i,j)=1.d0
C            end do
C         end do
C      end if

      write(0,*) ""
      write(0,*) "======= INC NR: ",inc, "============================="
      write(0,*) "======= ELM NR: ",N(1),"============================="
      write(0,*) "======= WRITE Macroscopic Deformation Tensor ========"
      write(0,'(e9.3,1X,e9.3,1X,e9.3)') (FFN(L,1),L=1,3)
      write(0,'(e9.3,1X,e9.3,1X,e9.3)') (FFN(L,2),L=1,3)
      write(0,'(e9.3,1X,e9.3,1X,e9.3)') (FFN(L,3),L=1,3)
      open(20,file='MacroFM.dat',FORM='FORMATTED')
      write(20,*) FFN
      close(20)


C     [3] Start RVE calculation with macropscopic deformation tensor
      isystem=system('./startRVEcalc')
      statusRVE=0
 100  open(30,file='status.dat',iostat=istat,FORM='FORMATTED')
      if (istat.ne.0) then
         call sleep(1)
         goto 100
      end if
      read(30,*,iostat=istat) statusRVE
```

```
      if (istat.ne.0)  then
         close(30)
         call sleep(1)
         goto 100
      end if
      if (statusRVE.ne.3004) then
         close(30)
         call sleep(1)
         goto 100
      end if

C     [4] Read 2nd PK macroscopic stress results from RVE calculations
!      write(0,*) "================== Read:MacroPM.dat ================="
C      if (inc.eq.0) then
C         call scla(PM,0,3,3,0)
C      else
         open(20,file='MacroPM.dat',FORM='FORMATTED')
         read(20,*) PM
         close(20)
         isystem=system('rm -f MacroPM.dat')
C      end if

      write(0,*) "========= Stress Tensor - 2ND PK           =========="
      write(0,*) (PM(L,1),L=1,3)
      write(0,*) (PM(L,2),L=1,3)
      write(0,*) (PM(L,3),L=1,3)
      write(0,*) "====================================================="

C     [5] Calculate perturbed macroscopic deformation tensor
C         tangent is calculate by numerical differentiation (miehe,96)

!      write(0,*) "======= Macroscopic Tangent              ========"
      eps=1e-8
      call gmtra (FFN,FMt,3,3)
      !call inv3x3(FMt,iFMt,tmp,0)
      call invert(FMt,3,iFMt,0,detF,3)
C     create 3x3 unity matrix
      do i=1,3
         do j=1,3
            Ei(i,j)=0
         end do
         Ei(i,i)=1
      end do

C     [6] Calculate new stresses based on perturbations
C         (6 perturbations required for 3D)
      do K=1,3
         do L=K,3
          ! Create pertubation Fcd
            do i=1,3
               Ci(i)=Ei(i,K) !Ec
               Di(i)=Ei(i,L) !Ed
            end do
            do i=1,3
               FtEc(i)=iFMt(1,i)*Ci(1)+iFMt(2,i)*Ci(2)+iFMt(3,i)*Ci(3)
               FtEd(i)=iFMt(1,i)*Di(1)+iFMt(2,i)*Di(2)+iFMt(3,i)*Di(3)
               do j=1,3
                 deltaFcd(i,j)=(eps/2)*((FtEc(i)*Di(j))+(FtEd(i)*Ci(j)))
                  Fcd(i,j)=FFN(i,j)+deltaFcd(i,j)
               end do
            end do

C           [7] Calculate perturbed stresses and contribution to macroscopic tangent
C           [7.1] Delete old data files
            isystem=system('rm -f MacroFM.dat')
            isystem=system('rm -f PoroRVE_CalcRVE.pid')
C           [7.2] Save Deformation Tensor To Disk
            open(20,file='MacroFM.dat',FORM='FORMATTED')
            write(20,*) Fcd
            close(20)
C           [7.3] Start RVE calculation with macropscopic deformation tensor
            isystem=system('./startRVEcalc')
            statusRVE=0
 101        open(30,file='status.dat',iostat=istat,FORM='FORMATTED')
            if (istat.ne.0) then
               call sleep(1)
               goto 101
```

```
            end if
            read(30,*,iostat=istat) statusRVE
            if (istat.ne.0)  then
               close(30)
               call sleep(1)
               goto 101
            end if
            if (statusRVE.ne.3004) then
               close(30)
               call sleep(1)
               goto 101
            end if
C           [7.4] Read 2nd PK macroscopic stress results from RVE calculations
            open(20,file='MacroPM.dat',FORM='FORMATTED')
            read(20,*) Shat
            close(20)
            isystem=system('rm -f MacroPM.dat')
C           [7.5] Calculate addition to Macroscopic Tangent Matrix
            call gmsub(Shat,PM,CMe,3,3)
            do i=1,3
               do j=1,3
                  CM(i,j,K,L)=CMe(i,j)
               end do
            end do

         end do
      end do

C     [8] Assemble macroscopic stress and tangent into apprpriate vectors for MSC MARC
      S(1)=PM(1,1)
      S(2)=PM(2,2)
      S(3)=PM(3,3)
      S(4)=PM(1,2) !=PM(2,1)
      S(5)=PM(2,3) !=PM(3,2)
      S(6)=PM(1,3) !=PM(3,1)

      D(1,1)=CM(1,1,1,1)
      D(1,2)=CM(1,1,2,2)
      D(1,3)=CM(1,1,3,3)
      D(1,4)=CM(1,1,1,2)
      D(1,5)=CM(1,1,2,3)
      D(1,6)=CM(1,1,3,1)

      D(2,2)=CM(2,2,2,2)
      D(2,3)=CM(2,2,3,3)
      D(2,4)=CM(2,2,1,2)
      D(2,5)=CM(2,2,2,3)
      D(2,6)=CM(2,2,3,1)

      D(3,3)=CM(3,3,3,3)
      D(3,4)=CM(3,3,1,2)
      D(3,5)=CM(3,3,2,3)
      D(3,6)=CM(3,3,3,1)

      D(4,4)=CM(1,2,1,2)
      D(4,5)=CM(1,2,2,3)
      D(4,6)=CM(1,2,3,1)

      D(5,5)=CM(2,3,2,3)
      D(5,6)=CM(2,3,3,1)

      D(6,6)=CM(3,1,3,1)
      do i=1,5
         do j=i+1,6
            D(j,i)=D(i,j)
         end do
      end do

      RETURN
      END
```

Appendix B Microscopic element subroutine

This subroutine is used to control the microscopic element in the computational homogenization scheme.

```
C     ================================================================
C      UBGINC
C     Read all required files (only needs to be done once)
C     ================================================================
      subroutine ubginc(inc,incsub)
      implicit real*8 (a-h,o-z)

      dimension FM(3,3),PBNodeID(4)
      common/macro/FM,PBNodeID,PBRForces

      if (inc.eq.0) then
         write(0,*) "======= Read Input ==============================="
         write(0,*) "======= Macroscopic Deformation Tensor ==========="
C        ---- Read Macroscopic Deformation Tensor from disk - MacroFM.dat ----
         open(20,file='MacroFM.dat',FORM='FORMATTED')
         read(20,*) FM
         close(20)
         write(0,'(e9.3,1X,e9.3,1X,e9.3)') (FM(N,1),N=1,3)
         write(0,'(e9.3,1X,e9.3,1X,e9.3)') (FM(N,2),N=1,3)
         write(0,'(e9.3,1X,e9.3,1X,e9.3)') (FM(N,3),N=1,3)
C        ---- Read Prescibed Node ID - PrescribedNodeID.dat ----
         open(20,file='PrescribedNodeID.dat',FORM='FORMATTED')
         read(20,'(e12.6)') PBNodeID
         close(20)
      end if

      RETURN
      END
C     ================================================================
C      FORCDT (OLDSTYLE tables - array of applied displacements are incremental)
C     take macroscopic deformation tensor and use it to define
C     boundary condition on corner nodes.
C     ================================================================
      SUBROUTINE FORCDT (U,V,A,DP,DU,TIME,DTIME,NDEG,NODE,
     * UG,XORD,NCRD,IACFLG,INC, IPASS)
      IMPLICIT REAL *8 (A-H, O-Z)
      DIMENSION U(NDEG),V(NDEG),A(NDEG),DP(NDEG),DU(NDEG),UG(NDEG),
     * XORD(NDEG)

      dimension FM(3,3),PBNodeID(4)
      common/macro/FM,PBNodeID,PBRForces

      if (inc.eq.0) then
C      du=(FM-I)*X !watch notation A(column,row)
       DU(1)=((FM(1,1)-1)*XORD(1)+FM(2,1)*XORD(2)+FM(3,1)*XORD(3))/5
       DU(2)=(FM(1,2)*XORD(1)+(FM(2,2)-1)*XORD(2)+FM(3,2)*XORD(3))/5
       DU(3)=(FM(1,3)*XORD(1)+FM(2,3)*XORD(2)+(FM(3,3)-1)*XORD(3))/5
C       write(0,*)
C       write(0,*) "======= Applied Node Disp Node:",NODE," ============"
C       write(0,'(e9.3,1x,e9.3,1x,e9.3)') (DU(N),N=1,3)
C      Applied Displacement DU is saved in memory, thus will be applied every iteration
C      Also it is automatically saved for which node DU is applied.
      end if

      RETURN
      END


C     ================================================
C      HYPELA2
C     ================================================
      SUBROUTINE HYPELA2(D,G,E,DE,S,T,DT,NGENS,N,NN,KC,MATUS,NDI,
     * NSHEAR,DISP,DISPT,COORD,FFN,FROTN,STRECHN,EIGVN,FFN1,
     * FROTN1,STRECHN1,EIGVN1,NCRD,ITEL,NDEG,NDM,NNODE,
     * JTYPE,LCLASS,IFR,IFU)
      IMPLICIT REAL *8 (A-H, O-Z)
      DIMENSION E(1),DE(1),T(1),DT(1),G(1),D(NGENS,NGENS),S(1)
```

```
      DIMENSION N(2),COORD(NCRD,NNODE),DISP(NDEG,NNODE),
     * DISPT(NDEG,NNODE),FFN(ITEL,ITEL),FROTN(ITEL,ITEL),
     * STRECHN(ITEL),EIGVN(ITEL,ITEL),FFN1(ITEL,ITEL),
     * FROTN1(ITEL,ITEL),STRECHN1(ITEL),EIGVN1(ITEL,ITEL)
      DIMENSION MATUS(2),RC(3,3),RCinv(3,3),Cse(3,3,3,3),
     * secondPK(3,3),FFNt(3,3),Green(3,3),elmvol(10000)
      DIMENSION A(3,3),At(3,3),Ainv(3,3)
      common/tissues/bone,cartilage,fibrous
      common/elvol/elmvol
      include  '../common/concom'
      include  '../common/cdominfo' !contains: iprcnm, inc

C     Calculate element Volume, used in UEDINC to determine total volume
      call elmvar(78,N(1),NN,KC,elmvol(N(1)))

C     Do not read from tissues.dat, but instead set a fixed tissue composition
C     Later convert this to read settings from macro properties
      bone=0.1394211E-01
      cartilage=0.1865109E-01
      fibrous=0.8101763E-03

C     calculate the material properties (Young's Modulus, Poisson's Ratio)
      call matextract(et,xmu,bone,cartilage,fibrous)

C     Temperature effects are not relevant here
      do i=1,6
         G(i)=0.d0
      end do

C     Calculate Elasticity matrix and stresses
C     FFN1 is def tensor at current time step
      call gmtra(FFN1,FFNt,3,3)
      call gmprd(FFNt,FFN1,RC,3,3,3)
      call inv3x3(RC,RCinv,det_RC,0)
      detJ=sqrt(det_RC)

C     Lame constants
      dlambda=xmu*et/((1.d0+xmu)*(1.d0-2.d0*xmu))
      dmu=et/(2.d0*(1.d0+xmu))
C     Elasticity matrix (neo-Hookean material )
      do i=1,3
         do j=1,3
            do k=1,3
               do l=1,3
                  Cse(i,j,k,l)=dlambda*(RCinv(i,j)*RCinv(k,l))+
     *             (dmu-dlambda*log(detJ))*
     *             (RCinv(i,k)*RCinv(j,l)+RCinv(i,l)*RCinv(k,j))
               end do
            end do
         end do
      end do
      D(1,1)=Cse(1,1,1,1)
      D(1,2)=Cse(1,1,2,2)
      D(1,3)=Cse(1,1,3,3)
      D(1,4)=Cse(1,1,1,2)
      D(1,5)=Cse(1,1,2,3)
      D(1,6)=Cse(1,1,3,1)

      D(2,2)=Cse(2,2,2,2)
      D(2,3)=Cse(2,2,3,3)
      D(2,4)=Cse(2,2,1,2)
      D(2,5)=Cse(2,2,2,3)
      D(2,6)=Cse(2,2,3,1)

      D(3,3)=Cse(3,3,3,3)
      D(3,4)=Cse(3,3,1,2)
      D(3,5)=Cse(3,3,2,3)
      D(3,6)=Cse(3,3,3,1)

      D(4,4)=Cse(1,2,1,2)
      D(4,5)=Cse(1,2,2,3)
      D(4,6)=Cse(1,2,3,1)

      D(5,5)=Cse(2,3,2,3)
      D(5,6)=Cse(2,3,3,1)

      D(6,6)=Cse(3,1,3,1)
```

```
      do i=1,5
         do j=i+1,6
            D(j,i)=D(i,j)
         end do
      end do

C     Second Piola-Kirchhoff stress
      do i=1,3
         do j=1,3
            if (i.eq.j) then
               secondPK(i,j)=dlambda*log(detJ)*RCinv(i,j)+
     *          dmu*(1.d0-RCinv(i,j))
            else
               secondPK(i,j)=dlambda*log(detJ)*RCinv(i,j)-
     *          dmu*RCinv(i,j)
            end if
         end do
      end do
      S(1)=secondPK(1,1)
      S(2)=secondPK(2,2)
      S(3)=secondPK(3,3)
      S(4)=secondPK(1,2)
      S(5)=secondPK(2,3)
      S(6)=secondPK(1,3)
      RETURN
      END

C     ================================================================
C      IMPD
C     Here the following is calculated: sum(f*X) for the prescribed nodes
C     in order to get the macroscopic stress tensor, it must still be divided
C     by the original volume which is done in UEDINC.
C     ================================================================
      subroutine impd(lnode,dd,td,xord,f,v,a,ndeg,ncrd)
      implicit real *8 (a-h, o-z)
      dimension dd(ndeg), td(ndeg), xord(ncrd), f(ndeg), v(ndeg),
     * a(ndeg), lnode(2)
      dimension FM(3,3),PBNodeID(4),PBRForces(3,3),PBRForcesTMP(3,3)
      common/macro/FM,PBNodeID,PBRForces

      if ((lnode(1).eq.PBNodeID(1)).or.(lnode(1).eq.PBNodeID(2)).or.
     *     (lnode(1).eq.PBNodeID(3)).or.(lnode(1).eq.PBNodeID(4))) then
         do i=1,3
            do j=1,3
               PBRForces(i,j)=PBRForces(i,j) + f(i)*xord(j)
            end do
         end do
      end if


      return
      end

C     ================================================================
C      UEDINC
C     At the end of the simulation calculate the macroscopic stress
C     tensor and the tangent and store them.
C     ================================================================
      subroutine uedinc(inc,incsub)
      implicit real*8 (a-h,o-z)

      dimension elmvol(10000),PBRForces(3,3),MacroPK(3,3)
      common/elvol/elmvol
      common/macro/FM,PBNodeID,PBRForces

C     Calculate total volume
      do i=1,10000
         V0=TotalVolume+elmvol(i)
      end do

      if (inc.eq.5) then
         do i=1,3
            do j=1,3
               MacroPK(i,j)=(1/V0)*PBRForces(i,j)
            end do
         end do
```

```
C        only do volume averaging at last incement
C         write(0,*) "======= Calculate Macroscopic Properties ========"
C        -------------- Calculate Macroscopic Stress -------------------
C         write(0,*) "======= F*XORD @ Prescribed nodes========"
C         write(0,*) (MacroPK(N,1),N=1,3)
C         write(0,*) (MacroPK(N,2),N=1,3)
C         write(0,*) (MacroPK(N,3),N=1,3)
C         write(0,*) "-------------------------------------"

C        ------------ Write Macroscopic Results to disk ----------------
C         write(0,*) "======= Save to:MacroPM.dat              ========"
         open(20,file='MacroPM.dat',FORM='FORMATTED')
         write(20,*) MacroPK
         close(20)
C         write(0,*) "================================================="

C        ---------------------------------------------------------------
C        ---------------- Write status file to disk --------------------
 101     open(20,file='status.dat',iostat=istat,FORM='FORMATTED')
!        maak een bestand op de disk aan waar '3004' instaat
!        voor een kopieer opdracht uit ipv schrijven naar disk
         isystem=system('rm -f status.dat; cp statusReady status.dat')
!         if (istat.ne.0) then
!            call sleep(1)
!            goto 101
!         end if
!         status3004=3004
!         write(20,*) status3004
!         close(20)
C        ---------------------------------------------------------------
      end if

      RETURN
      END


C     ===================================================================
C      MATEXTRACT
C     determine material properties
C     input :  tissue fractions of this element;
C              bone,cartilage,fibrous
C     output:  Young Modulus  [et]
C              Poisson Ratio  [xmu]
C
C     ===================================================================
      subroutine matextract(et,xmu,bone,cartilage,fibrous)
      implicit real *8 (a-h,o-z)
      include  '../common/elmcom'
C     young modulus
      YoungGr=0.2e+6
      YoungF=2e+6
      YoungC=10e+6
      YoungB=6000e+6
      YoungMarrow=2e+6

C.    Poisson ratio
      xMuGr=0.1667d0
      xMuF=0.1667d0
      xMuC=0.1667d0
      xMuB=0.3d0
      xMuMarrow=0.1667d0

      if (mats.ne.1.and.mats.ne.2) then
         write(0,*) 'ERROR: material identifier is not known in HYPELA'
         call quit(1234)
         return
      end if

      if (mats.eq.1) then
         xmu=bone*xMuB+cartilage*xMuC+
     *    fibrous*xMuF+(1.d0-(bone+fibrous+cartilage))*xMuGr
         et=bone*YoungB+cartilage*YoungC+
     *    fibrous*YoungF+(1.d0-(bone+fibrous+cartilage))*YoungGr
      else
         et=YoungMarrow
         xmu=xMuMarrow
      end if
```

```
        return
        end
```

Appendix C Python script to apply servo links in the RVE

This python script can be called from MENTAT to create the servo links that determine the periodic boundary conditions. It requires 6 sets of nodes named: Top, Bottom, Left, Right, Front and Back. Note that the script does not automatically remove duplicate links. Duplicate periodic boundary conditions must be prevented by careful selection of the nodal sets.

```
#!/usr/bin/env python
#
from py_mentat import *
def main():
    print "=== Start Python Script ==="
    # 6 sets: Top Bottom Left Right Front Back
    for L in range(3):
       CornersSetName="Corners"
       if L==0:
          TiedSetName="Top"
          TiedToSetName="Bottom"
          LinkDirectionName="Z"
       elif L==1:
          TiedSetName="Left"
          TiedToSetName="Right"
          LinkDirectionName="X"
       elif L==2:
          TiedSetName="Front"
          TiedToSetName="Back"
          LinkDirectionName="Y"

       n = py_get_int("nsets()")
       for i in range(1,n+1):
         id = py_get_int("set_id(%d)" % i)
         sn = py_get_string("set_name(%d)" % id)
         if sn == TiedSetName:
              TiedSetID=id
         elif sn == TiedToSetName:
              RetainedSetID=id
         elif sn == "Corners":
              CornersSetID=id

       if L==0:
         print "* Identify Corners"
         Corners=[[],[],[],[]]
         Corners=[[-1]*8,[-1]*8,[-1]*8,[-1]*8]
         CornersFound=0
         for i in range(8):
           Corners[0][i]=py_get_int("set_entry(%d,%d)" % (CornersSetID, i+1))
           Corners[1][i]=round(py_get_float("node_x(%d)" % Corners[0][i]),8)
           Corners[2][i]=round(py_get_float("node_y(%d)" % Corners[0][i]),8)
           Corners[3][i]=round(py_get_float("node_z(%d)" % Corners[0][i]),8)
           # Find CornerNodes
           Check=[Corners[1][i],Corners[2][i],Corners[3][i]]
#           print "Node: ",Corners[0][i],"| coords: ",Check
           CornersName=round(py_get_float("Rtmp")*2,8)
           if Check==[0,0,0]:
               u000=Corners[0][i]
               CornersFound=CornersFound+1
#                 print "u000 found, node: %d" %Corners[0][i]
           elif Check==[CornersName,0,0]:
               u100=Corners[0][i]
               CornersFound=CornersFound+1
#                 print "u100 found, node: %d" %Corners[0][i]
           elif Check==[0,CornersName,0]:
               u010=Corners[0][i]
               CornersFound=CornersFound+1
#                 print "u010 found, node: %d" %Corners[0][i]
           elif Check==[CornersName,CornersName,0]:
               u110=Corners[0][i]
               CornersFound=CornersFound+1
#                 print "u110 found, node: %d" %Corners[0][i]
```

```
            elif Check==[0,0,CornersName]:
                u001=Corners[0][i]
                CornersFound=CornersFound+1
#                 print "u001 found, node: %d" %Corners[0][i]
            elif Check==[CornersName,0,CornersName]:
                u101=Corners[0][i]
                CornersFound=CornersFound+1
#                 print "u101 found, node: %d" %Corners[0][i]
            elif Check==[0,CornersName,CornersName]:
                u011=Corners[0][i]
                CornersFound=CornersFound+1
#                 print "u011 found, node: %d" %Corners[0][i]
            elif Check==[CornersName,CornersName,CornersName]:
                u111=Corners[0][i]
                CornersFound=CornersFound+1
#                 print "u111 found, node: %d" %Corners[0][i]
          if CornersFound!=8:
            print "!!! ERROR not all corners found, only found %d corners" %
CornersFound
          print "* Prescribed Corner Node ID's saved to [PrescribedNodeID.dat]"
          filename=open("PrescribedNodeID.dat","w")
          filename.write("%e\n%e\n%e\n%e" % (u100,u010,u001,u000))
          filename.close()
          print "Prescribed Node ID's: ",u000," ",u100," ",u010," ",u001

        print "=== Create Periodic BC's (Servo Links) ==="
        print "Nodes from ",TiedSetName, " to ", TiedToSetName, " in
",LinkDirectionName," direction"
        # Create Servo Links
        nSym1 = py_get_int("nset_entries(%d)" % TiedSetID)
        nSym2 = py_get_int("nset_entries(%d)" % RetainedSetID)

        if nSym1!=nSym2:
          print "!!! Sets do not contain same mount of nodes thus cannot be linked"
          print "!!! Set 1: ",TiedSetName," %d Nodes" % nSym1
          print "!!! Set 2: ",TiedToSetName,"%d Nodes" % nSym2
          return 1

        nSym=nSym1
        Wall1=[[],[],[],[]]
        Wall1=[[-1]*nSym,[-1]*nSym,[-1]*nSym,[-1]*nSym]
        Wall2=[[],[],[],[]]
        Wall2=[[-1]*nSym,[-1]*nSym,[-1]*nSym,[-1]*nSym]
        MapWall=[[],[]]
        MapWall=[[-1]*nSym,[-1]*nSym]
        # Obtain Coordinates for all nodes in both sets
        for i in range(0,nSym):
            # Wall1=TiedSet, Wall2=RetainedSet
            Wall1[0][i]=py_get_int("set_entry(%d,%d)" % (TiedSetID, i+1))
            Wall1[1][i]=py_get_float("node_x(%d)" % Wall1[0][i])
            Wall1[2][i]=py_get_float("node_y(%d)" % Wall1[0][i])
            Wall1[3][i]=py_get_float("node_z(%d)" % Wall1[0][i])
            Wall2[0][i]=py_get_int("set_entry(%d,%d)" % (RetainedSetID, i+1))
            Wall2[1][i]=py_get_float("node_x(%d)" % Wall2[0][i])
            Wall2[2][i]=py_get_float("node_y(%d)" % Wall2[0][i])
            Wall2[3][i]=py_get_float("node_z(%d)" % Wall2[0][i])

        k=0
        for i in range(0,nSym):
          if LinkDirectionName=='X':
            # Create Servo Links in X-direction
            for j in range(0,nSym):
                if (Wall1[2][i]==Wall2[2][j]) and (Wall1[3][i]==Wall2[3][j]):
                   MapWall[0][k]=Wall1[0][i]
                   MapWall[1][k]=Wall2[0][j]
          if LinkDirectionName=='Y':
            # Create Servo Links in Y-direction
            for j in range(0,nSym):
                if (Wall1[1][i]==Wall2[1][j]) and (Wall1[3][i]==Wall2[3][j]):
                   MapWall[0][k]=Wall1[0][i]
                   MapWall[1][k]=Wall2[0][j]
          if LinkDirectionName=='Z':
            # Create Servo Links in Z-direction
            for j in range(0,nSym):
                if (Wall1[1][i]==Wall2[1][j]) and (Wall1[2][i]==Wall2[2][j]):
                   MapWall[0][k]=Wall1[0][i]
                   MapWall[1][k]=Wall2[0][j]
```

```
          k=k+1
       print "created 3x%d=%d servo links" % (k,3*k)

       for i in range(0,k):
          if MapWall[0][i]==-1:
            print "!!! ERROR in Retained Node"
          py_send("*new_link *link_class servo *tied_node %d *servo_nterms 3 *tied_dof
1 " % MapWall[0][i])
          py_send("*servo_ret_dof 1 1 1 1")
          py_send("*servo_ret_coef 1 1 -1 1")
          if LinkDirectionName=='Z':
             # uTIE=u111-u110+uRET
             py_send("*link_class servo *servo_ret_node 1 %d %d %d" %
(u001,u000,MapWall[1][i]))
          if LinkDirectionName=='Y':
             # uTIE=u100-u110+uRET
             py_send("*link_class servo *servo_ret_node 1 %d %d %d" %
(u010,u000,MapWall[1][i]))
          if LinkDirectionName=='X':
             # uTIE=u010-u110+uRET
             py_send("*link_class servo *servo_ret_node 1 %d %d %d" %
(u100,u000,MapWall[1][i]))
          py_send("*copy_link")
          py_send("*link_class servo *tied_dof 2")
          py_send("*servo_ret_dof 1 2 2 2")
          py_send("*copy_link")
          py_send("*link_class servo *tied_dof 3")
          py_send("*servo_ret_dof 1 3 3 3")
          if (i % 10)==0:
             print "--created links: %d --" % ((i+1)*3)
#     print "Create servo link for fixed node"
#     py_send("*new_link *link_class servo *tied_node %d *servo_nterms 3 *tied_dof 1 "
% u001)
#     py_send("*link_class servo *servo_ret_node 1 %d %d %d" % (u110,u111,u000))
#     py_send("*servo_ret_dof 1 1 1 1")
#     py_send("*servo_ret_coef 1 1 -1 1")
#     py_send("*copy_link")
#     py_send("*link_class servo *tied_dof 2")
#     py_send("*servo_ret_dof 1 2 2 2")
#     py_send("*copy_link")
#     py_send("*link_class servo *tied_dof 3")
#     py_send("*servo_ret_dof 1 3 3 3")

    print "=== Finished Creating Servo Links ==="
    print "=== End Python Script             ==="



    return

if __name__ == '__main__':
    py_connect('',40007)
    main()
    py_disconnect()
```

Appendix D Batchwrites – e.g. for stimuli.dat files

One of the issues is that files on the disk cannot be written simultaneously by the computer system. The solution is to use 1 file for each domain. This appendix explains the concept of the batchwrite system.

This example is worked out for the stimuli.dat files, but its general concept is also applied to the tissues.dat files that store the tissue fractions of each element.

The biophysical stimuli are calculated in the mechanical simulation. This simulation is parallelized over 6 domains. In the computer, each domain creates an array that can contain all elements (the values in the array are initially set to zero). This implies that the 6 separate domains each have an array with the length of the number of elements. If calculations on an element are complete, the element number and its properties are saved in the array. At the end of the iteration the 6 domains each have an array in many zeros, but the entries that contain information are marked with the element number. These 6 arrays are saved to the disk at the end of the iteration.

The reading of the stimulus files is done by the biological model. The model reads the complete 6 arrays into the memory and scans them once to identify the entries which have been marked by the element number. Together a complete array can be compiled that contains all data for all elements.