A differential equation approach to the solution of non-linear equations of a static physical system

Blankevoort, T.P.F.

T.P.F. Blankevoort

A differential equation approach to the solution of non-linear equations of a static physical system

Bachelorscriptie, 26 september 2011

Scriptiebegeleiders:
prof.dr.ir. B. Koren
en
dr.ir. L. Blankevoort

Mathematisch Instituut, Universiteit Leiden

A differential equation approach to the solution of non-linear equations of a static physical system

Tijmen P.F. Blankevoort, *Student,* Leendert Blankevoort, *Associate Professor* and Barry Koren, *Professor*

***Abstract*—This thesis investigates the Newton-Raphson solution routine used in the quasi-static multibody model of diarthrodial joints (Kwak et al. 2003). We suggest a more reliable method for solving non-linear systems of equations that is based on numerical methods for solving differential equations, and show, given the physical nature of the model, that it will find a solution more reliably. This new method is called the Taylored Backward Euler method. It is very reliable, but takes more time to find a solution than the Newton-Raphson algorithm.**

I. Introduction

THIS thesis investigates methods by which a physical system of non-linear equations can be solved. This is done within the context of the Joint Model software, created by Kwak et al. (2000). The techniques that will be described are more generally applicable to any system of non-linear equations with certain physical properties.

The Joint Model program is a piece of software that is used to simulate the mechanical behaviour of human joints. It is a multibody model that can quite accurately represent any diarthrodial joint. With it, it is possible to calculate the forces on bones and ligaments for a given joint pose and external loads. Because the model is specifically tailored to finding only equilibrium positions of joints, it is very fast at calculating the forces in such situations. Several researchers in the field of Orthopaedics, including for example Cohen et al. (2003), Borotikar (2009) and Dvinskikh et al. (2011), have succesfully used the software and its underlying mathematical model for practical research purposes. Models of joints have succesfully helped researchers evaluate orthopaedic surgical procedures and could possibly be used to calculate the effects of surgery on individual patients as well (Cohen et al. 2003). This thesis will give a brief introduction to this model in order to place the remainder of this thesis in a suitable context. For a thorough description of the model, we refer to the article by SD Kwak et. al. (2000). Their ”Quasi-Static Multibody Model of Diarthrodial Joints” will hereafter simply be referred to as the joint model.

Along with other researchers, Cohen (2001) reported problems with the algorithm that finds the equilibrium state in a given model with initial values. Sometimes the algorithm does not converge automatically to the required equilibrium state, meaning the model parameters have to be adjusted in such a way that the algorithm does converge. This can be a very tedious process, especially when the calculations are done on very large datasets which can take hours to complete. The existing program uses a damped Newton-Raphson root finding algorithm. The question was raised if the damped Newton-Raphson algorithm could be improved such that it would converge better and be more stable, while keeping a comparable rate of convergence. This is the research question for this thesis.

In this paper we will explain why the Newton-Raphson method does not converge sometimes, after which we will introduce a different approach to root finding that works specifically for physical equations in which stable equilibria have to be found. The problem is cast into a system of first order differential equations, after which numerical differential equation methods are used to follow the direction of the force and find a stable equilibrium.

The algorithm found this way is derived from backward Euler. We will show that this ‘Taylored Backward Euler method’ is preferred over the Newton-Raphson method due to its stability and reliability for this specific application, although it has a higher computation time.

II. The model

The joint model distinguishes between material bodies, which have six degrees of freedom (DOFs), three rotations and three translations, and particles with only three translational DOFs. Material bodies can represent bones of a diarthrodial joint, while particles are typically embedded in soft tissue structures, such as tendons and ligaments, to allow the wrapping around articular or bony surfaces. The bones and articular surfaces are represented by 3D-mesh models. The bodies are connected to each other with elastic links. These links come in different flavors; surface-to-surface contact, particle-to-surface contact, ligament links and tendon links. In essence these links are the structures that apply the forces on the bodies in the model. For surface-to-surface contact the applied force is calculated from the overlap of the surfaces. These are compressive forces. In ligament links the ligament force is calculated from the link length, by means of a specified function. These forces can be compressive or tensile. In tendon links the force is constant, as supplied by the user. External forces and moments can also be applied to any body in the model, both relative to the body-fixed coordinate system and relative to the ground-fixed coordinate

T.P.F. Blankevoort is a student at the department of Mathematics, Leiden University, Leiden, e-mail: (TiRune@gmail.com)

L. Blankevoort is with the Academic Medical Center Amsterdam and the University of Amsterdam. B. Koren is with the Centrum Wiskunde & Informatica and Leiden University

B. Koren is with the Centrum Wiskunde & Informatica and Leiden University

Manuscript turned in September 25, 2011

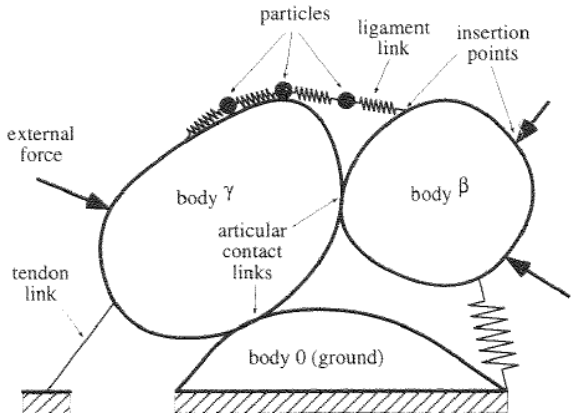


Fig. 1. The 3D multibody model prescribes the bones of a joint as material bodies (6 DOFs); the interactions between the bones include articular contact links, and ligament and tendon links. Wrapping of tendons or ligaments around bone or articular surfaces is modeled by imbedding particles (3 DOFs) within these soft tissue structures (Kwak 2000).

system. As most models only use the ground-fixed coordinate system for external forces we can limit our perspective to this. The geometry of the model is defined by the surfaces of the articular and bony surfaces, the insertions of the ligament links and tendon links and the point of application of the external forces. See figure 1 for a schematic overview of the model and the terminology used.

By using the positional and rotational information, we can calculate the forces that the ligaments, tendons and contact forces apply on the bodies. The forces and moments on the body are decomposed into 3 forces along the coordinate axes and 3 moments about the coordinate axes. These forces change by motion of the body, making them dependent on the DOFs of motion. The model will calculate the rotations and translations of the bodies in the multi-body system for given external forces and/or moments. Hence, the DOFs of motion are the dependent variables. The system as a whole is at rest, i.e. in an *equilibrium position*, if the sum of all forces and the moments for each DOF, is equal to zero for each body. This results in a system of non-linear equations as for every body β is required that

$$\mathbf{f}^{\beta}(\boldsymbol{\theta}, \mathbf{a}) = \sum_i \mathbf{f}_i^{\beta}(\boldsymbol{\theta}, \mathbf{a}) = 0, \tag{1}$$

with i a counter for the links attached to body β and $\mathbf{f}$ the three-dimensional vector with all forces that act on the body. The force-vector depends on the translations $\mathbf{a}$ and rotations $\boldsymbol{\theta}$ of body β and the bodies connected to β. In the same way we have

$$\mathbf{m}^{\beta}(\boldsymbol{\theta}, \mathbf{a}) = \sum_i \mathbf{m}_i^{\beta}(\boldsymbol{\theta}, \mathbf{a}) = 0, \tag{2}$$

with $\mathbf{m}$ the three-dimensional moment vector. For notational simplicity we will define a generalized force vector $\mathbf{f}$ where

$$[\mathbf{f}] = [\mathbf{f}^1\, \mathbf{m}^1\, \mathbf{f}^2\, \mathbf{m}^2\, \ldots\, \mathbf{f}^n\, \mathbf{m}^n]^T, \tag{3}$$

with n the total number of bodies.

The fundamental question is: Given a starting configuration of the bodies, what are the positions and orientations of the bodies in which the sum of the six separate forces and moments equals zero for each body. The model only works with these equilibrium positions, which give the necessary information for medical purposes. The forces and moments are calculated by the software from the translation and rotation vectors. Because of the complexity of modelling the surface-to-surface contacts, no explicit functions are given for the forces and moments. Rather, they are calculated on an ad-hoc basis given the position and orientation of the bodies.

III. The problem with the current solution method

The aforementioned question seems an easy problem, as we only have a set of non-linear equations that have to be solved. What is often the case with such practical problems is that the Newton-Raphson iteration procedure is applied to find solutions. With this idea in mind the original solution method was programmed as a standard Netwon-Raphson solution method. Newton-Raphson does converge fast when the initial position values are close to the projected equilibrium, but when starting farther away it may fail to converge. To remedy this, a relaxation factor α was introduced to adjust the step-size of the iterations by a set scaling factor. This gave the following relaxed Newton-Raphson iteration step that is used in the existing joint model:

$$\mathbf{x}_{new} = \mathbf{x}_{old} - \alpha\, \mathbf{J}(\mathbf{x}_{old})^{-1}\, \mathbf{f}(\mathbf{x}_{old}), \tag{4}$$

with $\mathbf{x}$ the vector that contains the rotations θ and translations $\mathbf{a}$ of every body, $\mathbf{f}$ the generalized force vector that contains the forces and moments in each direction for every body, and $\mathbf{J}$ the Jacobian $\frac{\mathrm{df}}{\mathrm{dx}}$. This alleviates the convergence problem somewhat, but it introduces a parameter α that has to be chosen for every run. If the parameter is not chosen correctly the algorithm will either take longer to converge or will not converge at all. For some problems, when the initial position is taken farther away from the equilibrium the algorithm also does not converge, no matter what α is chosen. We proceed by considering a specific problem for which (4) does not converge. This will later on be used as a benchmark for the new solution method.

A. The knee model

We will consider the model used by Borotikar (2009, Fig 10b, 10c) to simulate the machanics of the knee, for the purpose of predicting the forces in the ligaments with certain activities. It is a rather simplified model of the human knee joint with only 12 DOFs. When running this model from its initial position, it converges in 16 steps to the equilibrium solution. When looking at the graph of the residual sum of squares of the generalized force vector a small caveat becomes apparent already (Fig. 2).

On the first iteration the residual shoots up high, indicating that the algorithm moves large distance away from the

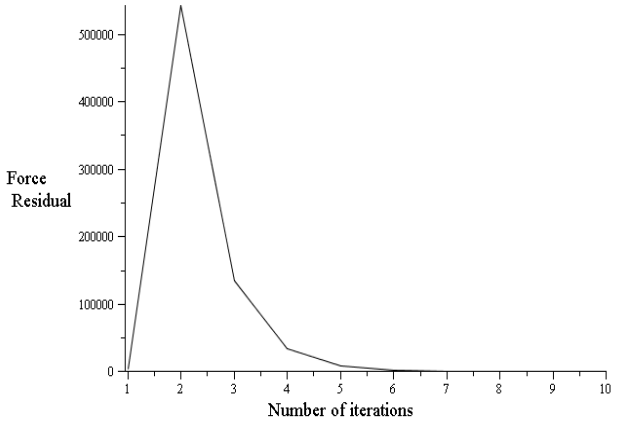


Fig. 2. The behavior of the residuals of the force vector for each Newton-Raphson iteration made on the initial configuration for the Borotikar (2009) knee model. We can see that the force first increases by a great amount before the solution converges in 14 steps to the equilibrium position. This behavior is caused by the Jacobian matrix to be close to singular.

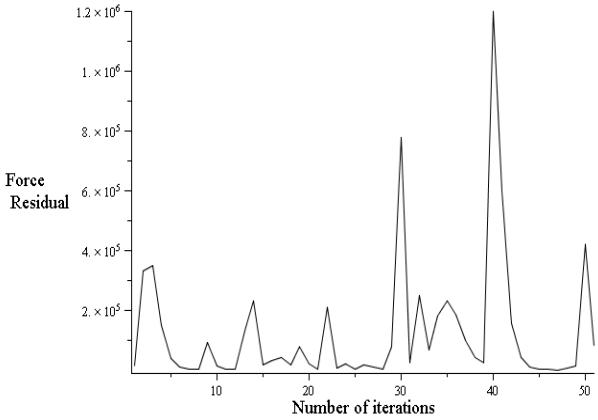


Fig. 3. The resdiuals for the first 50 out of 1000 iterations of the Borotikar (2009) knee model when the Femur is flexed 90 degrees. For the full 1000 iterations this spiky and oscillatory behavior persists and the solution does not converge.

solution at first, before eventually converging. This is usually unwanted behavior as it might interfere with finding an equilibrium solution that is close to the starting configuration. Newton-Raphson ignores the distance to the original solution, although in practice a knee move to a equilibrium position that is close. Moving far away from the equilibrium would also be counterproductive to the speed of the algorithm. Later on we will see why the Newton-Raphson algorithm causes solutions to sometimes shoot away radically. For this simple example this is only a minute problem, as the algorithm converges nonetheless, but there are also cases when there is no convergence at all.

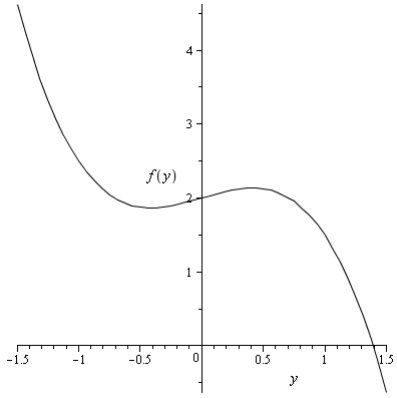
Upon choosing this new equilibrium position as a starting point, we flex the femur 90 degrees. From this new position the algorithm does not converge, no matter what relaxation factor is chosen. The residual graph is very spiky and oscillates a lot (Fig. 3). This is a very typical situation for the joint model, as any time it does not converge this same type of behavior can be perceived in the residual graph. The only way to make the algorithm converge properly is by increasing the flexion by small steps at a time and making the algorithm converge to those positions subsequently, gradually making it to the full 90 degrees flexion. Letting the algorithm converge this way is a tedious trial-and-error process. An approach - for problems for which an equilibrium solution exists - that always converges without fidgeting around with any parameter would greatly improve the applicability of the software.

B. Problems with Newton Raphson

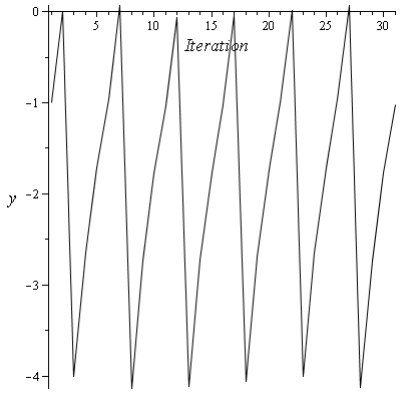
The Newton Raphson algorithm, albeit widely used for solving systems of non-linear equations, has some general problems associated with it. The significant problems that relate to its joint model application will be sketched here by two simple examples. The first example is a third degree polynomial function. (Fig. 4). We can clearly see that there is one equilibrium, i.e. the position where the force is equal to 0. This single equilibrium is located next to a small hill and a following concavity on the left of the equilibrium position. If we start the Newton-Raphson algorithm to the left of this concavity, the algorithm gets stuck in the local minimum (Fig. 4). Although the exact numbers differ each iteration, the oscillatory behavior persists, and the algorithm does not converge. This oscillatory pattern also appeared previously in the residual graph of the knee-model (Fig. 3), and we believe the cause of the problem is the same.

There is a second algorithmic problem associated with this example. The Jacobian for a one-dimensional problem like this is simply the differential of the function $f(x)$. At the bottom part of the concavity, a local minimum, the derivative is zero. This means that when the solution starts very close to the local minimum, the derivative is very small, and hence the inverse is very large. This causes the algorithm's solution path to jump away excessively from its current location, causing a spike in the residual count. The same happens in higher dimensions, where the Jacobian can become singular and inverting it gives very large values in the matrix. This same sort of spike behavior was observed in the initial convergence of the knee-model in the previous paragraph. We think that the same underlying mechanism causes the non-convergence in both cases. Local minima disrupt the convergence of the Newton-Rahpson method.

The second example used to illustrate the third problem with Newton-Raphson is that of a pendulum attached to a fixed point with a particle on the other end. A vertical force is applied to the particle, resulting from simulated gravity. Instead of the pendulum being infinitely stiff, we use a spring to connect the particle to the fixed point. Thus we get a spring-pendulum setup (Fig 5). The functions and constants that define this example are given in Appendix A. There are two equilibrium positions for this problem, the obvious one where the pendulum is at rest below the fixed point, and the unstable one where the pendulum rests directly above the fixed point. Now when we place the starting position of the Newton-Raphson method close to the unstable fixed point, it will converge to it (Fig. 6). Thus when there are unstable equilibria in systems of equations of a physical system, coupled with the



(a) Function of the force against the y-value of the particle



(b) y-value of particle as function of Newton-Raphson iteration

Fig. 4. (a) The function $f(y) = -y^3 + \frac{1}{2}y + 2$. Starting the Newton-Raphson algorithm from around $y = -1$ there is a small area with a local minimum and maximum that the solution has to cross before arriving at the equilibrium position. This causes problems with the Newton-Raphson method as can be seen in (b). The solution heavily oscillates around the local minimum. although only the first 30 iterations are shown here, the same behavior persists for all subsequent iterations that were checked. It is important to note that the initial value of -1 is not of particular significance, as starting from almost all values close to the points that the oscillation comes close to gives the same behavior.

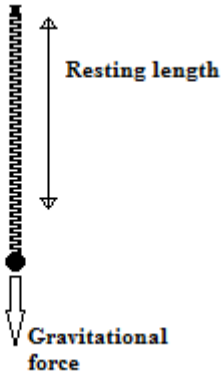


Fig. 5. The spring pendulum setup. The particle at the bottom can move around the whole two-dimensional space while the spring exerts a force on it dependent on the current length of the spring and its resting length. The spring is attached to one fixation point that is kept constant. An external gravitational force is applied to the particle so that there is one stable equilibrium (shown) and one unstable equilibrium (straight above the fixation point). The equations for this example can be found in Appendix A.

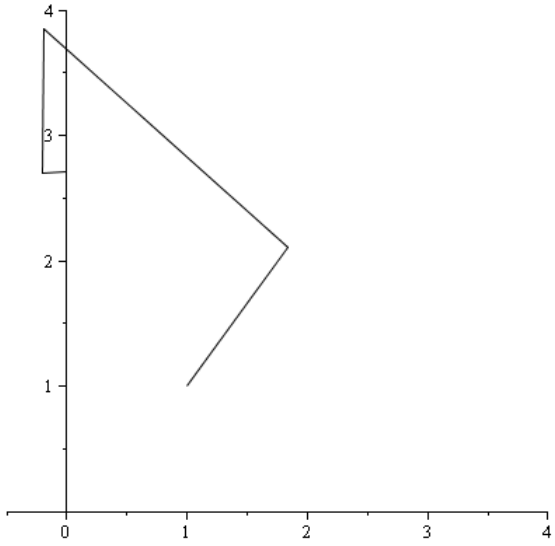


Fig. 6. The iteration steps for the Newton-Raphson method as applied to the spring-pendulum problem. The initial position of the particle is $(x, y) = (1, 1)$. It took 8 steps before the algorithm converged to the equilibrium within 10^{-4}. In this case the equilibrium position is the one above the point where the spring is fixed. Since gravity works in the negative-y direction this equilibrium position is unstable; the force points away from it.

sometimes spiking behavior of the Newton-Rahpson algorithm as described above, it is very well possible that it will converge to one of such unstable equilibria. In any setup where a physically meaningfull solution to a systems of equations like this has to be found, convergence to an unstable equilibrium may occur. We have yet to confirm the existence of such unstable equilibria in any of the models of real joints that were made, but the possibility itself is of great concern.

It is of importance that all three of these problems; the spiking behavior, oscillatory behavior around minima and convergence to unstable equilibria are corrected, whilst keeping a good speed of convergence. We will now turn to the main focus of this thesis, a novel differential equation approach to find the equilibrium of a physical system that is represented by a system of non-linear equations

IV. Differential equation methods for solving physical non-linear systems of equations

A very important observation is that the joint model represents a system with inherent physical properties. This makes a more specific root-finding algorithm possible. When bodies in the model are far from their prospective equilibrium position, the forces on the body will be directed towards that same position. Moving a knee cap far out of the knee, without the ligaments breaking, will force it back in. Furthermore, it is expected that if we slowly move a body in the direction of

the force that is exerted on it, it will gradually move to an equilibrium position. This is all due to the elastic nature of the ligaments and the articular contacts. If we turn the system of equations into a first order system of differential equations, we can make use of these properties, and use numerical methods that exist for solving systems of differential equations to find our solution. To simplify notation we will use the general force vector (3) to write our problem in the form

$$\mathbf{f}(\mathbf{x}) = \mathbf{0},$$

with $\mathbf{f}$ a vector-valued function and $\mathbf{x}$ containing all translations and rotations. The differential equations approach to solving this system simply requires us to add a first order time derivative term to the equation:

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} + \mathbf{f}(\mathbf{x}) = \mathbf{0}, \text{starting with } \mathbf{x} = \mathbf{x}_0. \tag{5}$$

At an equilibrium position of this system $\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t}$ and hence $\mathbf{f}(\mathbf{x})$ will be zero. It is possible so scale this dependance by multiplying $\mathbf{f}$ by a matrix with constants on the diagonal, but for simplicity we assume this to be the identity matrix.

It is important to note that the time variable introduced here has no correlation to actual physical time. The function $\mathbf{f}$ is also independent of the time variable. This means that we are not interested in the time-accuracy of the differential equation solving method we will use. We are only interested in finding the equilibrium position that the solution starting at $\mathbf{x}_0$ will converge to.

We noted earlier that the physical nature of the equations dictates that any body, or combination of bodies that moves too far away from the ground will have a force exerted on it that pulls it back. This is simply because all ligaments that govern the forces have a resting length on which the ligaments exert no force. The more is deviated from this resting length, the more force the ligament links will exert on the connected bodies in the direction of its resting length. In the joint model the ligaments also do not break, so all solutions are bounded. On top of this, if we do not use forces that are relative to any of the body-fixed coordinate systems, we get a Jacobian with no imaginary eigenvalues. This makes sure that there are no limit cycles that our solution can get stuck in. The Bolzano-Weierstrass theorem tells us that for our bounded solution there must be an accumulation point, which can only be a stable equilibrium when no limit cycles exist. Thus a solution started anywhere in the solution-space will eventually converge to a stable equilibrium if we follow the direction of the force, given that the solution method stays stable and no body-fixed coordinates are used for the external forces.

We will try to find such a stable equilibrium by applying numerical integration methods. There are two types of such methods, implicit and explicit.

A. Explicit methods: Forward Euler

The explicit methods have an upside in that they are computationally inexpensive, as they describe a preset function that calculates the the solution position from the previous position.

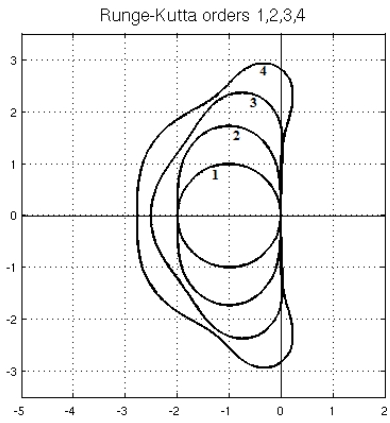


Fig. 7. The stability regions in the complex number domani, in which the eigenvalues of the Jacobian, multiplied by step size have to be for the numerical method to be stable. The four stability regions increase in size as the order of the method goes up. The smallest region corresponds to Runge-Kutta of order 1, which is equal to forward Euler.

Calculating the next position only requires the evaluatin of the generalized force function a few times at each time level. There are a lot of different flavors to explicit methods, but the only differences between them are time-accuracy of the path and stability of the solution. For our application we do not require time-accuracy, as we are only interested in the end point of the path and not the precise road it travels towards it. As the Jacbian will have no imaginary eigenvalues, we can take an explicit method for which the stability region does not contain a finite part of the imaginary axis (stability regions 1 and 2 in Fig. 7). For this context it suffices to simply pick the easiest method, that of Forward Euler, which has stability region 1 in Fig. 7. The update rule for this method is:

$$\mathbf{x}_{new} = \mathbf{x}_{old} + h\,\mathbf{f}(\mathbf{x}_{old}). \tag{6}$$

In words, what this update rule does is taking a small step in the direction of the force vector from the starting position. This is very intuitive. When a net-force is exerted on a mass in a certain direction, it will move in that direction. Here h is the step size. With it we can control the stability of the solution. Naturally a large step-size will mean that the numerical approximation of our solution will move to the equilibrium point faster, however the larger the steps we take the less stable our approximation may become. Theory dictates that h should be chosen small enough so that the eigenvalues of the Jacobian multiplied by h all lie within the stability region of the Forward Fuler method (region 1 in Fig. 7).

Initial test results with this method on the previously introduced simple problems described in Section III seemed very promising. For the pendulum problem the approximation converges to the equilibrium point at a speed comparable to that of the Newton-Raphson algorithm (Fig. 8a). On top of this there are a few pleasant characteristics of this method as compared to Newton-Raphson. The solutions will exhibit no spiking behavior as long as the method remains stable. This is because the solutions follow a stable path defined by the

continuous system of differential equations. We have argued before that no limit-cycles exist in the model, so the solutions will also not oscillate in such a way that no solution can be found. Convergence to unstable solutions is also impossible, as the force vectors close to such unstable equilibria point away from them, and Forward Euler follows their direction. As a bonus, the approximation moves somewhat like you would expect a physical solution of the system to behave, meaning that if there are multiple equilibria it will more likely converge to the one that an actual joint would move to. This is no surefire trait, but more than the Newton-Raphson method it does take the proximity of the equilibria and the forces of the system into account when finding a solution. Choosing h through trial and error is possible, but the process is easily automated by calculating the Jacobian every few steps and determining its most negative, i.e. largest in absolute value, eigenvalue. This can then be used for the following update rule to decide what h should be:

$$h = -\frac{c}{\min_i(\lambda_i)},\ 0 < c < 2 \tag{7}$$

with λ_i the eigenvalues. This update rule ensures the stability of the Forward Euler algorithm (Vuik et al. 2006) The parameter c can be chosen to scale h relative to the in absolute value largest eigenvalue. When $c = 2$ the Forward Euler update rule is on the edge of stability. This is not advised as (7) is only a linear approximation of its stability. The closer h is chosen to the upper bound of stability, the more the solution will show unstable oscillatory behavior. For our experiments we used $c = 1.5$. Note that if there would be only positive eigenvalues, the physical problem itself would be unstable, which is what cannot occur here. Positive eigenvalues do occur, but the solution algorithm will rapidly move away into a region where only negative eigenvalues occur. The choice $c = 1.5$ also places h close to its highest possible value, which can increase the speed of convergence of the algorithm, although expensive computations have to be done in order to find the Jacobian and its eigenvalues.

This Forward Euler method with stability stability rule (7) was implemented in the joint model. The new algorithm took many millions of iterations to converge even for the simplest of problems. This took an impractical amount of time, making the Forward Euler method no good alternative. The problem lies in the stiffness of the articular contact links and the particle-to-surface contacts of the model. The Jacobian represents the stiffness of the system. The larger the ratio of the largest over the smallest eigenvalues is in absolute value (the condition number), the larger the stiffness.

For the articular contact links, slight overlap between two bodies causes the force exerted on both to be very large. The eigenvalues of the Jacobian become very large in absolute value as a large disparity in position causes a large difference in force. We know from the aforementioned stability criterion that in absolute value very large eigenvalues make the approximation unstable unless a very small h is chosen. If we increase the stiffness of the spring in the pendulum-spring example, and apply the same Forward Euler algorithm we can show how this stiffness affects the speed of the solution (Fig. 8b). Even for such a simple problem the stiffness increases the amount of iterations needed for convergence to an impractical amount. So the explicit family of algorithms will not function well for our problem.

B. Implicit methods: Backward Euler

As we have seen, explicit methods are impractical due to the stiff nature of joint models. This is a very well-known problem of explicit integration methods. To make this more mathematically formal we introduce the concept of A-stability. We can analyze general stability for stiff problems by means of the test equation $y' = k\,y$ with $k \in \mathbb{C}$. The solution for this equation is $y(t) = e^{k\,t}$. This solution approaches zero as $t \to \infty$ for all k for which $\Re(k) is finite and negative$. Any numerical method that also exhibits this behavior is called A-stable. The stability domain of A-stable methods contain the entire left half of the complex number domain. For A-stable methods we know that they do not show the problem with stiffness in equations, thus from our practical point of view any method we implement has to be A-stable. Explicit methods, like Forward Euler, are not A-stable. But an implicit method, like Backward Euler *is* A-stable (Dalhquist 1963).

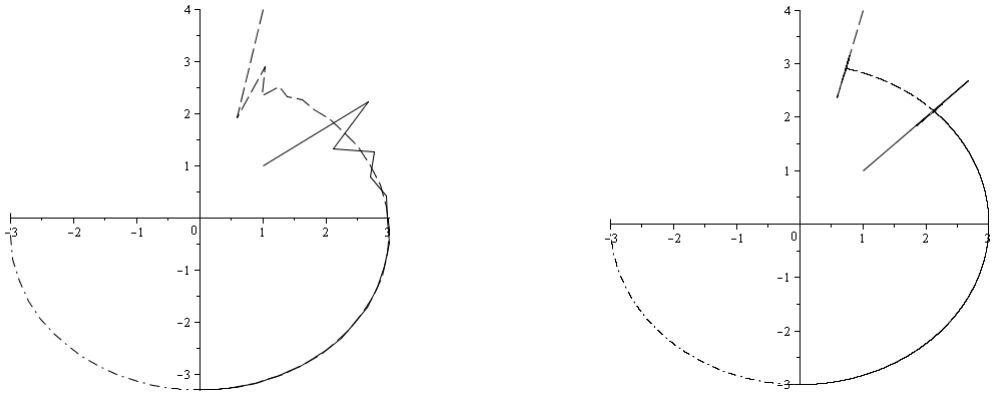
The Backward Euler method seems like a good candidate as the method is stable for all negative eigenvalues, no matter how large they are in absolute value. The update rule for the Backward Euler method reads:

$$\mathbf{x}_{new} = \mathbf{x}_{old} + h\,\mathbf{f}(\mathbf{x}_{new}). \tag{8}$$

The implicitness of this method lies in the fact that $\mathbf{f}$ now has to be evaluated in the new position vector instead of the old. For non-linear vector functions $\mathbf{f}$ this requires an algorithm that can solve non-linear systems of equations. Usuallyis the Newton-Rahpshon method is used for this, precisely the algorithm we are trying to find an alternative for. Also, specifically for the joint model software, the function $\mathbf{f}$ is not explicitly known. This makes solving (8) implicitly hard. Thus we have to find an estimate for the $\mathbf{f}(\mathbf{x}_{new})$ term in (8). We can make a Taylor expansion of $\mathbf{f}(\mathbf{x}_{new})$ around $\mathbf{f}(\mathbf{x}_{old})$ and presume that the function is smooth enough for this to be a good approximation for small changes of $\mathbf{x}$. Taking the first two terms of the Taylor expansion gives us the following update rule.

$$\mathbf{x}_{new} = \mathbf{x}_{old} + h\,(\mathbf{I} - h\,\mathbf{J}(\mathbf{x}_{old}))^{-1}\,\mathbf{f}(\mathbf{x}_{old}). \tag{9}$$

with $\mathbf{I}$ the identity matrix. This rule can also be found by taking (8) and running one iteration step of Newton-Raphson on it to solve it (see Appendix B). A more informative way of looking at this rule is as a hybrid method combining the backward Euler method and the Newton-Raphson method. If we take $0 < h \ll 1$ then $(\mathbf{I} - h\,\mathbf{J}) \approx \mathbf{I}$ and the update rule is similar to forward Euler with a small step size h. On the other hand if we take $h \gg 1$ then $\mathbf{I} - h\,\mathbf{J} \approx -h\,\mathbf{J}$ and the update rule reverts back to the Newton-Rapshon method (4) with parameter $\alpha = 1$. By changing h we can hence continuously combine both methods.



(a) Forward Euler normal spring-pendulum (b) Forward Euler stiff spring-pendulum

Fig. 8. (a) Solution trajects of the Forward Euler algorithm applied on the spring-pendulum problem. Values chosen were $F_g = 3$, $L_0 = 3$ and $K = 10$. The solid curve starts on (1,1), the dashed curve starts on (1,4) and the dashdot curve starts on (-3,0.1). All three converge to the equilibrium within 10^{-4} distance in less than 100 steps.The computing time is comparable to that of the Newton-Raphson algorithm. (b). The same problem but stiff, with $K = 1000$. Here, for these solutions to get within 10^{-4} distance of the equilibrium around 100.000 iterations are needed. Because of the stiffness of the spring, the solution oscillates a couple of times before it is at the point where the spring is at resting length. From there on the algorithm takes very small steps, as the stiffness causes the force to increase a lot when the position deviates a small bit from the resting length of the spring.

This linearized Backward Euler update rule is more than just a convenient way of alternating between both methods. When h is fixed at a reasonably small value, making the update rule behave similarly to a Forward Euler method, for stiff directions the terms in the Jacobian become large and (9) behaves more like Newton-Raphson. This mitigates the problems the Forward Euler method has with stiffness. The other way around, choosing a high h makes the Backward Euler rule resemble the Newton-Raphson method along with the problems of converging to unstable equilibria. The condition of the identity matrix $\mathbf{I}$ to $-h\,\mathbf{J}$ has a regularizing effect; it eliminates possible zero eigenvalues of the Jacobian $\mathbf{J}$. For Jacobians with no positive real eigenvalues, our case, no new zero eigenvalues can be introduced by the addition. The parameter h is used to tune the regularization.

Because the method for large h resembles the original Newton-Raphson method, the Taylored Backward Euler update rule (9) is best used with small h, mimicking the behavior of Forward Euler, but balancing the solution path with Newton-Raphson when the difference in forces forces become relatively high.

We applied this algorithm to the spring-pendulum problem (Fig. 9). The algorithm converged to the target stable equilibrium in about 30 steps from many of different positions in the solution space. This is quite comparable to the Newton-Raphson algorithm that converged in 5 to 10 steps on average. However, the Backward Euler update rule makes sure that for small h it always converges to the stable equilibrium. The speed of the algorithm can be increased by choosing larger h as the force residual goes down. The solution is expected to be close to the target equilibrium when the force residual is low, which means that changing the update rule into a Newton-Raphson method by increasing h could be profitable.

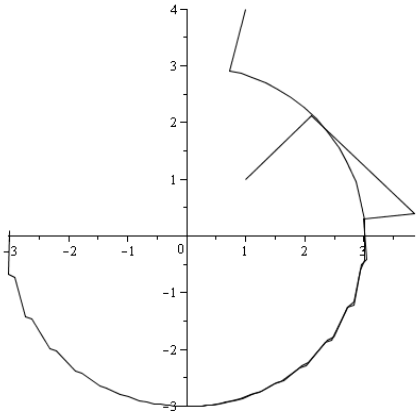


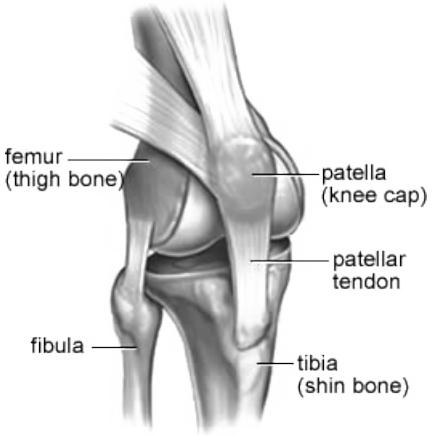
Fig. 9. Results for the backward Euler method applied on the stiff spring-pendulum problem. We chose $h = 0.3$ so the method would be more similar to Forward Euler than to Newton-Raphson. As in Fig. 8b we take $K = 1000$. We see from the figure that the algorithm preforms very well, despite the high stiffness. The moment the solution deviates from the resting length path of the spring, the Jacobian becomes larger and compensates for the difference. The algorithm converges to within 10^{-4} of the equilibrium solution in 30 steps.

C. Implementation in the joint model software

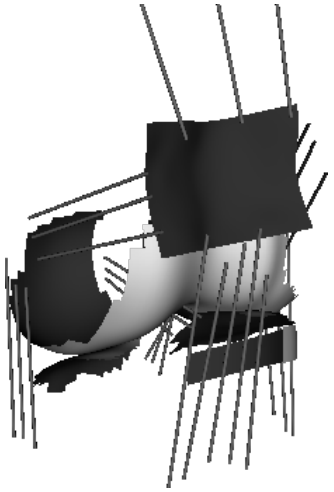
The Taylored backward Euler method was implemented in Joint Model the software and a few initial tests were ran (Fig. 10). For all of the runs done with the algorithm we took $h = 0.1$ a small parameter. When the force residual became smaller than 1 we switched the parameter to $h = 100$, so the method would behave like Newton-Raphson. Taylored backward Euler was tested on the knee model for flexion degrees between 10 and 90. For all of the tests that were done with the algorithm, it converged very smoothly (Fig. 10d).

V. Discussion

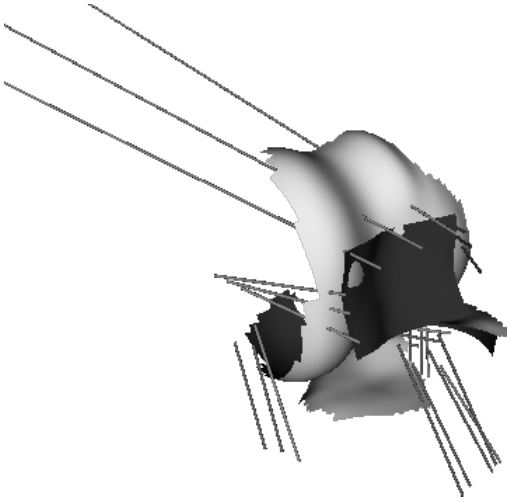
In this article we have described the joint model and the issues it had with the Newton-Raphson method that solves the



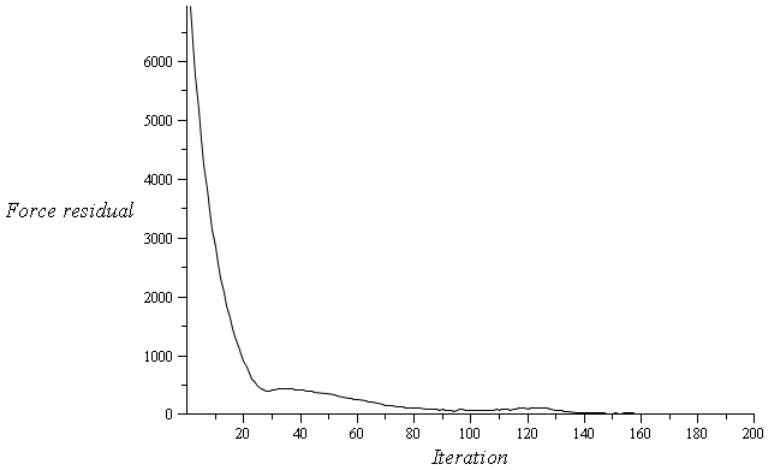
(a) Schematic drawing of the knee



(b) Knee model in extension



(c) Knee model converged in 90 degrees flexion



(d) Residual graph

Fig. 10. **(a)** Schematic model of the knee (Seif Medical Graphics). **(b)** Borotikar (2009) knee model that we use as a benchmark for the algorithm. This picture shows the knee from its initial extended state. The picture was taken directly from the joint model software. **(c)** The same knee, but now converged to the 90 degrees flexion situation. **(d)** Residual graph of the Taylored backward Euler method, as applied to the knee flexon problem. This shows how the algorithm behaves when it gets from situation (b) to situation (c). It converges to within 10^{-4} residual close to the equilibrium in 283 steps. A great improvement over the Newton-Raphson algorithm shown in Fig. 2.

model. Newton-Raphson only reliably converges when it starts close to equilibrium solution. We also argued that Newton-Raphson might converge to unstable equilibria without warning, although for practical purposes of the joint model only stable equilibria are of interest. In this light we introduced a new method, the Taylored Backward Euler method for solving non-linear systems of equations. Taylored Backward Euler relies heavily on the physical properties of the equations that are inherent to the modelling of joints. The force needs to stand roughly in the direction of the equilibria on points far away, making sure the solutions of the differential equations are bounded. Also, if the differential system of equations defines limit-cycles in the space, it is possible that the algorithm converges to these and goes into an oscillatory loop. This can only occur if the Jacobian has imaginary eigenvalues, which we do not expect if no external body-fixed coordinates are used. If these conditions are met, the algorithm is sure to converge to a stable equilibrium.

The smoothness with which Taylored Backward Euler converges comes from the fact that it roughly follows a solution of the systems of differential equations, which defines a smooth path. The stability comes from the stability of the Backward Euler method, that always remains stable, even for stiff equations. Some care has to be taken into interpreting this however, as we do not use the full Backward Euler method, but a taylored version of it. The theory also shows in practice if we look at the results of the spring-pendulum problem and the knee model (Fig. 9 and 10d). From these results we can argue that this new algorithm outpreforms Newton-Raphson in terms of reliability. We have proven that it will converge to a stable equilibrium, and because it follows the direction of the force, the solution is also one that is more likely to be physically feasible. There is a cost attached to this however, as we do take more iterations to converge to an equilibrium than Newton-Raphson. For the spring-pendulum problem Newton-Raphson often converges in 4-5 steps as

opposed to 30 steps. When applied on lower flexions in the knee model, the Taylored Backward Euler algorithm also took about 4-5 times more steps to converge than Newton-Raphson. The calculation cost of the iterations are roughly the same. If the high reliability of the Taylored Backward Euler method is not needed, i.e. when the initial values are expected to be close to an equilibrium, the Newton-Raphson algorithm will still outpreform it. In some cases when stiffness is not an issue the Forward Euler algorithm we described might be faster than both other methods. Because every step of the Forward Euler algorithm we only needs one evaluation of the force vector. We have payed no further attention to this method as all models of joints have too high a degree of stiffness. This article has by no means thoroughly shown that the Taylored Backward Euler algorithm works with the joint model for all cases. We merely proved that the concept of the algorithm can work, and argued why theoretically the algorithm is oft a good idea. More testing is needed with different and more complex models to show that it works. It will also be informative to find a more accurate estimate of the trade-off between reliability and speed between this method and Newton-Raphson. An argument was made about the stability of the algorithm based on the non-Taylored version of the Backward Euler method, but it is unsure how the Tayloring of the implicit term affects the stability. More research has to be done to investigate this. Other improvements over Newton-Raphson, like line-search (Box et al. 1969) or the quasi-Newton methods like Broyden's method (Broyden 1965) and SR1 algorithm (Conn et al. 2003) were also not discussed as they more or less exhibit the same problems that Newton-Rhapson has with initial values that start far away from the solution.

VI. Conclusion

The Taylored Backward Euler method to solve the non-linear system of equations of the joint model is a better alternative to the Newton Raphson method. We have proven theoretically that it will always converge and that its convergence path is more stable. The trade-off is a higher computation time compared to the Newton-Raphson algorithm.

Appendix A
Equations for spring-pendulum problem

The equations of the forces on the particle in the spring pendulum in terms of the x and y position are:

$$F_x(x,y) = \frac{\text{sign}(y)\,x}{y\sqrt{1+\frac{x^2}{y^2}}}\,K\,(L_0 - \sqrt{x^2+y^2}), \qquad (10)$$

$$F_y(x,y) = \frac{\text{sign}(y)}{\sqrt{1+\frac{x^2}{y^2}}}\,K\,(L_0 - \sqrt{x^2+y^2}) - F_g. \qquad (11)$$

K is the spring constant, indicating the stiffness of the spring. The higher it is, the stiffer the spring. L_0 is the resting length of the spring and F_g is the gravitational force that acts on the particle.

Appendix B
Derivation of the Taylored Backward Euler scheme

We collect all terms of (8) on one side and define:

$$\mathbf{N}(\mathbf{x}_{new}) \equiv \mathbf{x}_{old} - \mathbf{x}_{new} + h\,\mathbf{f}(\mathbf{x}_{new}) = 0. \qquad (12)$$

We can apply Newton-Raphson iteration to $\mathbf{N}(\mathbf{x_{new}})$, with l the counter for each iteration. This gives us the following iteration protocol for solving the above system of equations:

$$\mathbf{x}^{l+1}_{new} = \mathbf{x}^{l}_{new} - \left(\frac{\mathrm{d}\mathbf{N}(\mathbf{x}^l_{new})}{\mathrm{d}\mathbf{x}_{new}}\right)^{-1}\mathbf{N}(\mathbf{x}^l_{new}),\ l = 0,1,2,\ldots. \qquad (13)$$

Now the Jacobian of $\mathbf{N}$ can be calculated from its definition:

$$\frac{\mathrm{d}\mathbf{N}(\mathbf{x}^l_{new})}{\mathrm{d}\mathbf{x}_{new}} = -\mathbf{I} + h\frac{\mathrm{d}\mathbf{f}(\mathbf{x}^l_{new})}{\mathrm{d}x_{new}}. \qquad (14)$$

Taking $\mathbf{x}^{l=0}_{new} = \mathbf{x}_{old}$ we get equation (9).

References

[1] Broyden, CG. A class of methods for solving noninear simultaneaous equations. Mathematics of Computation. 1965;19(92):577-593.

[2] Borotikar BS. Subject Specific Computational Models of the Knee to Predict Anterior Cruciate Ligament Injury. PhD thesis, 2009, Cleveland State University, Cleveland, OH, USA.

[3] Box MJ, Davies D, Swann WH. Non-Linear optimisation Techniques. Oliver & Boyd 1969.

[4] Cohen ZA. Analysis of Diagnostic Tools and Treatment Methods for Osteoarthritis in the Patellofemoral Joint: Vivo Methods Employing Magnetic Resonance Imaging. PhD Thesis, 2001, Columbia University, New York, NY, USA.

[5] Cohen ZA. Henry JH, McCarthy DM, Mow VC, Ateshian GA. Computer simulations of patellofemoral joint surgery. Patient-specific models for tuberosity transfer. Am J Sports Med. 2003 Jan-Feb;31(1):87-98.

[6] Conn AR, Gould NIM, Toint PhL. Convergence of quasi-Newton matrices generated by the symmetric rank one update. Mathematical Programming. 1991;50(1):177-195

[7] Dahlquist G. A special stability problem for linear multistep methods. BIT 1963; 3(1): 27-43.

[8] Dvinskikh NA, Blankevoort L, Strackee SD, Grimbergen CA, Streekstra GJ. The effect of lunate position on range of motion after a four-corner arthrodesis: a biomechanical simulation study. J Biomech. 2011 Apr 29;44(7):1387-92.

[9] Kwak SD, Blankevoort L, Ateshian GA. A mathematical formulation for 3D quasi-static multibody models of diarthrodial joints. Comput Methods Biomech Biomed Engin. 2000; 3(1):41-64

[10] Vuik C, van Beek P, Vermolen F, van Kam K. Numerieke Methoden voor Differentiaal-vergelijkingen. VSSD 2006: Delft, The Netherlands.